University of Cape Town

EEE4022S - Final Year Project

Spatial Calibration of multi-sensor systems

Michael Katsoulis
KTSMIC005

11 November 2020

Plagerism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Michael Katsoulis
11 November 2020

Contents

Abstract

This paper seeks to develop and implement a method of spatially calibrating a lidar-camera pair using the surrounding environment. The methods developed can be implemented using real-world scenes using hardware that is predominantly preexisting in current vehicles. The methods described in this paper do not require any additional information about the surrounding environment and surfaces, are robust under different illuminations and fast enough to be implemented on-the-fly.

The methods described utilise the intensities of the lidar and camera data without the need for feature extraction while still resulting in accurate estimations of the spatial calibration parameters

---

Part I

Introduction

1 The Problem

In the past hundred years, modern vehicles have come a long way from the first cars developed at the end of the 19th century. Many of the vehicles seen on the road today have an array of sensors which help the vehicle make sense of their environment. Some companies are taking this new perception of the world to the next level by producing vehicles that can drive themselves. These algorithms are still dependent on the quality of information that they receive from their sensors. Sensors may lose their initial factory alignment when vehicles are used in a real-world situation. Sensor drift is caused by vibrations, impacts and wear-and-tear from everyday movements and use. This drift from their factory locations can lead to dangerous misunderstandings of a vehicle's environment resulting in costly miscalculations unless a system can be devised to recalibrate and re-align sensors on the fly.

Modern vehicles need to be able to understand both the positioning of items in the world around them as well as the information visually portrayed by these objects (such as the text on road signs). To solve this requirement, the combination of different types of sensors is used. Lidar is an exceptionally accurate way of mapping the world around a vehicle but it gives poor information about the visual cues in the world. Cameras on the other hand are great at determining visual cues from objects but, because an image is 2D, the distance to objects is hard to determine. By combining these sensors, vehicles can understand the world around them to a much greater degree.

The calibration of these two types of sensors provides its own set of challenges. This is due to the different ways in which these sensors perceive the world around them. In this paper, a method of using the maximisation of mutual information will be investigated as a metric for the alignment of the sensor data from a lidar-camera pair. Thereafter, a method is developed to provide on-the-fly recalibration of the sensors.

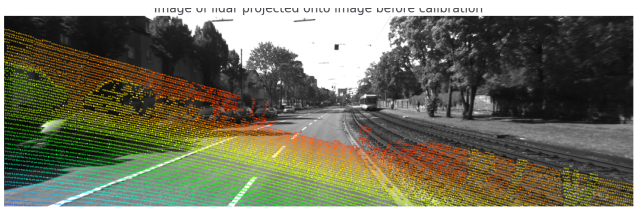


Figure 1: Example of an uncalibrated system with spatial and temporal errors

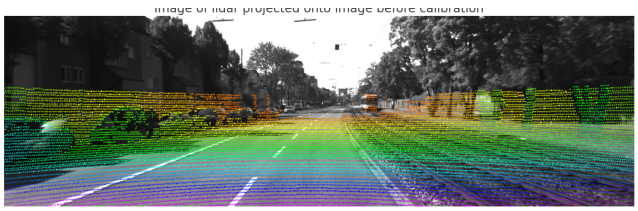


Figure 2: Example of a calibrated system where the lidar is aligned with the image

A comparison between calibrated and uncalibrated lidar-camera systems with lidar points are coloured by depth

The goals of this project are to produce the alignment of sensor data as illustrated from fig. 1 to fig. 2 without using markers or any aids other than the natural environment around the vehicle. The calibration methods developed also need to be generic enough that it can be applied to a range of other vehicles in their natural environments whether they are agricultural robots or autonomous vehicles in a warehouse.

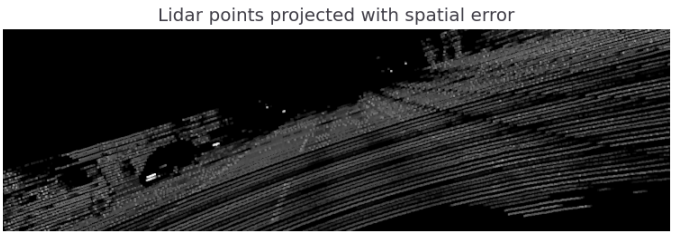
2 Types of Calibration Errors

Errors in the calibration between two sensors broadly fit into two categories, Spatial and Temporal errors. Just as these errors have different causes, the approaches to solving them is different. The next two subsections give an introduction to these categories of errors.

2.1 Spatial Errors

The simplest explanation of a spatial error is when the information from two sensors if out of alignment. This could be caused by the sensors facing in a different direction to where they are meant to face. As a result, when a system has not been spatially calibrated properly, the readings for each sensor cannot accurately be transposed and matched to another sensor's data. A spatial calibration error is generally caused because the sensors have been moved from the location in which they were calibrated. This movement can be caused by vibrations or other mechanical stresses and impacts but the result is a drift away from the initial location of the sensor. This physical movement is what results in the misalignment of the sensor readings and therefore, the vehicle's world view being inaccurate or distorted.

To give an intuitive understanding of what a spatial error actually looks like, a lidar point cloud was projected into the image frame of a camera mounted on the same car. The spatial calibration parameters for the 6-DOF transformation between these two sensors was given some noise to create a spatial error. The results from the projection are shown in fig. 3.



Lidar points projected into image frame with a spatial error

Camera image taken at the same time as the lidar scan

Figure 3: Example of a *spatial* calibration error between a lidar-camera sensor pair

If there were no spatial error, the image made from the projected lidar points would appear the same as the image from the camera. Instead, there are noticeable pitch and roll rotations. There are also translational errors present.

2.2 Temporal Errors

Temporal errors are an error in the timing of a sensor's readings. Temporal errors can be caused by many factors that affect both when a sensor's reading is taken and when the reading is registered by the system. To give an intuitive understanding of what a temporal error looks like, an example of a temporal error between a stereo pair of cameras is shown in fig.4. Note that although the cameras are spatially calibrated, the image from camera 3 on the right is ahead in time (further along the route) when compared to camera 2 on the left.

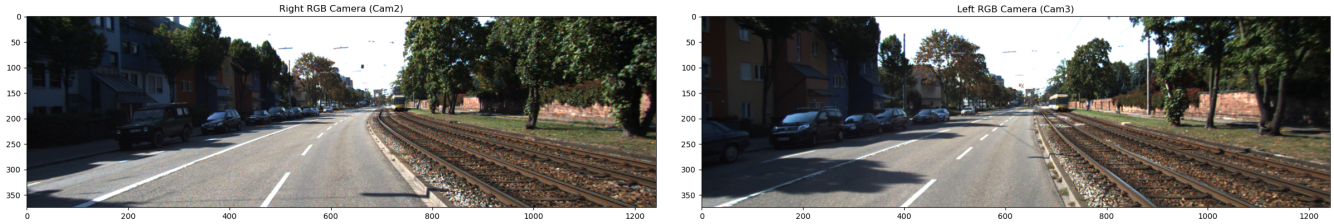


Image for frame #3 from camera 2 in the KITTI dataset

Image for frame #3 from camera 3 in the KITTI dataset

Figure 4: Example of a *temporal* calibration error between two cameras

It is common for cameras and lidar sensors to have independent onboard clocks which may not be perfectly synchronized with the clock of the central processor. There may be filters on the readings, different lengths of communication channels and other factors too all which influence when data is received from a sensor. As a result, a temporal error can cause a comparison of two sets of information that are not from the same instance in time. Therefore, a camera's image could be compared with lidar data which is in the camera's future, resulting in the two not being comparable and leading to a garbage-in-garbage-out effect on the control algorithms.

Temporal calibration ensures that the data used by the vehicle to make decisions all reflects the same point in time where the vehicle was at a specific location. There have been attempts to fix the temporal calibration problem through hardware solutions such as clock synchronization and time stamping sensor readings but these solutions are more expensive. Software solutions are easier to implement, more flexible and can be updated. These software solutions often require the sensors to be spatially calibrated first and when they do not, the solutions become increasingly complex.

2.3 Scope of This Paper

For this paper, the focus will be on recalibrating spatial errors. In order to avoid temporal errors, the dataset chosen had its hardware connected such that any temporal errors would be small enough (less than the time between frames) to not make a significant impact on the spatial alignment of data from the lidar and cameras used. For more information, see the paper by Geiger et al. [1] on how the KITTI dataset was created.

Part II

Related Literature and Work

Developing a technique for lidar-camera spatial calibration is a problem that many researchers have contributed towards in the past 20 plus years. Trying to find a method that is lightweight, efficient and can work in many different environments further adds to the challenge. Lidar and camera sensors are fundamentally very different sensors. Le and Ng explained that "This is because the camera gives 3D rays whereas a laser range finder gives 3D points."[2]. The approach to solving the calibration problem is naturally determined by the way that these sensors work. As such, there are limitations and constraints resulting from the nature fo the sensors. The first is that there is a common field of view constraint. Both the lidar and the camera need to be able to see the same objects. The second is that a pair of sensors need to be both the spatial and temporal calibrated. The requirement for temporal and spatial calibration makes the problem complex when they need to be solved simultaneously. However, there are techniques such as [3] that may be implemented in order to separate the temporal and spatial calibration into two parts. This allows the spatial and temporal calibrations to be performed individually. The focus of the literature review will be on the literature and work centred towards solving the spatial calibration challenge.

The types of spatial calibration can be split into two main methods, target-based and targetless. The use of calibration targets was first implemented in 2004 and is mostly used in older methods. More recently, targetless calibration has become increasingly prevalent thanks to advancements in processing power and the resolution of lidar sensors.

3 Target Based Calibration

In one of the first attempts to solve the spatial calibration challenge, the researchers Zhang and Pless [4] used a checkerboard to find a direct algebraic solution to the calibration of a camera and a 2D lidar. This was solved using the constraints between the views that the lidar and camera had of the calibration target. While the methods described in this paper cannot be used for full 6-DOF systems, they laid the foundation for more complex calibration techniques.

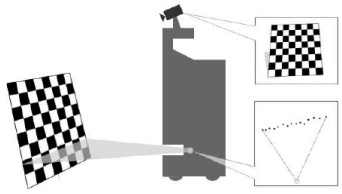


Figure 5: The vehicle calibrated by Zhang and Pless [4] consisting of a 2D lidar, a camera and a calibration target

To simplify the problem of simultaneously solving temporal and spatial calibration, the solution proposed by Fu et al. [3] was to remain stationary while capturing a 360^{o} image and 3D lidar scan of the surrounding environment. The environment contained 2D, planar checkerboards that were identified in the 3D image as well as the lidar map. Using 2D checkerboards and a 3D lidar scan, the researchers were able to calibrate 6-DOF systems. Additionally, there is no temporal calibration that needs to occur when a scene is stationary. This meant that the researchers could simply solve the spatial calibration on its own. A limitation to this research is that vehicles sometimes need to be recalibrated while in use. The method proposed does however solve the dual calibration problem. Their approach of remaining stationary can be used with other techniques where a temporal and spatial calibration both need to take place.

One of the issues with calibration targets is that they need to have specific appearances and sizes.The most common types have checkerboard patterns or irregular holes on a 2D plane. Calibration targets are not common items and so recalibration for target-based methods has the added overhead time of aquiring targets. In an attempt to make marker-based lidar - camera calibration easier to implement, Z. Pusztai and L. Hajder. Pusztai and Hajder [5] developed a way to use ordinary cardboard boxes as the calibration targets. This meant that calibration of the system spatially became a simpler, less specialised task. Boxes of known size were placed in a room in arbitrary configurations. Then the lidar point maps were used to find planes in the environment and from there, isolate the potential cardboard boxes. They also developed a way of predicting where the corners of the boxes were if they were between two points. This became particularly useful when the targets were further from the vehicle and the lidar point map was starting to become spread out with distance. All that was left was to find the boxes in the camera images and map the images from the lidar onto the camera images. They used the dimensions of the boxes to then finish calibrating the system spatially. Similarly to the method proposed in Fu et al. [3], the problem with this technique is that it requires supervision and cannot be done on the fly in an unknown environment. But it provides some good ideas for showing there is no need for complex

planar targets by achieving the same accuracy with simple boxes.

4 Targetless Calibration

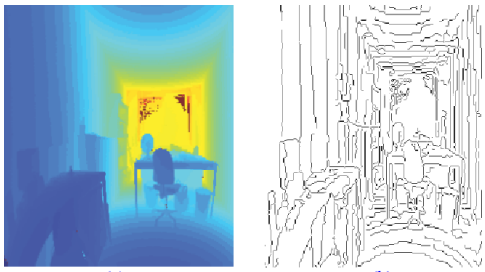


Figure 6: An example of the depth map and extraction of features from [6]

Scaramuzza et al. [6] developed one of the first methods of calibrating a 3D lidar to a camera without the use of target objects. By creating 2D depth images from the 3D lidar and then extracting the edges, they were able to enhance features that a user could then identify in an image. These common points were then used to determine the transformation between the two sensors. This new school of thought paved the way for many of the research that followed. As a result, there have been many methods developed that can calibrate a lidar-camera pair using the information in the surrounding environment.

In an effort to make calibration into a task that did not require human intervention, Levinson and Thrun [7] produced a method that uses the edges in an image and sharp changes in the distance of the lidar point-cloud to quickly determine if there has been a small change to the sensors such that the calibration is now inaccurate. They managed to obtain accuracies of 0.25 degrees for phi, θ and ψ while in x, y and z they could detect changes of less than 10cm, claiming the most accurate offline method at the time of publication. Their method utilised a kernel that used the number of lidar edges that lay ontop of image edges in order to gauge if the calibration was accurate. This method was developed as a lightweight calibration check, however, this comparison of edges could be also be used to find the spatial calibration of a system. It may be too calculation heavy for use in this application but their stress of the importance of calibration checks for real-world robots is noteworthy.

A different approach to determining the spatial calibration parameters has centred on using motion to determine information about the environment and the location of sensors on a vehicle. Historically, the use of motion has been used extensively in the calibration of sensors on robot arms with well-established algorithms such as the hand-eye calibration being used for many years. In hand-eye calibration, the spatial transformation from the robot "hand" to the camera ("eye") is not known. When the robot undergoes a series of known transformations which can be used to determine the transformation between the eye and hand. This process is often referred to as "structure from motion" as developed by Andreff et al. [8].

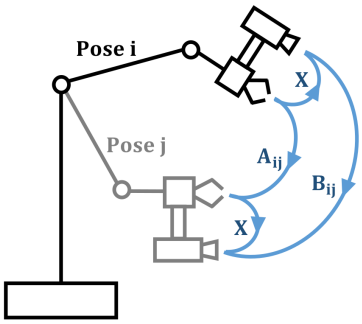


Figure 7: A diagram of the poses used for hand-eye calibration [9]

Another developed by Nagy and Benedek [11] used the motion of a car to synthesize a 3D image of the surrounding environment from a series of consecutive images. This 3D, structure from motion image was then used to synthesize a dense point cloud which could be used to map to the lidar's point cloud. To find the transformation between the two-point clouds, the researchers then found potential objects in the image and mapped these to the lidar point cloud. A finer alignment was then done using a Non-Uniform Rational B-Spline which allowed for flexibility in the shape of the point clouds. This allowed them to account for ellipsoid-shape and other types of distortion due to the motion of the vehicle.
Using movement to calibrate the camera-lidar system removes the dependency on high fidelity GPS and inertial measurement units. Another benefit to this technique of generating synthetic point clouds and mapping those to the lidar point clouds is that there was no need for hardware time synchronization. A downside to this method is that there needs to be a fairly powerful GPU on board if this were to be done on the vehicle. The researchers used an Nvidia GTX1080Ti GPU with 11 GB RAM and it took close to 3 minutes to complete the whole calibration from start to finish. This is a decent time but considering the hardware used, it is unlikely that embedded systems could perform the same steps in similar times. This makes it impractical to implement this method with currently available technology.

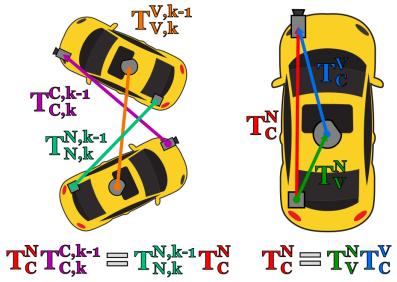


Figure 8: Diagram showing the how the movement of a car can be used to determine structurefrom motion [9]

By adapting these methods, Taylor and Nieto [9] showed that a similar algorithm could be used to determine the spatial offset of a camera and lidar from a vehicles navigation sensor (IMU and GPS). This meant that a coarse estimation of the calibration parameters can be determined by using a similar approach to the hand-eye problem.
The coarse estimation allowed for the calibration to be refined using methods that are not globally convex. The methods used in this paper for calibration refinement were Gradient Orientation Measure by Taylor et al. [10] and those set out in Levinson and Thrun [7]
While the researchers developed a good way to calibrate vehicles without the need for calibration targets, the assumption in this paper was that the vehicle in question would have reliable, highly accurate GPS and IMU data as well as a good amount of computing power onboard. The vehicle also needed to be able to perform manoeuvres in order to obtain the information necessary for structure from motion. The method is however a very easy way to use the same technique for different vehicles which the authors demonstrated.

5 Non-parametric Methods

In 1995, Viola and Wells [12] introduced a novel way of aligning data between 2D camera images and a 3D model, even when the model was in a slightly different orientation. They used a mutual information metric to align the 3D model produced using magnetic resonance imaging with 2D image stills from a video, rotating the 3D model to match the orientation in the 2D stills. This method proved robust even in varying scene illumination. This is particularly useful in the calibration of LiDAR and cameras in the autonomous vehicle environment.

Pandey et al. [13] built on the work of [12] and showed that Mutual Information can be a viable method for spatial calibration of LiDAR - camera sensor pairs. They showed that by using the reflectivity of the lidar point cloud, it can be aligned with a grayscale image.
Their methods showed that maximisation of mutual information can be used to align lidar and camera data without using feature detection or segmentation.

Scott et al. [14] further added to the work of [13] and [12] to show that with proper scene selection, maximisation of mutual information can provide accurate alignment of lidar and camera data. They showed that Normalised Information Distance can be used as a metric for scene selection so that the scenes chosen are optimal for use with a non-parametric alignment method.

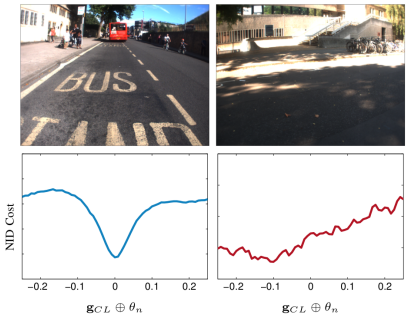


Figure 9: Comparison of a good (left) and bad (right) scenes including their associated cost functions when using normalised information distance [14]

A problem with many of the non-parametric methods is that they require good resolution from the LiDAR sensors that are used. 64-beam lidar can cost upwards of 75,000 USD which makes them unaffordable. This provides a barrier to the use of methods like those described in [12][13][14]. For these methods to be implemented in the real world, they need to be implementable using cheaper lidar sensors that have sparse point clouds. You et al. [15] used stereo cameras to generate a pseudo lidar image in an attempt to remove the dependence of many systems on expensive, high-performance lidar sensors. The researchers used cheaper, sparse measurement lidar sensors in order to de-bias a pseudo lidar. This method removes one of the main weak performances of pseudo lidar, allowing them to accurately estimate the depth of faraway objects. A corollary of their research is that their methods of creating pseudo-lidar using only stereo cameras could be used as an alternative calibration method in 3D space instead of in the 2D image plane of a camera. The main benefit to the goals of this paper is that, if lidar data is sparse, there is still ways to calibrate lidar and monocular cameras provided that there is a stereo camera pair available to create a pseudo-lidar point cloud. This helps avoid one of the pitfalls of non-parametric methods

6 Reflection on Related Work

The literature has been influential on the direction of this paper by providing a clearer description of the problem and the approaches that have been undertaken in attempts to solve it. From the literature that has been reviewed, there are two main methods for calibrating a lidar-camera sensor pair. The first method used markers that are of known shape, texture and placed into what is often a known, simple environment. This is a more hands-on, involved approach used in the early attempts to solve the lidar-camera challenge The other type of method was markerless calibration. In these cases, the goal is to use the information in the natural environment around the vehicle in order to calibrate the sensors. Some of the methods extracted features from the environment that could be found in both the lidar and camera sensor data before using these to align the data. These methods became challenging to implement when objects in the environment have textures which make it harder to determine the edges of the shape.

The Structure from Motion approach seemed like a good way to deal with the texturing problem while also separating the spatial and temporal aspects of the calibration into two separate challenges. However, it can be more computationally expensive than some of the other techniques and also requires a high fidelity IMU or navigational sensor which may not be available. Structure from motion and other motion-based techniques also often involve identifying and removing targets that are moving relative to the global frame(dynamic targets) and trying to isolate those which are static. These are then used to determine the calibration parameters. This process can sometimes be challenging to implement especially where computing capabilities are limited. This limits their application in "offline" scenarios.

Non-parametric methods seem to be the lightest type of markerless calibration but they are limited by the scenes surrounding the vehicle. They also require the lidar sensor to have a good amount of resolution. If this is not a problem, the use of maximisation of mutual information seems to be a very effective method provided that multiple frames can be used.

The literature has influenced the trajectory of this paper by showing that markerless calibration is the best option. The use of segmentation and object detection is not something that can be implemented on limited hardware, the same is true for motion-based methods. According to the literature reviewed, the best methods for calibration of lidar-camera pairs in autonomous road vehicles are non-parametric methods that use some form of mutual information metric. These methods are lightweight in comparison and can be implemented on the fly without the need for large amounts of computational power.

In this paper, the focus was on implementing a mutual information-based method for the estimation of the spatial calibration parameters between a lidar-camera sensor pair.

Part III

Methodology

7 Choice of dataset

This project required a dataset that would give easy access to both LiDAR and camera images. The dataset preferably needed to be one with a good amount of support online as well as being something that well known in this field of research.

There were two main datasets that stood out; the first was the oxford robot car dataset [16]

while the other was the KITTI dataset [1]. Both datasets are well known, provided good support to users and had a good community around them.
For this project the KITTI dataset was chosen. The main reason was that the researchers had ensured that the data did not have any significant temporal errors. For more information on how this was achieved, refer to [1]. The KITTI dataset is also simpler since it only had Velodyne LiDAR, IMU and camera sensors onboard. This made it a more suitable dataset as this was all the data that was needed for this project as only the lidar and camera sensors are needed.

7.1 About the dataset

The KITTI dataset was made by the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago. Their goal was to make a standard, accessible dataset somewhat like the MNIST dataset in machine learning.

The sensors onboard this dataset are:

- 1 Inertial Navigation System (GPS/IMU): OXTS RT 3003
- 1 Velodyne LiDAR
- 2 Grayscale cameras
- 2 Color cameras
- 4 Varifocal lenses
- as well as GPS and an IMU

For the purposes of this project, the sensors used will be the LiDAR and grayscale cameras. The location of these sensors is shown in Fig. 10

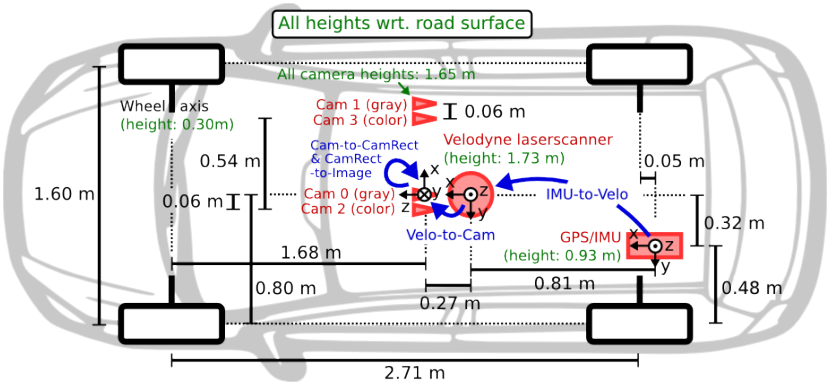


Figure 10: Top view and locations of sensors in the KITTI dataset
Figure courtesy of Geiger et al. [1]

One of the main reasons for choosing this dataset was the placement of a switch on the lidar which triggered the camera's every time the lidar was facing exactly forward. This, combined with the very fast shutter time compared to the LiDAR's scanning rate meant that the temporal errors were very small.
This made the dataset an easier choice for finding the spacial calibration parameters as there was no longer a need to find images and scans that correlate to the same point in time thanks to this trigger mechanism.

7.2 Raw Data from the dataset

7.2.1 Grayscale Cameras

The grayscale cameras provided high-resolution images of the environment. These images could be accessed in a corrected and raw format. Once corrected, the images were corrected for lens distortion and other artefacts such that they resembled images from an ideal pinhole camera.

These pinhole images were used as correcting for image artefacts is not a necessary part of the calibration process as these parameters are already supplied and there is a large amount of literature available on the process.

An example of the images from the grayscale cameras as well as the colour cameras is shown in Fig. 11

Figure 11: Examples of images from cameras 0 and 1 respectively

7.2.2 Velodyne LiDAR

The LiDAR used in the creation of this dataset is the Velodyne HDL-64E which produces scans at 10 frames per second. It is a high-resolution LiDAR with 64 channels and approximately 100 thousand points captured per frame according to its datasheet. These features made it a good candidate for use when refining a calibration technique as the high data output make the high density scans easier to compare to the pixels in the camera's images.

The point clouds output by the Velodyne have four dimensions, $x, y, z,$ and $Intensity$. To give a good idea of what this data looks like without any pre-processing, a point cloud was plotted and the intensity was used for colouring the points. This is shown in Fig. 12 for the same forward view as shown in the images in Fig. 11.

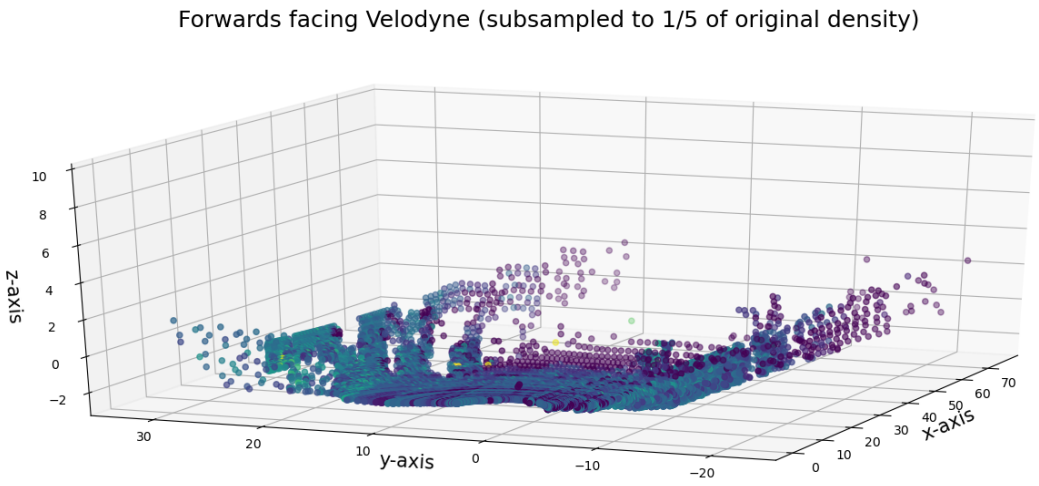


Figure 12: Point cloud from the raw velodyne data

Although it is hard to see, the yellow dots in the left of the image in Fig. 12 are the numberplates of the two cars parked in the shadow on the left of the images in Fig. 11. The stripes of the train's rails are faintly visible almost parallel to $y = -5$.
This data needs to undergo a transformation such that it is projected onto the image from camera 0.

8 Preprocessing of Data

9 3D Transformations and Rotations

9.1 Coordinate systems

The first major challenge of aligning point clouds with images is that the data is inherently in different dimensions. LiDAR, by nature, determines the location of a point in 3D space relative to the Velodyne sensor. This is very different to a camera which projects points through an aperture onto a plane.

A LiDAR point is normally given in the form of $p_{velodyne} = [x, y, z, I]^T$
Where I is the intensity/reflectance value and x, y, z are the positions of the point relative to each axis in the LiDAR's reference frame. In this project, the homogenous coordinates were used [17]. This meant that each point had two values associated with it, it's reflectivity and it's coordinate. These were in the form

$$p_{velo} = [x, y, z, 1]^T \text{ and } I(p) \tag{1}$$

For congruency with the literature reviewed, a point P_i in the LiDAR frame will be referred to as $\boldsymbol{X_i}$ while the same point P_i captured in the image frame will be $\boldsymbol{Y_i}$

The coordinates used for images in this paper will also follow the standards set out by Barfoot [18].

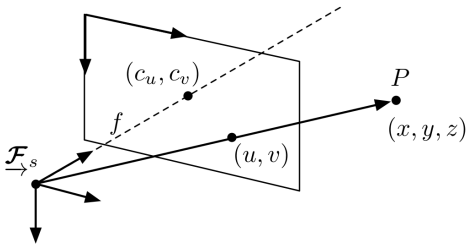


Figure 13: Simplified diagram of ideal camera model and coordinates from Barfoot [18]

In the image plane, pixels are represented by their homogenous coordinate as well as their intensity. Therefore a pixel can be fully described by:

$$p_{cam0} = [u, v, 1]^T \tag{2}$$

Where u and v are the distances from the center (c_u, c_v) and the camera's pinhole is at $\underrightarrow{\mathcal{F}}_s$

A diagram showing this model is shown adjacent in figure 13

When comparing reference frames, the vehicle frame was not used as the scope of this paper is on the calibration of lidar-camera sensors. As a result, the calibration with the IMU and GPS was outside of the scope of the project. For a consistent reference coordinate frame, camera 0 was used throughout this paper.

This choice meant that all other sensors reference frames are with respect to camera 0. If the lidar points are needed in a different frame it is just a case of projecting them to another camera frame from where they were projected to into camera 0's frame.

9.2 Projections and Rotations

To compare the data points from each of the sensors, the first requirement is that they be in the same reference frame. This means the LiDAR points in 3D space needed to be projected back into the image plane where the corresponding pixels could be sampled. After this, the data from the two sensors overlapped and could be compared to see if the calibration parameters were correct.

Figure 14 below shows the transformation in terms of $[R, t]$ that needed to be determined to project the lidar's data (X_i) for the real world point P_i back onto the camera's data (Y_i for the same point in 3D space.

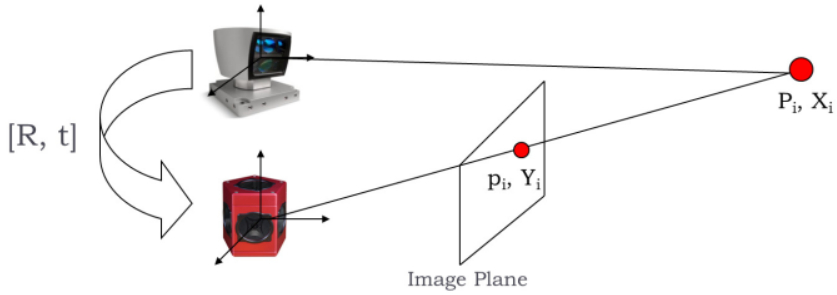


Figure 14: Diagram of the projection of point P from X in the lidar frame to Y in the camera frame
Pandey et al. [13]

Before that could be done, the rotation and translation matrices need to be defined.

9.2.1 Image Plane Transfromations

Each of the cameras in the dataset have slightly different locations and orientations. Therefore, the images from each camera need to first be projected onto a common image plane. Once this is done, it becomes simpler to compare common points in images. This uses the process of rectification. In the case of this paper, the LiDAR points can be projected onto any camera image if they are first projected onto the reference frame and the transformations from the reference frame to the camera's frame are known.

The transformation from reference to rectification planes is supplied with the dataset. Although the goal for this paper was to find the spatial calibration parameters for camera 0, the techniques described can easily be adapted to find the calibration parameters using another camera if there is a problem with the first camera.

Figure 15 shows a model of the rectification of two images of the same point P in 3D space photographed by the cameras.

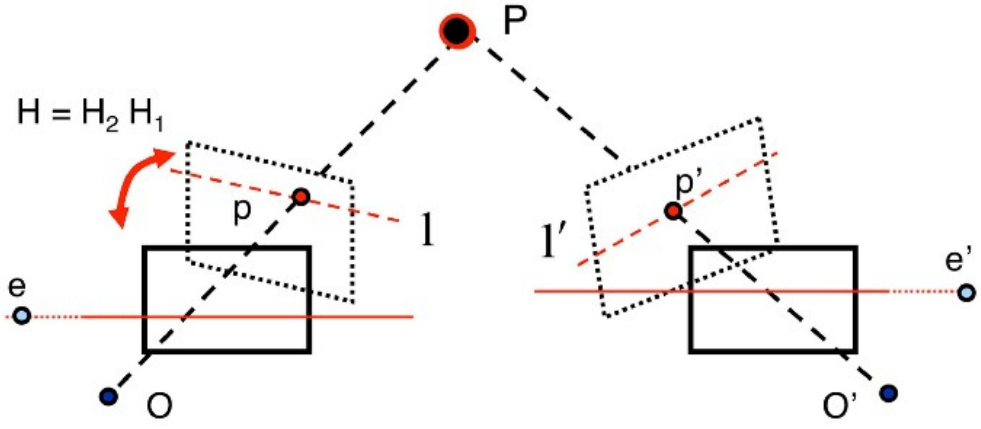


Figure 15: Figure of the reference to rectification plane transformation [19]

Rectification is the process of projecting images taken from one or more cameras onto a common plane so that the features in the images may be compared more easily. Features in images from the different lidar/camera sensors can also be compared in directly in the rectification plane.// In this paper, the rectification plane is used to project lidar points into cam0's plane and the images from cam0.

From the rectification plane, the matrix $P_{\text{rect-plane to cam0}}$ was then used to transform the lidar points to camera 0's plane. This transformation is determined by factors such as the width, height and distance of the focal plane as well as the location of the optical axis and any lens distortions. These parameters produce the *intrinsic parameter matrix* K where f_u and f_v are the focal lengths in horizontal and vertical pixels respectively. c_u and c_v are the horizontal and vertical displacements of the optical axis from the center of the image. This is shown in Eqn. 3.

This gives $P_{\text{rect-plane to cam0}}$ the form:

$$P_{\text{rect-plane to cam0}} = \begin{pmatrix} f_u^{(i)} & 0 & c_u^{(i)} & -f_u^{(i)} b_x^{(i)} \\ 0 & f_v^{(i)} & c_v^{(i)} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3}$$

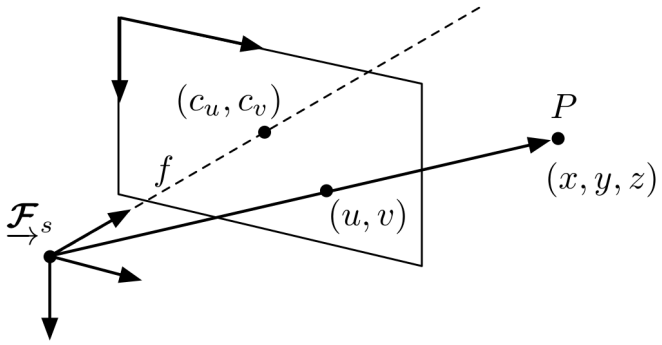


Figure 16: Figure of camera focal plane

9.2.2 Frame Rotations and Transformations

In order to go from the coordinates in the LiDAR frame to those in the image frame, two things need to happen. The first is that there needs to be a translation from the location of the Velodyne frame's centre to the centre of the reference camera's frame. This is then followed by a rotation into the right coordinates.
Therefore there are six degrees of freedom defining this system which will be refered to as Θ where:

$$\Theta = [x, y, z, \phi, \theta, \psi]^T \tag{4}$$

The transformation from velodyne to camera reference plane coordinates can then be given by:

$$T_{\text{velo to cam0}}(\Theta_{calibration}) = \begin{bmatrix} R_{\text{velo to cam0}} & t_{\text{velo to cam0}} \\ 0 & 1 \end{bmatrix} \tag{5}$$

Where $R_{\text{velo to cam0}}$ is a Tait-Bryan Z-Y-X rotations matrix about $[\phi, \theta, \psi]^T$ and t is the transformation in $[x, y, z]^T$ [20].

$$R_{\text{velo to cam0}} = R_\phi R_\theta R_\psi = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\cos\psi & -\sin\theta \\ -\cos\psi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{6}$$

And $t_{\text{velo to cam0}}$ is the translational components.

$$t_{\text{velo to cam0}} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{7}$$

9.3 Projecting Lidar onto an image

Now that all the transformation matrices have been defined, the next step is to project the data from the lidar sensor onto an image from the reference camera, cam0.

The transformations need to follow the correct order to go from the Velodyne reference frame to that of Camera 0. This order is shown in Fig. 17. Where the black arrows are the path taken when projecting Velodyne data onto an image

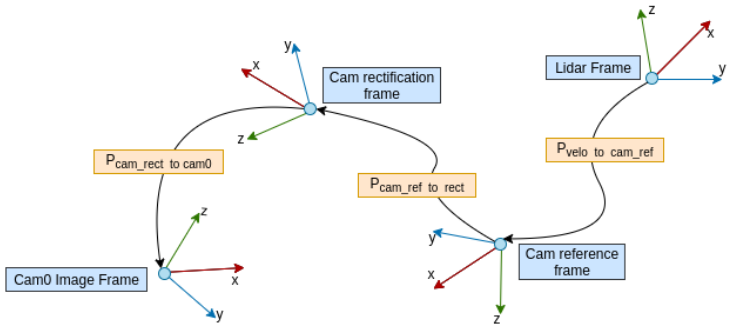
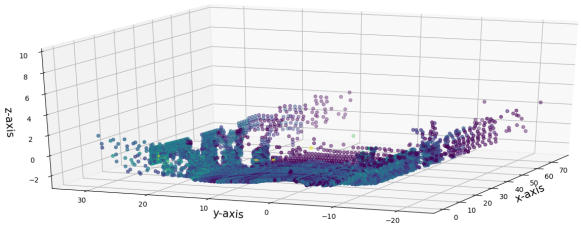


Figure 17: Transformation steps from velodyne to Camera 0

,

Therefore, the transformation from Velodyne to Camera 0 has the following mathematical equation:

$$\begin{bmatrix} u \\ f \\ 1 \end{bmatrix}_{\text{Cam0 coordinates}} = P_{\text{rect-plane to cam0}} \cdot R^{-1}_{\text{cam-ref-plane to rect-plane}} \cdot T_{\text{velodyne to cam-ref-plane}}(\Theta) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{\text{Inertial coordinates}} \tag{8}$$



A slice of the velodyne data facing up the street

An image from camera 0 in at the same time

Figure 18

The velodyne data in Fig. 18 is was then projected onto the camera image displayed next to it using the equation described in equation. 8. The points were then coloured according to their distance from the sensor and the result was the image shown below in Fig. 19.

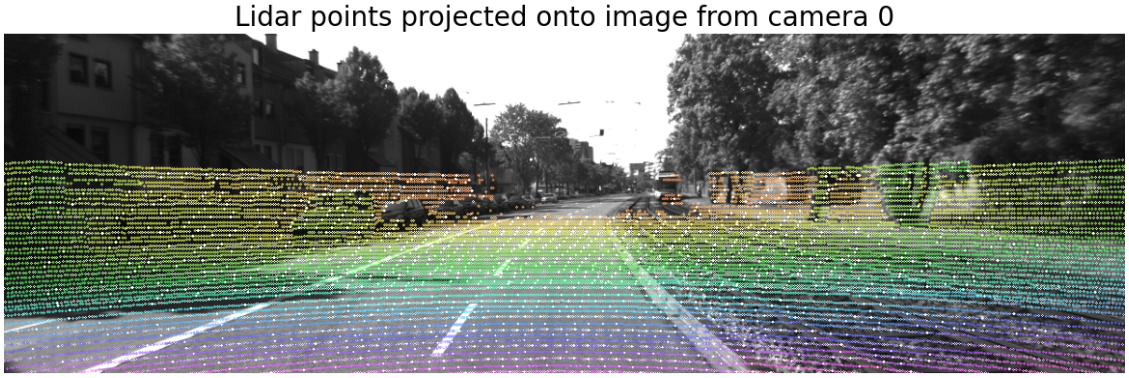


Figure 19: Results after projecting lidar onto camera 0

The image above in Fig. 19 shows how the projected lidar points match the scene captured by camera 0. Since $P_{\text{rect-plane to cam0}}$ and $R_{\text{cam-ref-plane to r}}$ are both determined by the characteristics of the cameras used, all that needs to be optimised for the calibration of the system is the calibration parameters, Θ, that determine $T_{\text{velodyne to cam-ref-plane}}$.
Theta is determined by the 6-DOF in this system and is defined as:

$$\Theta = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$$

To determine these parameters, the projected lidar points will be aligned with the image from camera 0 using the techniques used in the next sections.

10 Correlation Metrics

10.1 Mutual Information

Mutual information is a measure of the statistical dependence of one seemingly random variable on another. This can be interpreted as a metric for how much information one random variable communicates about another random variable. [21]

So if the two variables are statistically independent, their mutual information should be 0 as they provide no information about the other whereas if they are identical, they should have a normalised mutual information score of 1.

Viola and Wells [12] observed that the grayscale values of an image of an object taken from different sensors are similar enough that they could be used to align the images. They also demonstrated that this metric works when aligning 3D models with an image.This done by using mutual information as a metric which, when the sensor data is aligned, gives the highest score. Therefore, all that was required to find the best calibration parameters was to maximize the mutual information score

Mutual information also provides a metric to determine how close the best-found calibration is to the ground truth.

To calculate the MI score for two sets of points, in this case an image, the following equation is used:

$$MI(X,Y) = \frac{Joint\ Entrophy\ of\ (X,Y)}{P_{marginal}\,(X)\ \ P_{marginal}\,(Y)} \tag{9}$$

where the joint entropy is given by:

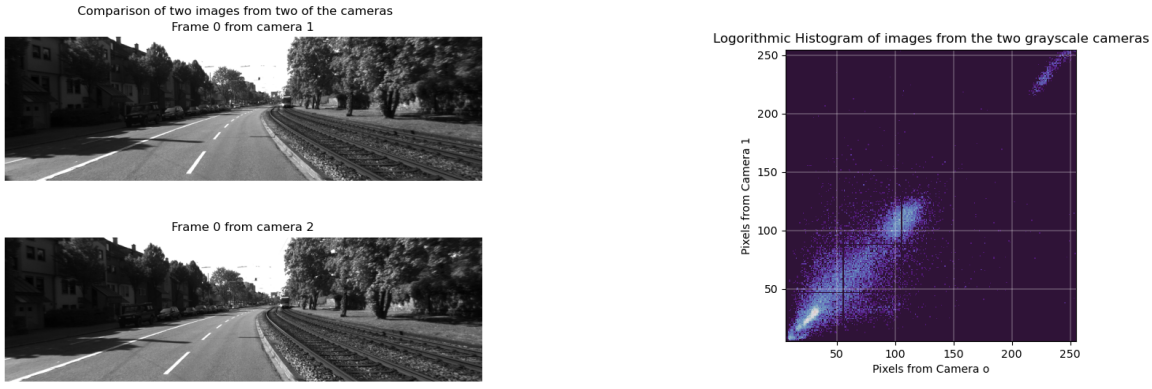
$$H(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y)\, log\, P(x,y) \tag{10}$$

For this project, to find the joint probability $P(X,Y)$ a 2D histogram was used. All of the values were scaled by 1/sum(Histogram) such that it represented the probability

Combining these, the mutual information is given by the following equation Learned-Miller [21]:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y)\, log \frac{P(x,y)}{P(x)P(y)} \tag{11}$$

To demonstrate how mutual information can be used for the data from two different sensors, the images from cameras 0 and 1 on the car (see Fig. 10) are compared before and after misalignment. In the histograms displayed the brighter an area along the diagonal is, the more the pixels in the same locations in the two images have the same value.



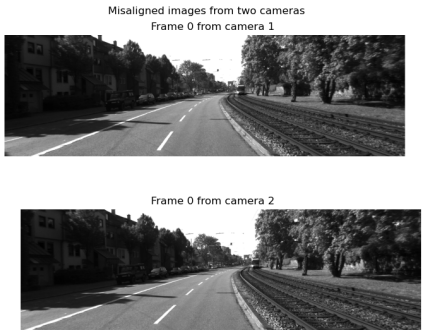
Images from the two grayscale cameras

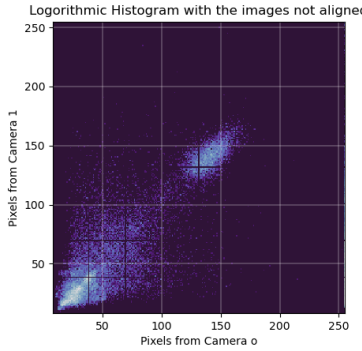
Histogram between two images

Figure 20: Comparison of the images with a good alignment

The images above are aligned using the ground truth calibration from the dataset. The histogram in Fig. 20 shows the entropy between these two images. This was used when calculating the mutual information score.

To give an incorrect alignment, the images were given a shift the equivalent of 50 in the positive x direction. These misaligned images can be seen in Fig. 21 as well as the histogram for this misaligned pair.





Images from the two grayscale cameras with offset

Histogram between two images with an offset

Figure 21: Comparison of the images once a translational misalignment is added

Notice the decrease in the brightness along the diagonal image. This shows that there are fewer pixels that correspond in the two images.
The two images were given further errors in alignments in order to demonstrate how the MI score changes the further the alignment is from the ground truth.

As can be seen from the adjacent Table 1, when the images are aligned properly, the MI score is significantly higher.

The mutual information score continues to decrease as the alignment error gets greater. This is what was needed in a metric of alignment.

Error in px		**Mutual Info score**
Aligned images	(Fig. 20)	4.8443
Misaligned by 10px		1.372
Misaligned by 20px		1.1903
Misaligned by 50px	(Fig. 21)	0.9788
Misaligned by 100px		0.819

Table 1: A comparison of the Mutual Information scores for the correct and incorrect calibration

In order to show how significant the drop off is in the mutual information score, the values in Table 1 were plotted. This gives a more accessible insight to the effectiveness of this score as an alignment metric.

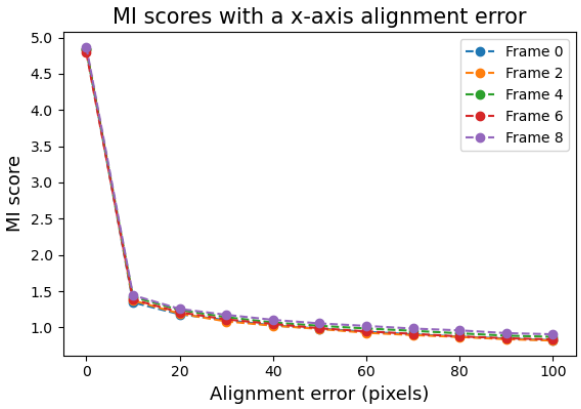


Figure 22: Mutual Info scores for different frames

The scores in Table 1 have been plotted in the adjacent Fig. 22.

The Mutual Information score decreases quite rapidly as the images are moved further out of alignment. This means that by optimising the mutual information score, the correct calibration can be found.

By taking the score over several different frames, the reliability of this metric for different scenes is demostrated.

Mutual information gives a featureless comparison of two sets of data points which do not have to come from the same sensor. This metric was normalised such that a MI score of 1 is a perfect correlation while a score of 0 means there is no correlation between the points.

The formula for the Normalised Information Score (NMI) is:

$$NMI(X, Y, \Theta) = \frac{2 \times MI(X, Y, \Theta)}{H(X) + H(Y))} \tag{12}$$

Where $H(X)$ is the entrophy of the points in X, the lidar data. Similarly, $H(Y)$ is the entrophy of the points in Y, the sampled camera points.

The result of factoring in the entropy of each of X and Y is that the score becomes more robust as well as being normalised. The results of the Normalised MI score are shown in Fig. 24 as a point of comparison with the MI score above in Fig. 22. Note that the scores obtained by NMI are not just normalised as they are different due to the entropy component.

The Normalised Mutual Information score the metric used for the optimisation process in this paper.

Error in px		**NMI score**
Aligned images	(Fig. 20)	0.4952
Misaligned by 10px		0.1402
Misaligned by 20px		0.1216
Misaligned by 50px	(Fig. 21)	0.0998
Misaligned by 100px		0.0832

Figure 23: A comparison of the Normalised Mutual Information scores for translational errors

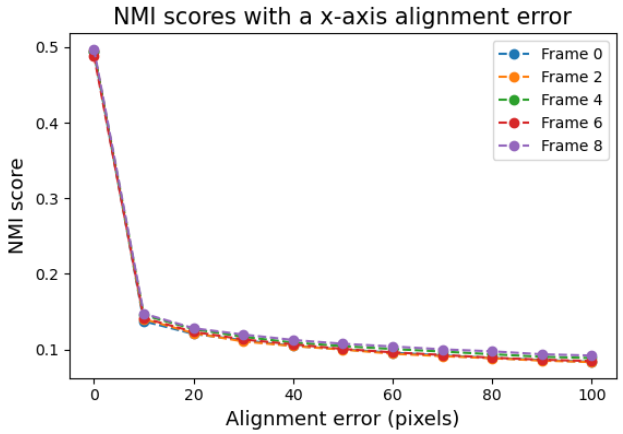


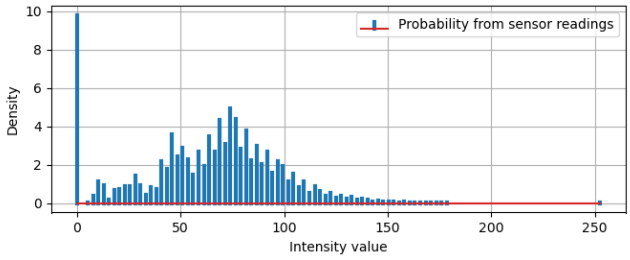
Figure 24: NMI scores over different frames

10.2 Kernel Density Estimates and Histograms

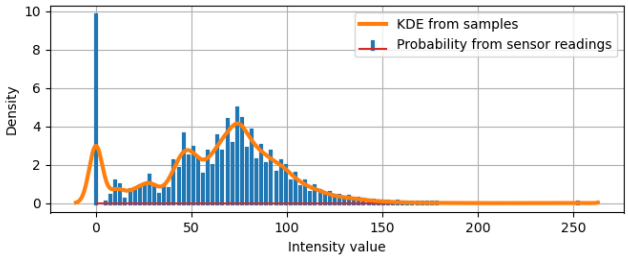
A kernel density estimate (KDE) is a way of estimating parameters that may have been missed during sampling or for the smoothing of a density function such as a histogram. Because the KDE is a non-parametric method, no knowledge or assumptions about the distribution is needed [22]. This makes them particularly useful for smoothing histograms of images as they can be erratic and far from a normal-distribution depending on the scene.

Smoothing the distributions of pixels in an image is important when data is obtained from two different perspectives of the same object. Images of an object obtained from slightly different locations will give slightly different distributions of pixels as the visual features of the object will be change. The effects of this difference in perspective can be smoothed out and missed data points in the distribution of samples can be estimated from the samples that are present. Because a KDE can be used to smooth the distribution curve with little knowledge of the distribution, it is a good step before for comparing image data and is why it is a good metric to use in this case.

To give an intuitive understanding of the difference a kernel density estimate makes, figure 25 below shows a 1D histogram smoothed using a KDE where the bandwidth is determined by Silverman's rule of thumb.



1D Histogram of an image from Camera 0



After the histogram was smoothed using a KDE

Figure 25: Effect of smoothing using a KDE

As can be seen in Fig. 25, the KDE normally extends past the boundaries of the possible values for the data. This is fixed by reflecting the overlapping region onto the allowed values and then summing it with the probabilities that it falls on. This results in a kernel density estimate that is bounded in the same way that the input data range was.

The KDE on the right was bounded so that the probabilities do not fall outside of the input range of $0 \leq x \leq 255$

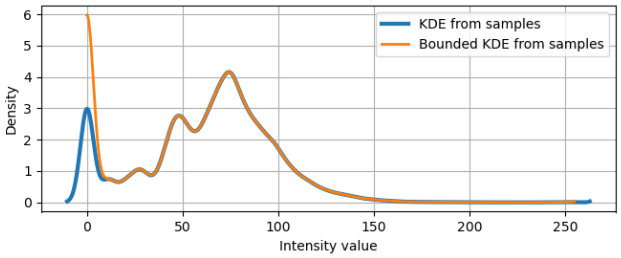


Figure 26: Final, bounded 1D KDE

10.2.1 Implementation method for a Kernel Density Estimation

A kernel density estimate works by convolving a series of points with a kernel. The kernel used in this paper is a gaussian curve that has its bandwidth determined by Silverman's rule. This convolution is effectively replacing each singular point with a gaussian curve and then summing the overlaps of the curves.

By placing a kernel at each location that there is a data point, the effect is that any gaps are smoothed by the side lobes of the gaussian.

This is particularly advantageous in this application as mutual information metrics can be distorted by shadows, changes in orientation of the objects due to the different views of the sensors among other causes. Using a KDE to smooth out an image's histograms can help reduce error from these sources.

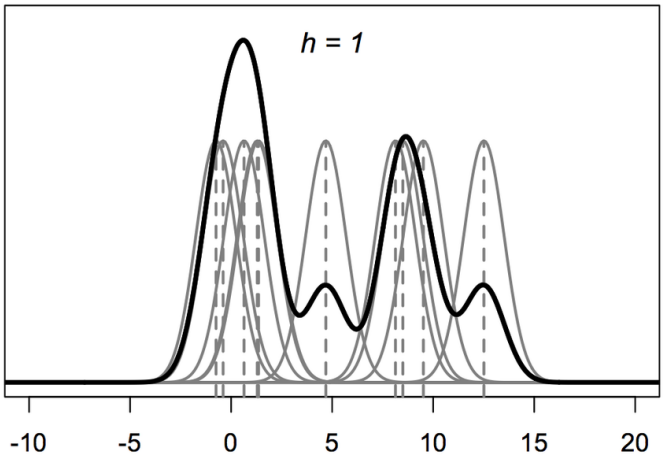


Figure 27: Figure from [23] showing the weighting of kernels to create the KDE shown by the dark line

The bandwidth of the KDE was calculated using Silverman's rule of thumb. For a 2D PDF between two same-sized sets of samples, the formula is:

$$\Omega = 1.06n^{1/5} \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \tag{13}$$

Where σ_x and σ_y are the standard deviations of X and Y.

The KDE kernel was then implemented to find the joint probability using the following equation [13]

$$\hat{p}(X = k_x, Y = k_y) = \frac{1}{n} \sum_{i=1}^{n} K_\Omega \left(\begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \right); \quad (k_x, k_y) \in ([0, 255] \times [0, 255]) \tag{14}$$

Where X is the projected lidar points and Y is the sampled pixels at the same points in the image. K_Ω is a symmetric kernel with a bandwidth determined by Ω which is computed using Silverman's rule of thumb as shown in Eqn. 13.

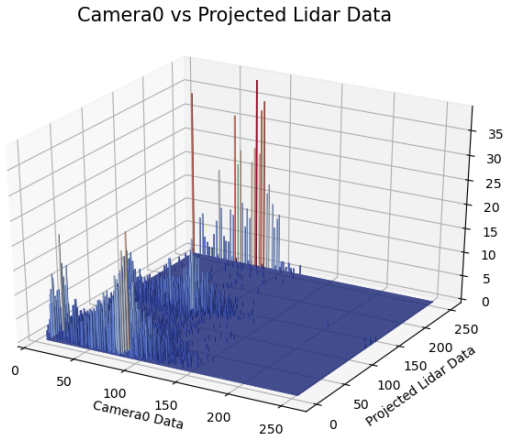


Figure 28: Histogram comparing projected lidar data and the camera data

When the 2D histogram similar to the one shown in Fig. 20 is plotted in 3D, the frequency of pixels with the same intensity occurring in the same spot to determines the height instead of colour. This method was used to create a 3D plot between the lidar points projected onto camera 0 and the sampled pixels from camera 0 for frame 0.

The resulting histogram can be seen in Fig. 28. This plot is a good example of how the height of the spaces between high points can be much lower and almost discontinuous.

This data is then smoothed using a KDE as described in this section. The resulting graph is much smoother with less of the sharp spikes and discontinuities that the histogram in fig .28 had.

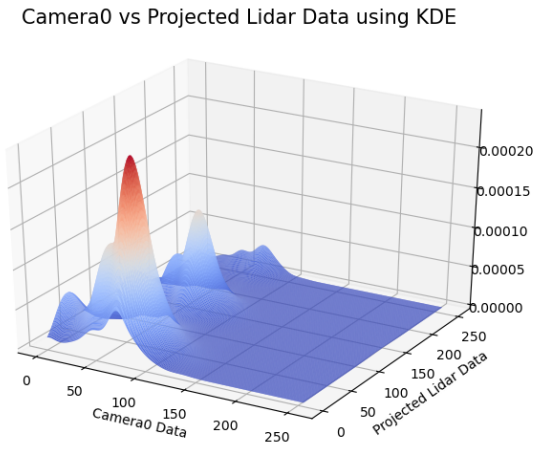


Figure 29: Histogram smoothed with KDE

11 Optimisation

Now that there is a way to get the sensor data into the same reference frame, find where they overlap and get the resulting correlation between the data. The alignment (or lack thereof) between the two sets of data points can be quantified.
All that remains is to maximise the alignment of the LiDAR and image in order to find the best calibration parameters.

11.1 Gradient Ascent Steps

11.1.1 Single Dimensional Gradient Ascent

Since the goal is to find the best alignment of the sensors and the corresponding maximum MI score, the technique chosen for the optimisation was a gradient ascent. Gradient ascent is comparable to trying to climb a mountain in a heavy fog. The process requires taking a step in a direction, seeing if you are higher than you were before and if not, going back the way you came. While there is no way of directly determining the gradient, since mutual information will only give one scalar score, a gradient can be estimated by comparing the score at two different calibration inputs.

The quality of the calibration is going to be assessed by the mutual information score for a given X, Y and Θ.

The gradient vector $\mathcal{G}$ could then be defined as:

$$\mathcal{G} = \nabla MI(X, Y, \Theta) \tag{15}$$

$$\therefore \ \mathcal{G}_n = \frac{MI(X, Y, \Theta_n) - MI(X, Y, \Theta_{n+1})}{\gamma_n} \tag{16}$$

Where the step size γ is defined as:

$$\gamma_n = min\_step + \alpha(1 - min\_step^{1/n}) \tag{17}$$

While α and min_step are hyperparameters that are set according to the expected magnitude of the component of Θ that is being tuned.

To step the calibration parameters from the previously found (Θ_n) to the new, hopefully better calibration parameter of Θ_{n+1} the following formula was used:

$$\Theta_{n+1} = \Theta_n + \gamma_n \frac{\mathcal{G}_n}{||\mathcal{G}_n||} \tag{18}$$

Thereafter, the cycle was repeated until a local maximum was found and the gradient ascent had converged. This should end up producing something akin to the staggering path shown in Fig. 30 adjacent.

Gradient ascent will normally start at a low point (in the case of the figure, at (0,0)) and end up at a local maximum. The method used to try avoid local maxima and instead find the global maxima is described in section 11.3.

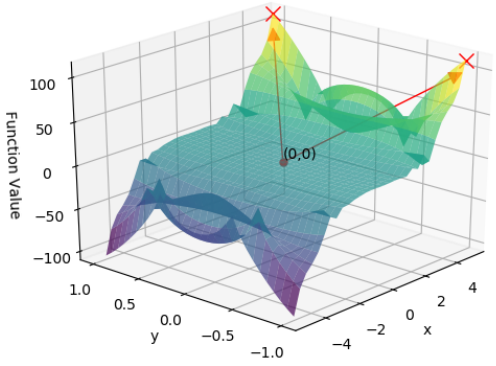


Figure 30: A diagram demonstrating the path taken during gradient ascent in 2D [24]

To test the optimisation of the calibration parameters in one dimension, gradient ascent was performed using the steps described above on the y parameter. The ground truth provided with the dataset was $y_{true} = -0.0751087$. To ensure the test was fair and extreme, an initial guess of $y_0 = -1.0000$

The following plots were generated during the gradient ascent:

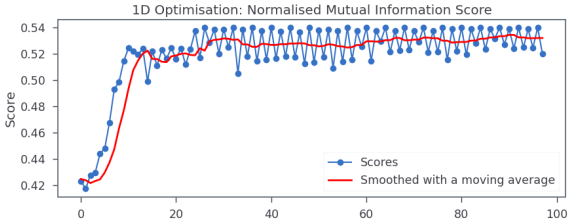


Figure 31: A plot of the MI scores while optimising y

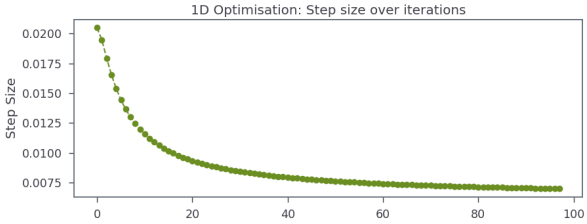


Figure 32: A plot of the step size while optimising y

11.1.2 A note on variable step sizes

Being able to vary the step size for a variable based on the amount of change that it makes in the scoring metric improves the performance of a gradient ascent algorithm. It prevents overshooting a maximum and taking fine steps while also taking large steps when far from the maximum. These types of approach are not as reliable as a preset stepsize that gradually decreases with the number of iterations.

An example of how a variable step size can be unpredictable is the variable step size is discribed in Pandey et al. [13] . The formula given for calculating the step size, γ, by this technique was:

$$\gamma_n = \frac{s_n^T s_n}{s_n^T g_n} \tag{19}$$

Where:

$$s_n = \Theta_n - \Theta_{n-1} \text{ and } g_n = \mathcal{G}_n - \mathcal{G}_{n-1} \tag{20}$$

The problem is that as the gradient approaches 0 when the gradient ascent begins to converge, the step size increases rapidly and this would misalign the images completely. This can cause there to be no points where the lidar points can be projected onto a pixel and therefore the MI score is 0.
At this point, the gradient ascent becomes impossible as most changes to the calibration parameters still result in a MI score of 0 and therefore no gradient.

The way this was solved was to have a step size that was small to begin with but enough to still significantly change the calibration parameter if the initial guess for Θ_k was very incorrect. The step size would then decrease by an exponential factor of $\frac{1}{n}$ where n is the number of steps that have been completed since the initial guess. The formula that was eventually used is shown in Eqn. 17.

As a rule of thumb, the hyperparameters were set according to the maximum expected value that the calibration could reach. For angles, this was easier as they were bounded to the range $[-\pi, \pi]$. For the translational components - x, y and z - the maximum distance that can be moved in most directions from the center to outside of the car is approximately 2m. Therefore the range for x, y and z was assumed to be $[-2, 2]$ meters.

The learning rate α and minimum step size min_step were then set by a percentage of the maximum expected value of Θ_k as given below:

$$\alpha = 10\% \; of \; \Theta_k$$

$$min\_step = 0.1\% \; of \; \Theta_k$$

Where $k \in [x, y, z, \phi, \theta, psi]$

For the formula

$$\gamma_n = min\_step + \alpha(1 - min\_step^{1/n}) \tag{21}$$

Results in the curve that these produces can be seen in the section above, Fig. 32.

11.2 Multi Variate Gradient Ascent

Optimising multiple variables simultaneously was a significant challenge.
Several different techniques were used and it became clear that optimising x, y and z could be done without much modification to the gradient ascent methods described above. The optimisation of the angles was significantly more difficult for reasons that will be explained in it's subsequent section.

11.2.1 Optimising $x, y,$ and z

Optimising the translational components, $x, y,$ and z, of the calibration parameters was done using nesting gradient ascent algorithms. This is described in more depth below but first, a word on why it worked using this method while the same method used for the angles did not.

The axis of $x, y,$ and z are independent on one another. As a result, a shift in the positive x direction does not affect the direction that the z axis now faces. Therefore, since they are independent, they can be treated as separate optimisation problems and as such, the algorithm shown in Fig. 33 was used to optimise the translational parameters.

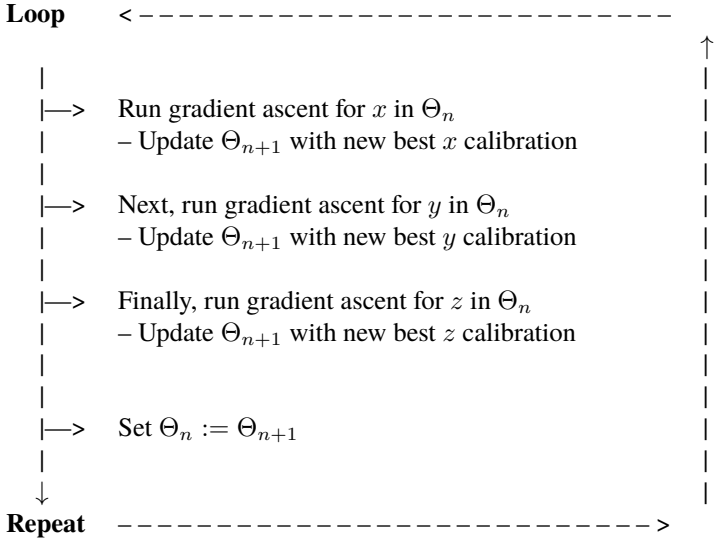


Figure 33: Algorithm for optimising the calibration parameters of $x, y,$ and z

11.2.2 Optimising ϕ, θ and ψ

The introduction of rotation angles results in the calibration parameters no longer being independent. This means that the parameters can no longer be solved individually as was the case with only x, y and z. Therefore, the parameters all need to be optimised simultaneously.

To determine which direction each angle needs to be stepped in, their gradients have to be calculated individually. This is due to the cost function/MI score being a scalar value. These gradients need to be used to step the angles together since they are dependent. This is different to the approach for x, y and z that was described in section 11.2.1, pg 25. where each parameter could be optimised on its own before moving on to the next.

The approach to optimising the calibration for all the angles is shown the algorithm in fig. 34. This shows how the gradients are determined individually and then combined to form a gradient vector. Using the gradient vector, the rotation angles can all be stepped at the same time. Then the process was repeated to find the new gradient vector for the new calibration. And again until the MI score for each big step plateaued and no better calibration parameters could be found.

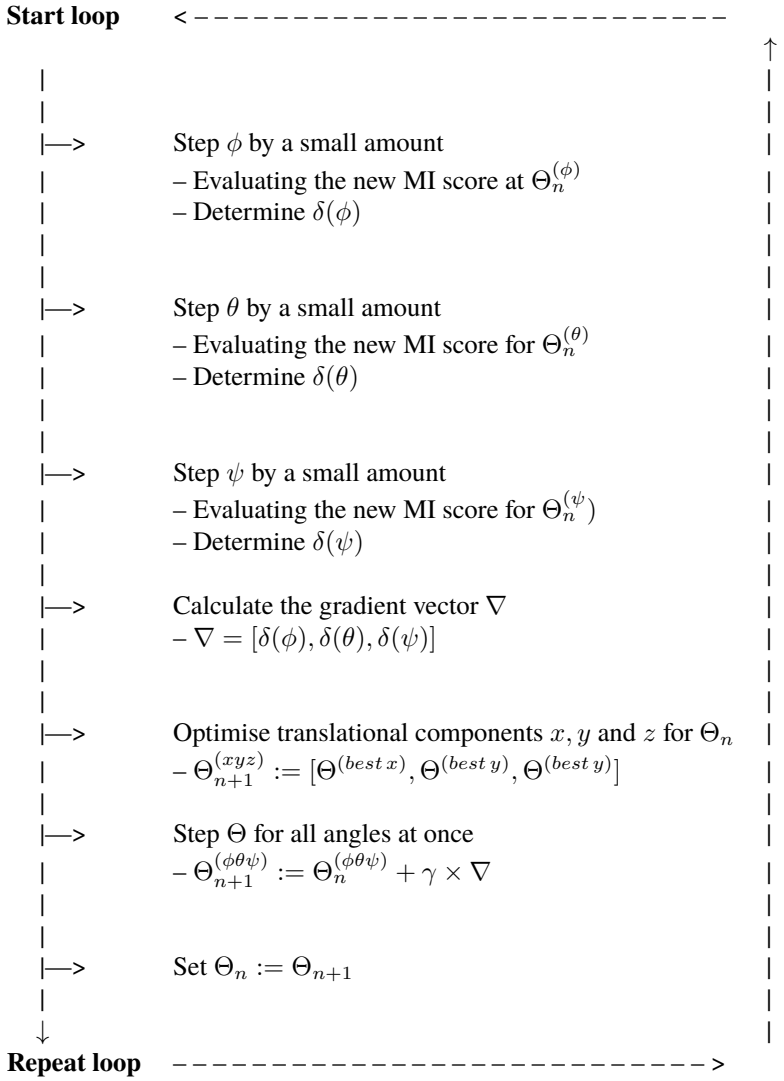


Figure 34: Algorithm for optimising the calibration parameters of $\phi, \theta,$ and ψ

The algorithm in Fig. 34 shows the complete optimisation algorithm. This is what was implemented in order to obtain the best estimation for the calibration parameters in this project. By combining the optimisation of the rotational and translational components into the same optimisation loop, the dependence of the variables was no longer a problem.

The calibration of the angles with the method described does provide a good, accurate solution. Variable step sizes may make it faster but the methods described are sufficiently quick to be implemented on vehicles with a midrange processor and no extra hardware. The accuracy obtained with this method is as accurate as the correlation metrics will allow. As a result, when these methods are used over several frames then the resulting calibration is almost as good as the ground truth. The more frames used in the averaging, the more accurate this method becomes.

11.3 Finding The Best Possible Calibration

The gradient ascent methods established in the preceding sections converge on the maximum mutual information score that can be found in each frame. These maxima are, however, all local maxima. In order to find the global maxima, gradient ascent needs to be run across several different frames that have different appearances.

The method used in this paper was to manually select a portion of the vehicles drive where the scene was well illuminated. Then 5 frames were used, spaced 10 framed apart. The optimisation was run over these 5 frames, using the local maximum calibration parameters from the previous frame as the starting point for the next. The learning rates were also reduced for the next frame such that they were at 80% of the previous frame's learning rate. This created an effective averaging and helped to finetune the estimated calibration parameters to be closer to the global maximum. By keeping each frame significantly different from the previous one, there was less chance of being stuck in a local maximum. As a result, it was easier to find the global maximum. The improvements using this method are depicted in Fig. 35

Another method used was to change the random seed used to generate the initial step directions. Noise can also be added to the starting point which results in slightly different optimisation paths and therefore different local maxima. By combining these two methods and finding a set of different "best calibration parameters" for the same set of frames, there is more chance that the actual global maximum is found. This worked the same as changing what the initial guess of the calibration parameters in order to find and compare different local maximums.

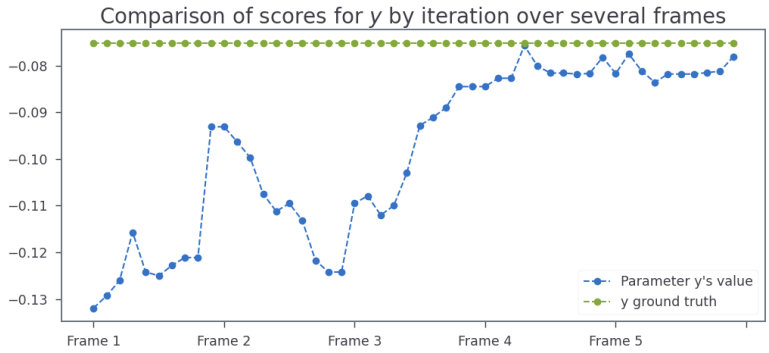


Figure 35: The depiction of the improvement form of using multiple frames

11.4 A note on the optimisation method

The optimisation methods described worked well in estimating the calibration parameters even when the initial guess caused a very small overlap of points between the lidar and camera data. Several methods can be investigated to improve the performance of these methods, they will improve the speed and reliability of the techniques described.

The optimisation of x, y and z can be done independently for each iteration. The methods described in this paper optimise these parameters sequentially for each step of the calibration optimisation. A potential avenue that was not explored due to time constraints is parallel processing of $x, y,$ and z since they all use the same starting point. This could provide close to a 3-fold speed increase to the algorithm.

The scoring used for the optimisation had to be performed without a KDE due to hardware constraints and more efficient KDE smoothing algorithm not being implemented successfully. This will affect the performance of the optimisation because the maximum likelihood estimation has a higher Mean Squared Error than a smooth with a kernel density estimate.
If a KDE is used, the results should be more accurate and consistent. [13]

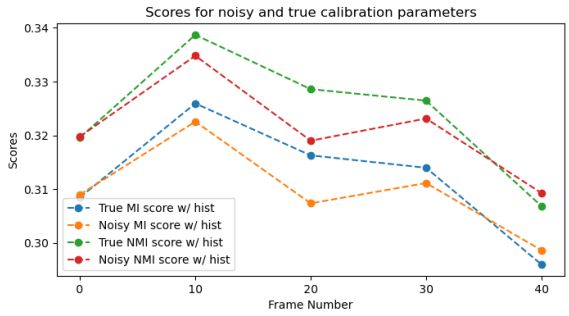
The use of maximum likelihood estimation in this paper was a compromise that was made because a KDE would, by extrapolation of an individual score calculation, require approximately 38 hours to complete the gradient ascent. This is opposed to the maximum likelihood estimation requiring roughly 5m 30s to run the gradient ascent.

Part IV

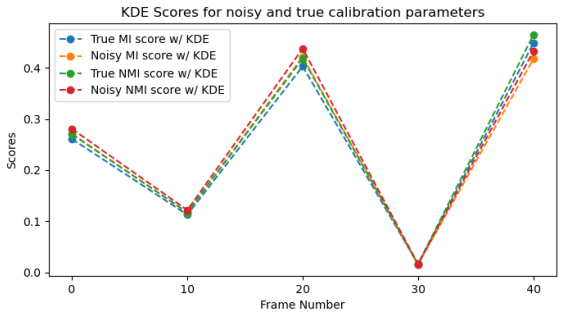
Results

12 Corellation metrics performance

To compare the performance of the correlation metrics, a noisy calibration and the ground truth calibration were scored over 5 different frames each spaced 10 frames apart. These scores are plotted in fig. 36 below.



Comparison of scores over frames when using a histogram



Comparison of scores over frames when using a KDE

Figure 36: Comparision of the KDE and histogram MI scores

The plots in fig. 36 show that the two different scoring methods produce very differently behaved results.

Type of scoring	$MI_{avg}(\Theta_{true})$	$MI_{avg}(\Theta_{noisy})$	$\sigma(\Theta_{true})$	$\sigma(\Theta_{noisy})$
MI scores (histogram)	0.3122	0.3125	0.0098	0.009
NMI scores (histogram)	0.3241	0.3242	0.0110	0.010
MI scores (KDE)	0.248	0.2484	0.1650	0.1614
NMI scores (KDE)	0.2571	0.2582	0.1714	0.1673

As can be seen in fig. 36, when comparing the scores from the KDE and histogram over several frames, the KDE scores have a much higher variation over the sequence of frames than the histogram's maximum likelihood measurement does.

In the KDE's scores, the noisy calibration parameters produce higher scores than the ground truth for two of the five frames. The mutual information scores should be higher the more aligned the sensors are and so the fact that the KDE is producing higher scores for bad alignments is undesirable.

The scores using a histogram were more consistent over the different frames, having a lower standard deviation. The true calibration also scored higher than the noisy calibration for all but the last frame. This combination of greater stability as well as the noisy calibration consistently scoring lower on most frames make the histogram a better scoring metric in this case.

In order to obtain the comparison of the corellation metrics, the data compared was the LiDAR points intensities and the image pixels that the lidar projected onto. The results of this comparison comparison are a better test of the performance as, when the data is in this form, each pixel only corresponds with pixel in the other sensor. This makes this method a better test of the correlation metrics but in practice these methods were not used as the goal was to boost the performance of the metrics.

In order to improve the reliability of the metrics and give a slight performance increase, it was observed that comparing dilated images gave better performance. To create a dilated image, the projected lidar points and the sampled camera pixels were used to create two different images that were mostly blank due to the sparsity of the lidar points. The pixels in these were dilated with a kernel of size (5,5)px. The resulting dilated images yielded more reliable mutual information scores than just comparing the flattened vector containing the intensities from the lidar or the sampled image since each data point now represents a 5x5 pixel square and so there is more chance of a match with the date from the other sensor. In essence, each pixel now corresponds to 25 other pixels using this method. While coarse, the resulting improvements outweighed the slight losses in accuracy.

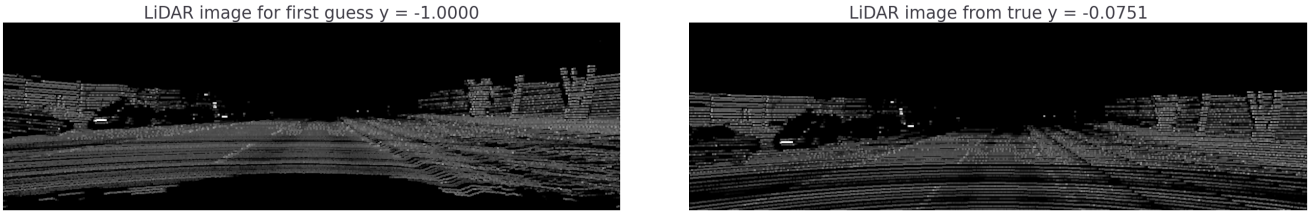
This method of using dilated images could not be used with a KDE in the optimisation as it takes an impractically long time for each KDE to be computed using the current methods for the 465 750 pixels in each image when this comparison is performed. This can be avoided in future by building the KDE smoothing from first principles in C++ as opposed to Python. The method was, however, reliably implemented using a histogram measure and the resulting performance of the scoring metric was improved.

13 Results from Optimisation

13.1 1D Optimisation

13.1.1 For only x, y and z

To test the optimisation for x, y and z, the parameters were given several different initial guesses and the outputs were checked. For comparison of results, the x, y and z parameters were all started at -1. This was done because its a worse estimate than a start of 0, which the parameters are all closer to. The assumption that if the gradient ascent can find the right outcome from a worse starting point then it can find it from an easier one. The initial guess of 0 would also be too close to the ground truth parameters and therefore not provide a good test.



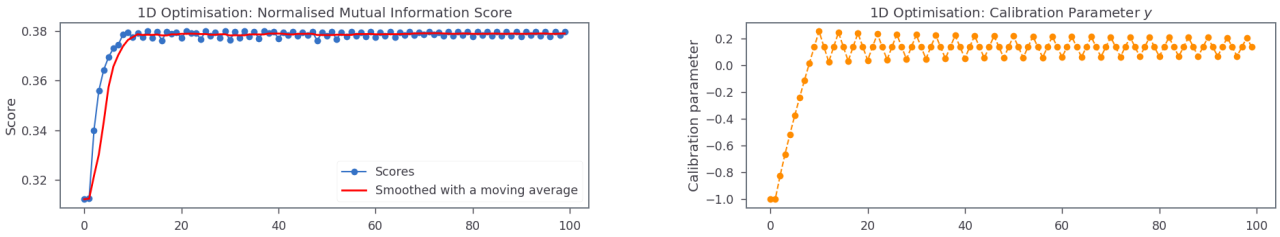
Results from an initial guess of $y = -1$ | Projections for ground truth y

Figure 37: Images comparing the correct and incorrect of y

The images in 37 hopefully give an understanding of how fine the calibration needs to be compared to the sensitivity of changes in the translational components of Θ. A starting point of -1 is more than one hundred times greater than the ground truth for y. This gives a large error quantitaatively but looking at the resulting projections of the lidar, there is not a big difference in the two. This shows the challenge of the sensitivity of this problem and highlights the need for very good correlation metrics.

The figures above in 37 show that the initial guess of $y = -1$ projects the lidar too high on the image. The difference is small but still produces a significant change in the NMI score. There is enough of an error for this initial guess that the roof of the vehicle is now in the image which shows as a shadow in the bottom of the left image in fig. 37

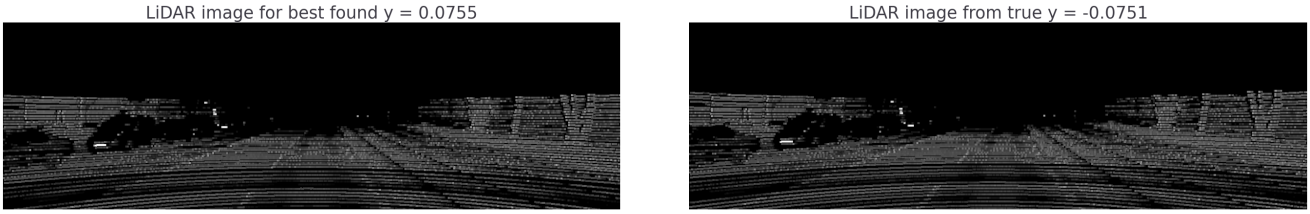
The results of running the 1D optimisation for the parameter y are shown in fig. 38



Change in NMI score over the optimisation | Change in the value of y

Figure 38: Plots of the NMI score and the value of parameter y over the optimisation

The plots in 38 both show that there is a strong local maximum that has been found. This is indicated by the flat portions of the graph where a better calibration parameter could not be found. The local maximum was found in a comparatively very short amount of time when compared to the optimisation of a single angle.
The resulting local maximum was then used to project the lidar data onto the camera as displayed in figure 39.



Final output after gradient ascent for y | Projections for ground truth y

Figure 39: Comparison of the projections for the optimised y with that of the ground truth

Qualitatively comparing the projection of the optimised y versus the ground truth as shown in fig. 39, the two are very close to being the same. It is very hard to distinguish where there are differences in the two. A quantitative comparison of the optimised components versus their ground truth is shown in Table 2

	x	y	z
Initial guess	-1.0000	-1.0000	-1.0000
Final values from optimisation	-0.0745	0.0755	-0.6409
Ground Truth	-0.0028	-0.0751	-0.2720

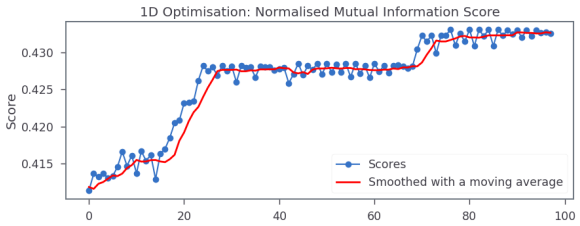
Table 2: Comparison of the calibration parameters for x, y and z before and after optimising them individually

Comparing the numbers found from the optimisation after running each parameter's optimisation for 50 iterations, the end values are significantly closer to the ground truth. Considering that the starting point of -1 was far from the end value as well as how close the optimisation got to the end value, this is a very accurate method.

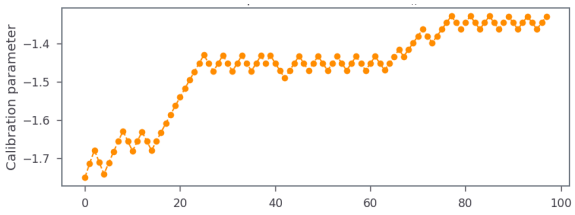
The results will become more accurate as the number of iterations increases and the optimisation is averaged over different frames. But for optimising using only one frame, this result is a significant improvement over the intial guess and most likely enough to call the system calibrated using only this one frame.

13.1.2 Optimising Angles Individually

To optimise only one angle, the rest of the calibration parameters in Θ were set to their ground truth values. The angle to be optimised was then given some amount of gaussian noise that would result in the sensors being misaligned. For a rotation of θ, the optimisation of only an angle yielded the following results:



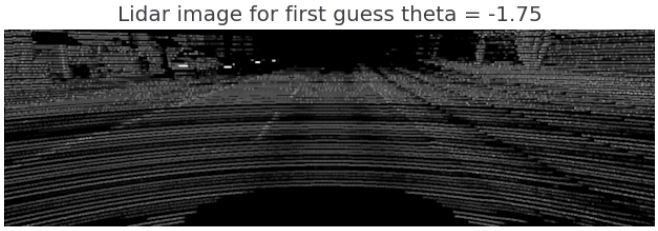
Mutual Information Scores for θ



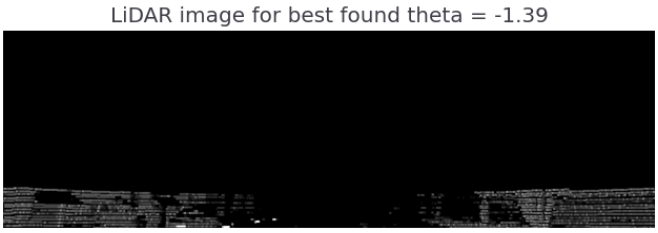
Plot of the change in θ over the steps in optimisation

Figure 40: Plots from the optimisation of θ, the roation about the $y - axis$

As can be seen in fig.40, the scores for θ have two flat areas. These areas are indicative that the alignment of the images has two local minima in its path. The optimisation does, however, settle on a value of -1.39 rad for θ at the end.



Starting point for optimisation of θ.

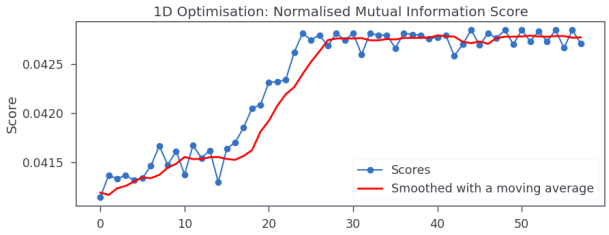


Final output after gradient ascent for θ.

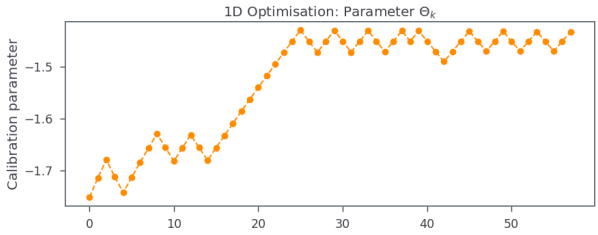
Figure 41: The plots of before and after when optimising θ given an initial starting point of $-1.75\,rad$.

As can be seen in fig. 41, the best mutual information score was found when there was less data to compare. This is because it is easier for fewer points to have a good correlation score than it is for many points to have the same correlation. As a result, the highest score was found by shifting the lidar points downwards until there were fewer points that could be projected back onto the camera image.

When the threshold limiting the minimum number of lidar points projected onto the image was introduced, this resulted in the second local minimum not being found. The best score from this optimisation is a closer match to what the image from camera 0. A comparison of the θ found using the thresholding method and the ground truth θ is shown below in fig. 43.

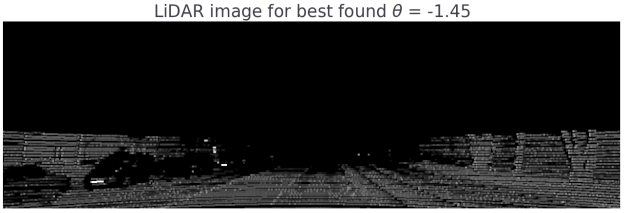


Plot of the MI scores with thresholding



Value of θ with thresholding in place

Figure 42: Plots of the optimisation process for θ



Optimised θ using the threshold technique

Lidar projected using the ground truth θ

Figure 43: Projections of the lidar points

While there is still much room for improvement, thresholding provides a solution for the problems faced by MI scores when there are fewer data points to compare. It should be noted that the threshold needs to be tuned in order to find a threshold that can work over several scenes. This may present a barrier to making this calibration process fully unsupervised.

	ϕ	θ	ψ
Ground Truth	0.0119	-1.5603	1.5483
Initial guess	0.0139	-1.4313	1.6392
Final values from optimisation	-0.2397	-1.450	1.4723

Table 3: Comparison of ϕ, θ and ψ before and after individually optimising

With the exception of ϕ, the results after optimising are close to the ground truth. These optimisations were only run once on the first frame of the dataset and so the angles obtained are the local maxima for frame 0. To get closer to the true value, the optimisation is run over several different frames, decreasing the starting sizes of the steps after every frame.

The results are as follows for running the optimisation twice on each frame over five frames with a decrease to the starting step size of 20% every frame:

	ϕ	θ	ψ
Ground Truth	0.0119	-1.5603	1.5483
Initial guess	0.0139	-1.4313	1.6392
Final values from optimisation	0.0075	-1.5650	1.5463

Table 4: Comparison of ϕ, θ and ψ before and after individually optimising over 5 frames

As can be seen from the results in table 4, the performance of the algorithm is much more accurate with only five frames. The more frames that are included, the more accurate the calibration estimation becomes. It also shows that this method does not require many frames to get a good result which shows that it can be used without requiring many resources.

13.2 Multi Dimension Optimisation

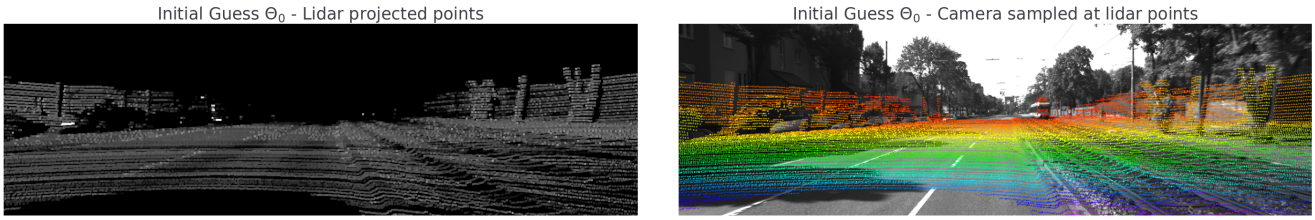
13.2.1 For only x, y and z

To begin the optimisation, the values of x, y and z were given a starting guess of:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Recall that these distances are from the Velodyne LiDAR to Cam0 on the car and the measurements are in the vehicle frame as shown in 10.

Starting using this initial guess for the calibration produced the following images:



Lidar points for initial guess coloured by reflectivity.

Lidar points overlayed onto Camera 0 and coloured by depth

Figure 44: Projections from the initial guess parameters

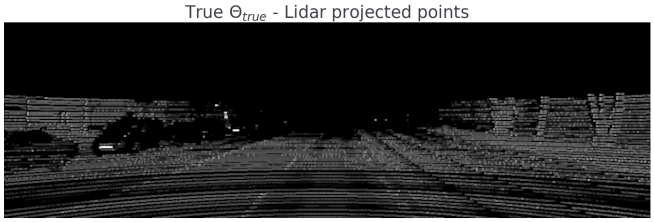
Where the lidar is overlayed onto the camera image, these points are coloured by the distance of the surface. These points were coloured to make them easier to see them on the grayscale image. **Green** means a closer to the camera while **red** means that the point is far away.

Note the large shadow at the bottom of the LiDAR points caused by the roof of the vehicle that the lidar is mounted on. Also note the way the points are out of alignment with the tree on the right of the image.

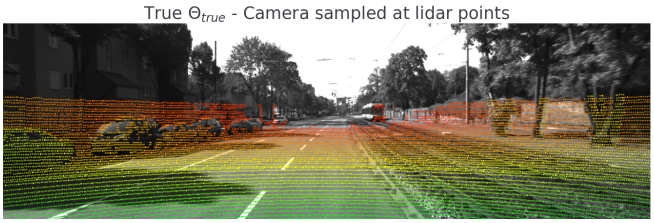
As a comparison, the "ground truth" supplied with the dataset was:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.00279682 \\ -0.07510879 \\ -0.2721328 \end{bmatrix}$$

This is significantly different from the initial guess of calibration parameters. The goal of the optimisation is to produce an output that has the same visual appearance as the projection from these ground truth calibration parameters.



Lidar points coloured by reflectivity

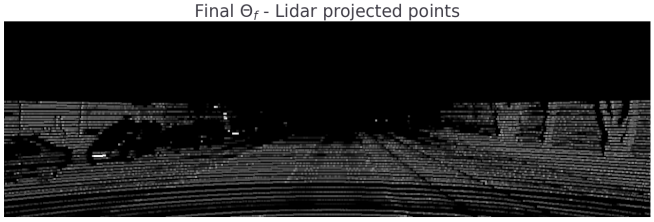


Lidar points overlayed onto the same Camera 0 and coloured by depth

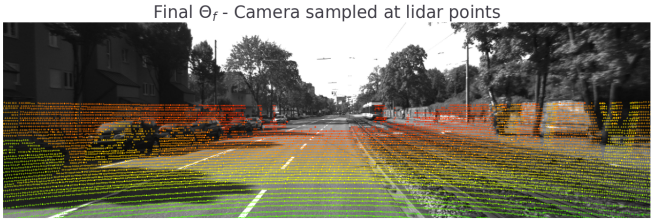
Figure 45: Projections using the ground truth

The images in 45 show the projection using the ground truth which is the ideal output from the optimisation.

The optimisation algorithm described in Part 11 was then implemented for 5 different frames, each spaced 10 frames apart. This resulted in a set of calibration parameters for x, y and z that yielded the following plots:



Lidar points after optimisation



Lidar after optimisation, coloured by depth and overlayed on Camera 0

Figure 46: Images projected using the parameters after optimisation

The image made by projecting the lidar points using the final calibration output by the optimisation (fig. 46) is visually much closer to the ground truth than the initial guess. The large shadow is no longer part of the image and the arcs produced by the sweep of the lidar beam are much closer to matching the ground truth.
On closer inspection of fig. 46, one can observe that the calibration parameters are not however perfect. Although the points showing the branches of the tree are now much closer to being aligned with the camera's image, they are off by a couple pixels. The cars are very close to being perfectly aligned with the windows and wheels being almost perfectly placed on their corresponding locations in the image from Camera 0. There is however still a slight error in the calibration which is just noticeable.

Qualitatively the calibration is almost perfect. Now to compare the values that were found. The values of the calibration parameters x, y and z, for each step are placed in table 5 below

	x	y	z
Initial guess	-1.0000	-1.0000	-1.0000
Final values from optimisation	0.0577	0.2582	-0.8082
Ground Truth	-0.0028	-0.0751	-0.2721

Table 5: Comparison of the calibration parameters before and after optimising

As can be seen from the values found for the x, y and z parameters, the optimisation does not reach ground truth values. These values provide a significantly better estimation of the calibration parameters than the initial guess but there is still a significant error margin. To the eye of an observer, the calibration is more than accurate enough to align the Lidar and Camera sensors but the values are still slightly different
Possible causes for the wide error margin are discussed in more detail in Part V.

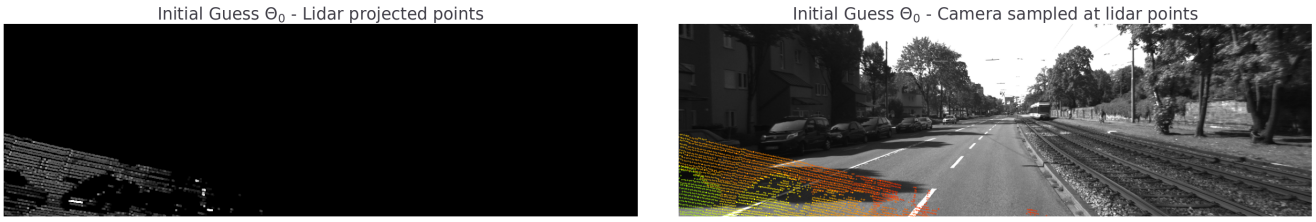
13.2.2 Including angles in the optimisation

For the full optimisation with the angles included, the results were as shown below:

The initial guess, created by adding noise, was:

$$\Theta_0 = \begin{bmatrix} -0.0011 & -0.1319 & -0.0149 & 0.016 & -1.3023 & 1.7301 \end{bmatrix}^T$$

This resulted in the projections shown in Fig. 47.

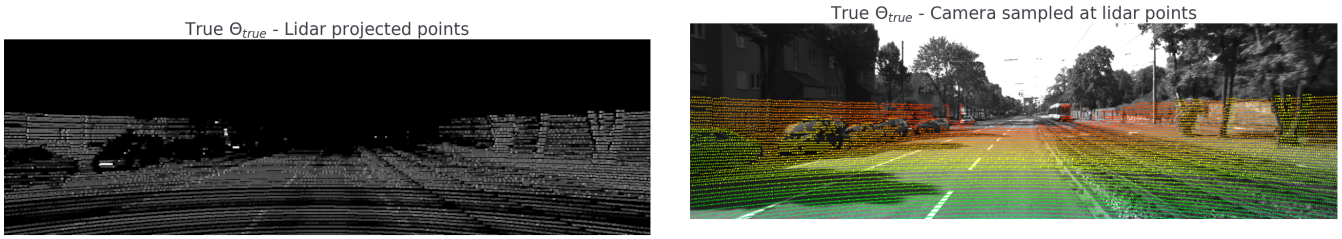


Lidar points for initial guess coloured by reflectivity.

Lidar points, coloured by depth, overlayed onto Camera 0

Figure 47: Projections from the initial guess parameters

The goal for the optimisation is the ground truth that results in the images shown in Fig. 48



Lidar points coloured by reflectivity

Lidar points overlayed onto the same Camera 0 and coloured by depth

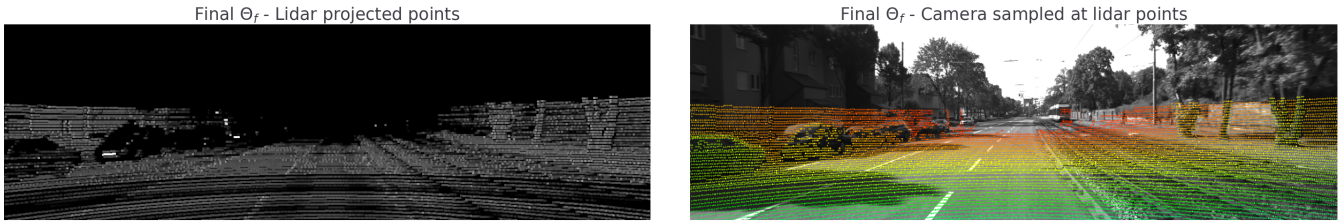
Figure 48: Projections using the ground truth

The optimisation was run with 5 different frames, each 10 frames apart. The resulting calibration parameters are displayed in table 6. These results were much closer to the ground truth than the initial noisy calibration parameters. They may have been closer if more frames were used but these results were satisfactory. The optimisation ran in 5m 26s when using only one core of a 9th gen intel i5. With some optimisation to the code this can be improved greatly.

	x	y	z	ϕ	θ	ψ
Ground Truth	-0.0028	-0.0751	-0.2721	0.0119	-1.5603	1.5483
Initial guess	-0.0011	-0.1319	-0.0149	0.016	-1.3023	1.7301
Final values from optimisation	-0.0318	-0.1038	-0.0079	0.0117	-1.549	1.5388

Table 6: Comparison of the calibration parameters of Θ before and after individually optimising over 5 different frames

To compare the qualitative findings, the results of the optimisation were used for the projection of the lidar data, it resulted in the images as shown in fig. 49.



Lidar points after optimisation

Lidar after optimisation overlayed on Camera 0 and coloured by depth

Figure 49: Images projected using the parameters after optimisation

As can be seen from the lidar that is projected onto the camera image, the lidar points match very closely with the features in the image. There are some errors, namely the lidar pixels do not line up perfectly with the train and the wheels are not perfectly aligned with the corresponding lidar data.

For only 5 images, the results are very close to the ground truth. Qualitatively and quantitatively being quite close to the goal output compared to the initial guess. This method would provide better results the more frames are used but for the purposes of this paper 5 was deemed enough to get satisfactory performance while still keeping the time for the optimisation reasonably short.

Part V

Reflection on findings

14 Reflection on Choice of Dataset

The KITTI dataset was chosen because it was simple and lightweight while also being easy to access and interface with the data. This aided in keeping this part of the project simple.

Many of the scenes in the KITTI dataset were of open roads. This lead to spots where the lidar was not able to read any data in some patches, due to the surface being too far away, while the camera was still able to capture the object. At these longer distances, small changes to the x, y and z components of the calibration made little difference. However,small changes to the angles would create large changes at distance. While these factors made the calibration more challenging, having these open road scenes was beneficial as it provides a better understanding of how these techniques would perform in the real-world environment.

An improvement could be made by having a form of pre-processing implemented to select the best scenes for the calibration. This can be done by using a scene descriptor metric such as in [14] to choose scenes that would provide a better, more reliable mutual information score. This will allow for finer calibration of the x, y and z calibration parameters. As seen from the results, this is only necessary if the calibration parameters need to be incredibly fine.

The open road scenes used in this paper still resulted in good estimates of the calibration parameters that produced projections that were very close to being perfectly aligned. The KITTI dataset is a good choice of a dataset when testing these techniques for real-world application.

15 Transformations between Lidar and Cameras

The transformations between sensor frames are a cornerstone of this paper as, if they are implemented incorrectly, there is be no way to find the correct calibration parameters. The transformation between the lidar and camera frames was implemented successfully using the equations and methods described in the methodology section of this paper.

The current setup of frames allows for the lidar to be projected onto any camera's image frame without any further calibration, provided that the transformation from a camera's image frame to the rectification frame was known. Therefore, the projection of the lidar points onto any camera is trivial to compute since the lidar is always projected to the rectification plane before being projected to a camera's image frame.

A limitation with the current method is that an Euler angle singularity does exist when the pitch approaches $\pm 90^o$. In the case of cars and most ground vehicles that rarely need to look straight up or down, this should not be a problem. Since these methods can also be used to calibrate camera-lidar pairs in many different environments, there is potential for use in other vehicles where gimbal lock may become a problem. If these methods were to be applied to other vehicles such as drones, the singularity constraint would need be solved. The rotations can be written such that quaternions be used instead to describe the pose and rotations which would solve this problem.

16 Reflection on Correlation Metrics and Scoring Methods

The results from testing the correlation metrics showed that they provide a reliable way of quantifying the alignment of two different types of sensors even when the orientation of an object in the frame is slightly different. This confirms the findings of [12].

There is however still room for improvement. The results found when finding the mutual information score using a KDE could not be explored sufficiently because the time taken to calculate a KDE for each comparison was impractical. If a KDE smoothing method is created from scratch and optimised for this application such that it can be computed in a reasonable time frame, this method can be investigated further.
Ultimately, the kernel density smoothing was not time efficient enough to implement practically. This prevented it being debugged and tweaked to run as well as the histogram-based maximum likelihood estimation. If a KDE smoothing function could be

implemented effectively, it would most likely provide better comparisons between the alignment of the images. This KDE would need to be much faster in order to be practical for real-world application.

The implementation of the histogram-based maximum likelihood estimation was shown to be an effective metric for quantifying the alignment of the sensor data. The higher mean squared error [13] did not seem to affect the accuracy to a large degree as shown across the different figures in fig. 24. The use of MLE Normalised Information Score in the optimisation proved to be a metric that is more than capable of finding accurate estimations of the ground truth calibration parameters as is shown in the results of the final optimisation.

Introducing a threshold that assumes a minimum number of points that will be projected into the image plane provided a performance increase in the optimisation. This effectively created a buffer zone that penalised the score for any projections that resulted in fewer points than the threshold. In order to not bias the results, the threshold was set substantially lower than the number of points that project into the frame under an average alignment. Setting the threshold to <30% resulted in good performance while not aiding the algorithm. Without the threshold, the calibration parameters were still found. There was, however, an increased chance of the local maximum being where there were very few points to compare. This was because it is easier for a few points to correlate well than it is for many points.

17 Optimisation of the calibration parameters

The methods discussed in this paper allowed the calibration to be optimised even when there was a small initial overlap between the sensors as shown in fig. 47 due to the initial guess for the calibration parameters being far off from the truth.

When the optimisation of a variable was run on only one frame, the resulting images were more aligned than the starting point, however, it was commonplace for the optimisation to have been stuck in a local maximum. Occasionally, the translational calibration parameters that were found using one frame would be further from the ground truth than the starting point. When the optimisation was run over several different frames, all while reducing the step sizes, the results were much closer to the ground truth and the values obtained were significantly more accurate.

The effective averaging by optimising over the different frames was key to avoiding local maxima and finding the best calibration that this method could achieve. However, some room for improvement still exists. Using different frames from different routes during different times of day would produce a more robust result. The selection of optimal scenes for use with the calibration would also result in better performance.

The optimisation implemented in this paper resulted in an alignment of the sensors that was *qualitatively* extremely close to a perfect alignment but there were a number of small errors that would prevent the result being classified as a *ground truth* alignment. Considering that only five different frames were used during the optimisation, the results surpassed the expectations of the researcher.

Quantitatively, the results obtained were close to the ground truth especially for the rotational parameters (see table 6). The long street scenes may have biased the results since small changes in translational motion make small differences at a large distance while the opposite is true for angles. This may have been the reason that the rotational parameters were much closer to the ground truth than the translational parameters. With the exception of the x parameter, the optimisation provided a significant improvement over the initial guess for all parameters.

The results from the methods described in this paper are accurate enough to provide a good understanding of the distance to any object in the image frame by using the projected lidar data. This means that the resulting calibration is sufficiently accurate to be used by a vehicle and therefore the results satisfy the aim of this paper.

18 Opportunities for Future Improvement

The biggest improvement that can be made to the methods described in this paper is to implement scene selection. The paper by Scott et al. [14] provides a good description of how a non-parametric scene description metric can be implemented as a preprocessing step before the spatial calibration methods described in this paper are implemented. This would ensure that the scenes that are used in the optimisation will have a concave mutual information scoring without having more than one local maximum. This will improve the accuracy of the results.

Using more scenes for the optimisation will also improve the accuracy of the result. A variation of scenes will help avoid a local

maximum that may arise when the calibration is run for frames in the same general environment. Data from different routes will provide a variation to the scenes and improve the estimation of the global maximum.

Preprocessing the images using blurring functions will average the grayscale intensities of an object in an image. This averaging would reduce the effects that the difference in perspective of the two sensors will have. As a result, blurring would help with the coarse alignment of the sensors and better the alignment results when the initial guess is far from the ground truth. After several iterations, the unblurred images would be then used for fine-grain calibration. This combination of coarse and fine calibration can be used to decrease the total number of iterations required for the optimisation.

Part VI

Conclusion

As vehicles become more autonomous, their dependence on their sensors only increases. Self-driving vehicles are part of our reality. Warehouses, roads and skies are seeing increasing automation and all of these vehicles need to make sense of their environments. In the real-world environment, the vibrations and impacts due to everyday movement can cause sensors to drift from their original calibration. This paper establishes a method that spatially calibrates lidar-camera sensor pairs without needing any additional information about the environment. Implementation of these methods will decrease calibration downtime, allowing vehicles to self-calibrate in the field. The calibration does not require user input while being targetless and using the vehicles surrounding environment to determine the spatial calibration parameters. The proposed method is independent of object classification. It has lower computing power requirements and can be implemented using existing onboard processors. The proposed spatial calibration method is versatile enough to be used by vehicles in diverse environments while remaining accurate and robust. This method only needs a relatively small overlap in sensor data to yield results that look almost the same as the ground truth.

In this paper, a method of non-parametric spatial calibration was presented and shown to work accurately. This demonstrates the effectiveness of statics based correlation metrics as a quantification of the alignment of lidar-camera sensor pairs. The correlation between lidar reflectivity and grayscale camera intensity is caused by both sensors relying on the same physical property of a surface. Lidar reflectivity and pixel intensity are both determined by the amount of light that is reflected by an object. This dependence is the reason these statistic based metrics work for quantifying lidar-camera alignment as is shown in this paper.

While KDEs can be used to improve the NMI scores, this paper demonstrates that they are not necessary for an accurate result. The use of histogram-based maximum likelihood estimation to calculate the NMI score is shown to yield alignments qualitatively analogous to the ground truth. This results in substantially faster calibration times and lower computational demands.

The accuracy of this method is dependant on the number of frames that can be processed. With as few as five frames, the resulting parameters yield a projection where the points predominantly in alignment. The methods developed in this paper can be adapted to be implemented on-the-fly. Therefore, many frames can be used in the calibration estimation, resulting in increasingly accurate projections.

The method presented can calibrate the lidar-camera pair using the environment surrounding a vehicle. There is no constraint on what needs to be in the surrounding environment since the method relies on the physical properties of the surfaces in the environment. This method saves time by removing the need to set up a calibration environment outfitted with calibration targets. There is no need for additional sensors and the computational requirements are not demanding.
The majority of existing methods use calibration targets, require additional sensors or large amounts of computational resources. This proposed method results in an improvement on these existing methods while still providing accurate results. The proposed method is a more elegant way of spatially calibrating the lidar-camera pairs in a variety of vehicles and environments. Therefore, this proposed method has the potential to further enrich the field of autonomous vehicles.

19 Acknowledgements

Special thanks need to be given to Andreas Geiger and team for the ongoing support and maintenance of the KITTI dataset. This paper would not have been possible without the data it provided.

An immense debt of gratitude is owed to my friends and family for all the love and support shown throughout the writing of this project. The foundation they built provided the stability needed to complete this project in the unprecedented times resulting from the COVID-19 pandemic of 2020.

Finally, I would like to thank my supervisor Dr Paul Amayo for his unwavering guidance and support. His patience and willingness to help at any time, no matter how trivial or complex the question, is deeply appreciated.

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[2] Q. Le and A. Ng, "Joint calibration of multiple sensors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, Nov. 2009, pp. 3651–3658.

[3] B. Fu, Y. Wang, X. Ding, Y. Jiao, L. Tang, and R. Xiong, "LiDAR-Camera Calibration Under Arbitrary Configurations: Observability and Methods," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3089–3102, Jun. 2020.

[4] Q. Zhang and R. Pless, *Extrinsic Calibration of a Camera and Laser Range Finder (Improves Camera Calibration)*, Jan. 2004, vol. 3.

[5] Z. Pusztai and L. Hajder, "Accurate Calibration of LiDAR-Camera Systems Using Ordinary Boxes," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. Venice, Italy: IEEE, Oct. 2017, pp. 394–402.

[6] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA: IEEE, Oct. 2007, pp. 4164–4169.

[7] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, Jun. 2013.

[8] N. Andreff, R. Horaud, and B. Espiau, "Robot Hand-Eye Calibration Using Structure-from-Motion," *International Journal of Robotics Research*, vol. 20, pp. 228–248, Mar. 2001.

[9] Z. Taylor and J. Nieto, *Motion-Based Calibration of Multimodal Sensor Arrays*, May 2015, vol. 2015.

[10] Z. Taylor, J. Nieto, and D. Johnson, "Automatic calibration of multi-modal sensor systems using a gradient orientation measure," in *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 1293–1300.

[11] B. Nagy and C. Benedek, "On-the-Fly Camera and Lidar Calibration," *Remote Sensing*, vol. 12, no. 7, p. 1137, Apr. 2020.

[12] P. Viola and W. Wells, "Alignment by Maximization of Mutual Information," in *International Journal of Computer Vision*, vol. 24, Jan. 1995, pp. 16–23.

[13] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information," *Journal of Field Robotics*, vol. 32, Sep. 2014.

[14] T. Scott, A. A. Morye, P. Pinies, L. M. Paz, I. Posner, and P. Newman, "Choosing a time and place for calibration of lidar-camera systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden: IEEE, May 2016, pp. 4349–4356.

[15] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Weinberger, *Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving*, Jun. 2019.

[16] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000km: The oxford RobotCar dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[17] A. Heyden and M. Pollefeys, "MULTIPLE VIEW GEOMETRY," . *Projective Geometry*, p. 63.

[18] T. D. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017.

[19] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[20] M. Ragab and K.-h. Wong, "Rotation within Camera Projection Matrix Using Euler Angles, Quaternions, and Angle-Axes." *I. J. Robotics and Automation*, vol. 24, Jan. 2009.

[21] E. G. Learned-Miller, "Entropy and Mutual Information," p. 4.

[22] J. Hwang, S. Lay, and A. Lippman, "Nonparametric multivariate density estimation: A comparative study," *IEEE Trans. Signal Process.*, 1994.

[23] S. Fortmann-Roe, R. Starfield, and W. Getz, "Contingent Kernel Density Estimation," *PloS one*, vol. 7, p. e30549, Feb. 2012.

[24] C. Maksimov, "Gradient Ascent Algorithm," https://chingismaksimov.com/2020/03/28/gradient_ascent.html, Mar. 2020.

[25] L. Clement, "utiasSTARS/pykitti," STARS Laboratory, Oct. 2020.

Appendices

A Accessing the source code

The source code is available on the authors github page. The link to this is:
https://github.com/Katsie011/KTSMIC005-Final-Year-Project

B Interfacing with the dataset

To avoid importing the data by each file, the *pykitti* library written by Clement, L from the STARS Laboratory at the University of Toronto [25] as it provided an elegant and efficient way to access the dataset in Python.

This library was used to import a recording of a vehicle's drive from the KITTI dataset. It was then used to make an object which contained the chosen selection of frames. This object could be used to access the LiDAR point clouds, camera images as well as the correction matrices for the images. The library was also used to import the default spatial calibration parameters in a more accessible format.

C Final results

Figure 50: Projection from the initial guess for the spatial calibration parameters. Lidar is coloured by distance and projected onto camera 0

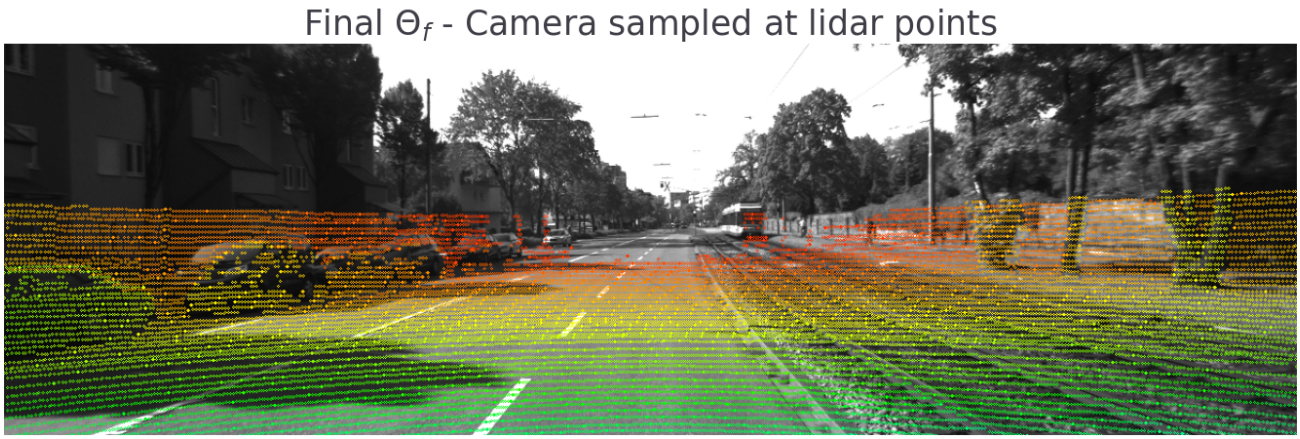


Figure 51: Results of calibrating the lidar camera pair. Lidar is coloured by distance and projected onto camera 0