CHALMERS
UNIVERSITY OF TECHNOLOGY

Extrinsic Calibration of Exterior Cameras Using the Car Body as Reference

Degree project report in Complex Adaptive Systems

ERIC WAHLSTRÖM & EFRAIM ZETTERQVIST

DEPARTMENT OF SPACE, EARTH AND ENVIRONMENT

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project report 2025

Extrinsic Calibration of Exterior Cameras Using the Car Body as Reference

ERIC WAHLSTRÖM & EFRAIM ZETTERQVIST

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Space, Earth and Environment
Chalmers University of Technology
Gothenburg, Sweden 2025

Extrinsic Calibration of Exterior Cameras Using the Car Body as Reference
Eric Wahlström & Efraim Zetterqvist

Supervisor: Elin Källman, Volvo Cars
Supervisor: Rasika Somalwar, Volvo Cars
Examiner: Claes Andersson, Department of Space, Earth and Environment

Degree project report 2025
Department of Space, Earth and Environment
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: An image of Volvo EX90. The figure is taken from the Volvo Cars website.

Typeset in LaTeX
Gothenburg, Sweden 2025

Extrinsic Calibration of Exterior Cameras Using the Car Body as Reference
Eric Wahlström & Efraim Zetterqvist
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

Ever since exterior cameras started to be part of modern vehicles, accurate extrinsic calibration has been a vital part of the production in order to deliver reliable advanced driver assistance systems. The latest car models from Volvo Cars use several cameras to assist the driver and increase the safety, which means an accurate calibration is of utmost importance. This thesis investigate the possibility of performing an extrinsic calibration using the car body as reference to replace the current factory calibration that uses static chessboard targets. In total, five different cameras were tested, four being identical fisheye cameras placed in different positions around the car and the last one being a pinhole camera placed at the top of the windscreen.

For images captured by the exterior cameras, different computer vision techniques were used in order to identify reference points in images. By projecting the points into 3D coordinates, the orientation angles could be determined by comparison to the nominal 3D coordinates that are known from the CAD model of the car in question. The final algorithm was tested on simulated images in which the camera had been rotated in all integer steps between $-3°$ and $3°$. All generated results had a maximum absolute error of below $0.3°$ with the fisheye front camera performing the best with a maximum absolute error below $0.2°$. The algorithm was also tested on five real cars in which the car body method could be compared to the current factory calibration method. Both methods produced similar results for some angles, although there were certain cases that showed significant differences. Since it is not possible to validate the true orientation of the camera in a non-simulated image, no definitive conclusion could be drawn about which method had the highest accuracy for real images. However, by overlaying a real image with a simulated image rotated using the calibration angles, the car body method appeared to be more accurate than the current factory calibration method.

Keywords: Calibration, Cameras, Car Body, Pitch, Roll, Rotation, Yaw.

Acknowledgments

First of all, we would like to thank both of our supervisors, Elin Källman and Rasika Somalwar, for giving us the opportunity to carry out this Master's thesis and for all their help during our work. We would also like to thank all our colleagues at Volvo Cars who took the time to answer our questions during the project. Finally, we want to thank our examiner, Claes Andersson, for taking on the role and for his help with the thesis.

Eric Wahlström and Efraim Zetterqvist, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BFGS	Broyden-Fletcher-Goldfarb-Shanno
CAD	Computer-Aided Design
FOV	Field Of View
MAE	Mean Absolute Error
OpenCV	Open Source Computer Vision Library
RMSE	Root Mean Square Error
ROI	Region Of Interest

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

u, v	Indices for pixel coordinates in an image
i, j	Indices for pixel coordinates in a second image

Sets

$\mathbf{X}^{ref}$	Set of nominal reference points
$\mathbf{X}^{rot}$	Set of rotated points

Parameters

I	Input image
T	Template image
σ	Standard deviation
k	Kernel size
k_i	Distortion coefficient for radial distortion
p_i	Distortion coefficient for tangential distortion
w	Width of an image
h	Height of an image
K	Intrinsic camera matrix
f	Focal length
c	Principal point
α	Step size

n	Number of reference points
s	Depth scale factor

Variables

$\mathbf{X}$	3D point
$\mathbf{X}'$	Rotated 3D point
$\mathbf{x}$	2D point
$\tilde{\mathbf{x}}$	Homogeneous representation of a 2D image point.
G_x	First derivative in horizontal direction
G_y	First derivative in vertical direction
G	Gradient magnitude
Θ	Angle of the gradient vector
r	Distance from the center of the image
R	Rotation matrix
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
R^2	Coefficient of determination
$\mathbf{p}$	Search direction
$\mathbf{v}$	Parameter vector
$\mathbf{y}$	Gradient difference
$\mathbf{s}$	Step vector
ρ	Curvature scalar
B	Hessian matrix

Contents

List of Figures

List of Tables

1

Introduction

Over the years, cars have evolved from mechanical machines into complex systems that integrate advanced electronics and sensors to improve safety, efficiency, and autonomy. Among these sensors, cameras play a crucial role in enabling applications such as lane detection, obstacle avoidance, and autonomous driving. Each year, the automotive industry takes more steps towards making cars self driving. In doing so, the cameras on the cars play a significant role and it is of high importance that they are well calibrated. However, calibration of the cameras can be both time and resource consuming.

1.1 Background

Traditional extrinsic calibration of cameras often involves some black and white targets with a chessboard structure. Chessboards are commonly utilized in camera calibration due to their easy construction and the planar grid pattern, which provides numerous distinct feature points in an image. In an image of a chessboard target it is easy to find the corner points by using different computer vision techniques. If one also knows the exact location of the target, it is straightforward to calibrate the cameras from there. This method usually provides high accuracy with a fully automated process [1]. This is why chessboard targets are the standard practice when calibrating cameras today.

This thesis was conducted at Volvo Cars and today they use stationary targets to calibrate their cameras. This process takes longer time than desired, because two reference systems are being used. One for the car where the cameras are mounted and another one for the placement of the targets. In order for the calibration to be precise, the placement of the car needs to be extremely precise, which takes a long time and has a high maintenance cost. For a company that produces thousands of cars in factories all over the world, there is much profit to be made if the production time can be reduced.

This thesis will explore an alternative calibration method to determine whether the current target based approach can be replaced. The idea is to not use any form of external targets, but instead to take advantage of the visibility of the car body

within the field of view (FOV) of the cameras. The initial focus will be on assessing whether precise camera calibration can be achieved using the car body. If successful, the thesis will examine whether the solution is possible to implement in Volvo Cars factories in the future.

Today, Volvo Cars have several cameras on the car with different optical properties and FOVs. This thesis will explore the possibility of using the car body to calibrate one pinhole camera and four different fisheye cameras. The four fisheye cameras are of the same model and are placed in different positions around the car, together giving a 360° view around the car. The pinhole camera is placed at the top of the windshield facing forward. The fisheye cameras are placed around the car with one in the front, one in the back, one on the left and one on the right. In Figure 1.1 one can see the placements of the cameras used in this project and an approximation of their FOV.

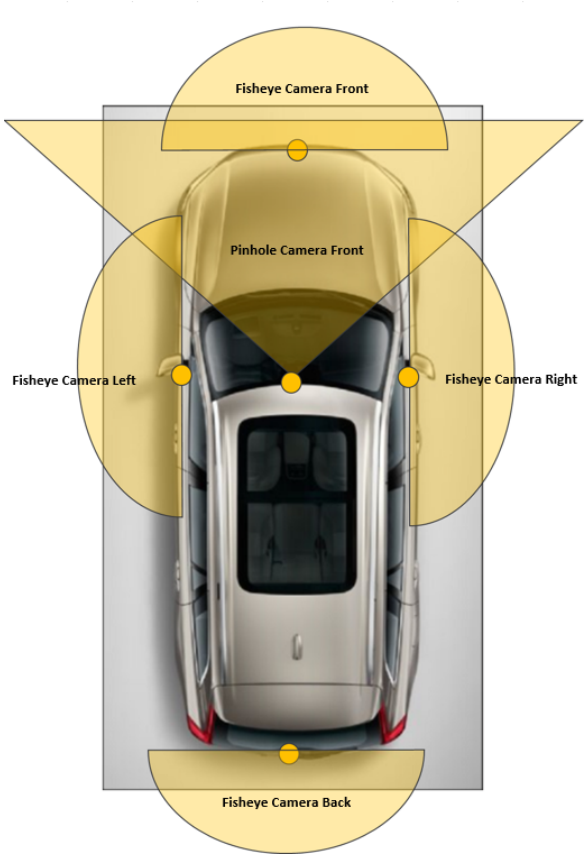


Figure 1.1: The position of the cameras on the car. The darker yellow dots are the position with the lighter yellow areas being an approximated field of view for the cameras.

A pinhole camera is a simple imaging device that captures visual information without the use of a lens. Instead, light from the environment enters through a small aperture, projecting an inverted image onto a sensor or surface inside the camera. The size of the hole affects the image quality. Smaller apertures enhance sharpness

but reduce brightness, whereas larger ones increase brightness at the expense of clarity because of diffraction effects.

A fisheye camera, on the other hand, is a wide angle imaging device designed to capture a large FOV, typically approaching or exceeding 180°. This is achieved through a specialized lens that introduces distortion, mapping a hemispherical scene onto a flat sensor. The resulting image has a strong curvature, especially near the edges, as straight lines far away from the optical axis appear to be curved.

1.2 Purpose

The purpose of this thesis is to develop and evaluate an alternative method for the extrinsic calibration of automotive cameras, using the car body as a reference. The ultimate goal is to replace the current target-based calibration approach with a more efficient and scalable solution suitable for factory implementation.

Since Volvo Cars are equipped with both pinhole and fisheye cameras, the developed method should work for both types, requiring only minor adjustments in their respective methods.

The proposed method should be validated using images captured in a simulation environment, where the camera orientation is known and can be compared against a ground truth.

Additionally, the solution should work for real-world images captured by Volvo cars, demonstrating the potential for future implementation. The method will be evaluated and compared to the current calibration method in terms of accuracy, as well as time and cost efficiency.

1.3 Limitations and Demarcations

The end goal of the thesis is that the developed method should be applicable on all Volvo Cars models. However, this Master thesis will be limited to only focus on one specific model. This choice was made because of the great availability of simulation data for the calibration process as well as the opportunity to get real life data for that model. Since there is no communication or data synchronization between the cameras, each camera will be analyzed independently. This means that if the algorithm works on all cameras, then it will work on other car models as long as the same part of the car body is visible.

Furthermore, for different car models, there is a difference in how much of the car body that can be seen from the different cameras. To be able to do the calibration based on the car body, there is an advantage if clear features of the car body are visible in the cameras. In order to remove the current calibration method, some part of the car body must be visible to every camera on the car.

Similarly, as the goal is to improve the current method, the upcoming solution needs to be able to run faster or at the same speed as the current one. It is also important to note that the solution must also be possible to implement in all Volvo Cars factories in the future.

Extrinsic calibration involves both position and orientation, but for our method it is assumed that the difference in position for the cameras between cars is negligible and calibration in our case will only consider orientation.

2

Theory

This chapter explains the main theory behind the camera calibration method and covers important techniques in image processing, geometry, and optimization. Firstly, the section introduces the definition of the different rotation angles and the calculation of the axis-angle representation. Secondly, the section will go over the the theory for the image processing. Then it will go over all the theory necessary to project image points to 3D coordinates. Ultimately, the theory for rotation and optimization will be presented.

2.1 Definition of Roll, Pitch and Yaw

In order to have a successful extrinsic calibration, the rotation angles roll, pitch and yaw should be correctly decided. The deviation angles are defined as the rotation from the nominal orientation in the camera coordinate system. Furthermore, the roll angle will always be measured as the angle that rotates around the X-axis, which is the axis that points away from the camera's point of view. The pitch angle is defined as the angle that rotates around the Y-axis, with the Y-axis pointed to the left. The final rotation angle, yaw rotates around the Z-axis which points straight up. A visualization of this can be seen in Figure 2.1

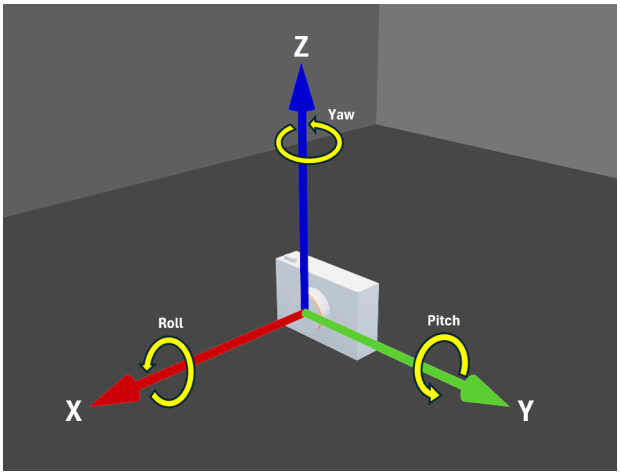


Figure 2.1: A visualization over how the rotation angles are defined around the camera coordinate system. All the arrows point in the positive direction.

2.2 Axis-Angle Representation

If one knows the rotation angles in roll, pitch and yaw, one can calculate the so called axis-angle representation. As the order in a rotation matters, it is not correct to simply add the angles together. Instead, one needs to create two rotation matrices. One is the matrix for the nominal or optimal rotation of the camera. The other matrix is the measured orientation of the camera. By taking the inverse of the nominal matrix multiplied with the rotated matrix, one gets the rotation matrix that represents the rotation from the nominal. One can then represent the deviation as a rotation vector. The rotation vector is three dimensional and is co-directional to the axis of rotation. The vector norm gives the combined rotation angle, which is the axis-angle representation [2].

In order to decide whether the camera is mounted in a correct way, Volvo Cars uses the value of the length for the axis-angle representation vector. If the vector's norm is within a specific range, the camera is mounted correctly on the car and the car can be moved on to the next step in the factory. This is one of the most important part of the factory calibration process and our method must also be able to determine the axis a with a high accuracy.

2.3 Image Segmentation and Masking

This section will go over the theory of the steps used to create a mask of the outline of the car body from the images.

2.3.1 OpenCV

Open Source Computer Vision Library (OpenCV) is a free and open source software library designed for computer vision and image processing applications. Mostly written in C++, OpenCV is cross platform and provides bindings for multiple programming languages, including Java and Python [3].

2.3.2 Gaussian Blur

Gaussian blur is an image processing technique for reducing noise and detail in an image. It is achieved by convolving the image with a Gaussian kernel, which smooths intensity variations between pixels. The transformation is defined by the two dimensional Gaussian function:

$$G(u, v) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \tag{2.1}$$

where (u, v) represents the pixel coordinates relative to the center of the kernel and σ (standard deviation) controls the degree of smoothing. Larger values of σ result in a stronger blur effect [4].

To apply the Gaussian blur, the image $I(u, v)$ is convolved with the Gaussian kernel $G(i, j)$, computed as

$$I'(u,v) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(u+i, v+j)G(i,j) \tag{2.2}$$

where k represents the kernel size. The new value of each pixel is a weighted sum of its neighboring pixels, with the weights determined by the Gaussian function. This results in a gradual transition between intensity values, reducing sharp edges and noise.

2.3.3 Canny Edge Detection

Canny edge detection is a widely used method for identifying edges in an image by detecting regions with significant intensity changes. It reduces noise and ensures that only the most relevant edges are preserved [5].

In order to achieve this, the algorithm goes through multiple steps. The first step is to smooth the image to minimize small variations and noise while containing the edges. This is done by a 5×5 kernel as explained in Section 2.3.2.

The second step is to identify areas of rapid intensity change to highlight edges. To refine the result, weaker edges are filtered out while maintaining continuous and well defined boundaries. The smoothed image is then processed using a Sobel kernel in both the horizontal and vertical directions to compute the first derivative in each respective direction. From these two resulting images, the magnitude and direction of the edge gradient for each pixel can be determined using the following formulas:

$$\mathbf{G} = \sqrt{\mathbf{G_x^2} + \mathbf{G_y^2}} \tag{2.3}$$

$$\mathbf{\Theta} = \arctan(\frac{\mathbf{G_y}}{\mathbf{G_x}}) \tag{2.4}$$

$\mathbf{G_x}$ and $\mathbf{G_y}$ are the first derivatives in the horizontal and vertical direction, respectively.

The angle $\mathbf{\Theta}$ is rounded to one of the four directions each time, either to the side, down, or to one of the two diagonals. If the angle is between $0° - 22.5°$ or between $157.5° - 180°$ the angle $\mathbf{\Theta}$ is set to $0°$ which represents the horizontal direction. If it is between $22.5° - 67.5°$ it sets $\mathbf{\Theta}$ to $45°$ which is the direction of the positive diagonal. Similarly, an angle between $67.5° - 112.5°$ maps to $90°$, which is the vertical direction and an angle between $112.5° - 157.5°$ maps to the negative diagonal direction.

After computing the magnitude and direction of the gradient, the image is scanned to eliminate unnecessary pixels that do not contribute to the edges. This is done by examining each pixel and checking whether it is a local maximum in its neighborhood along the gradient direction. In Figure 2.2 a visual representation shows how the edge is detected. Point A (which is on the edge in the vertical direction) is compared with point B and point C as they are in the gradient direction. If point A forms a local maximum between these points it goes to the next step, otherwise it is put to zero and not considered an edge.

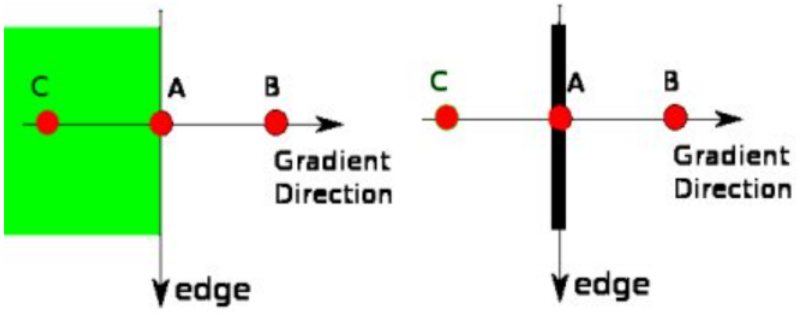


Figure 2.2: An image visualization that shows how the algorithm uses the gradient to decide if point A is on the edge. The image was taken from OpenCV [6].

The next step is to check if the edge is within the thresholds. To do this, a minimum threshold and a maximum threshold are set. This is decided by the user of the algorithm and decides how strong the edges must be to be considered an edge. Thus, it is really important that the user sets the correct threshold values depending on how many edges one wants in the image. Some threshold values will include all possible edges, while others may only include the really dominant edges in the image. In Figure 2.3 one can see how these threshold values work to decide if the edge should be kept or removed. In Figure 2.3, maxVal is the maximum threshold and minVal is the minimum threshold. Edge A is above the maximum threshold, which makes it a strong edge, and therefore it is kept. Edge C is below, but connected to edge A, so it is kept. Edge B is below and is not connected, which means it is removed from the final image. If an edge is below the minimum threshold, it is removed even if it is connected to a strong edge [6].

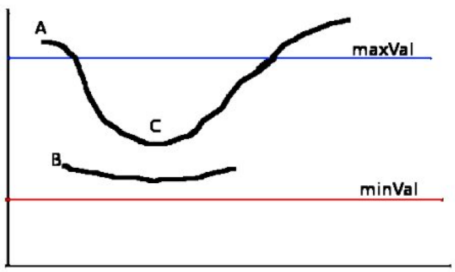


Figure 2.3: The image shows which edges that are kept based on the threshold values. The image was taken from OpenCV [6].

In summary, this method is commonly applied in object detection, image segmentation, and feature extraction, as it provides accurate edge representation while reducing noise interference. OpenCV has Canny edge detection implemented as a built-in function.

2.3.4 Dilation

OpenCV has a function that dilates the image. The function makes bright areas in an image bigger. If one has a narrow white line on a black background, then the function will fill the line and make it thicker. This is done by using a kernel with an anchor point, which is in the middle of the kernel. The kernel is then slided over the original image and for each position the maximum pixel value that the kernel overlaps with is calculated and put at the position of the anchor point. The maximum pixel value is calculated by the following formula where I' is the output image, I is the original image, u and v are the pixel coordinates of the anchor point in the image and i and j are the pixel coordinates in the kernel [7].

$$I'(u,v) = \max_{(i,j):element(i,j)\neq 0} I(u+i, v+j) \tag{2.5}$$

2.4 Image Distortion

Image distortion occurs when straight lines in a scene appear curved in the captured image. Distortion can be utilized to achieve a wider FOV than would otherwise be possible. The distortion introduced by a camera can be mathematically modeled using polynomials, allowing for correction through image processing techniques. There are two main types of distortion: radial distortion, which causes straight lines to bend outward or inward, and tangential distortion, which results from misalignment in the optical system, see Figure 2.4.

Figure 2.4: The effect of radial and tangential distortion in an image [8].

2.4.1 Radial Distortion

Radial distortion occurs when straight lines bend radially symmetrically from the center of the image. It can be classified into three types: barrel distortion, where lines curve outward; pincushion distortion, where lines bend inward toward the center; and mustache distortion, which is a combination of both.

This type of distortion is commonly modeled using a polynomial function of the radial distance r from the center of the image. A common formulation is the following.

$$r_{\text{distorted}} = r(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \tag{2.6}$$

where $k_1, k_2, k_3, \dots$ are the distortion coefficients that characterize the degree of distortion. Higher-order terms are often included for more precise modeling, especially in wide-angle and fisheye lenses [10] [11].

2.4.2 Tangential Distortion

Tangential distortion arises when the optical axis of the camera is not perfectly aligned with the image sensor, leading to a displacement of points in the image. This type of distortion is caused by slight manufacturing misalignment or lens mounting errors. Unlike radial distortion, which is symmetric, tangential distortion shifts image points asymmetrically. It can be modeled using two additional distortion coefficients, p_1 and p_2, as follows:

$$u_{\text{distorted}} = u + [2p_1 uv + p_2(r^2 + 2u^2)] \tag{2.7}$$

$$v_{\text{distorted}} = v + [p_1(r^2 + 2v^2) + 2p_2 uv] \tag{2.8}$$

where (u, v) are the original image coordinates and $r^2 = u^2 + v^2$ is the squared radial distance from the image center. By performing an intrinsic camera calibration, these distortion coefficients can be estimated and used to correct image distortions [12].

2.5 Template Matching

In order to find a template in an image or to find a small part of a larger image, OpenCV has a function called matchTemplate to do this. The function slides the template over the larger image and returns the position where the template matched the best on a pixel level accuracy.

In order to decide which position has the best match, several comparison methods can be used. An example of such a comparison method is the normalized cross-correlation coefficient. It takes an input image I and a template T of size $w \times h$. The template must be smaller than the input image in order for the function to be

able to slide the template over image I. The normalized cross-correlation method returns a value between -1 and 1 for each position, where 1 indicates that all pixels match and -1 indicates that all pixels in the template are opposite. The formula for computing this value can be seen in Equation 2.9, where $\bar{I}$ and $\bar{T}$ are the mean pixel values for the input image I and the template T, respectively. The indices u, v are the position of the template in the input image and the indices i, j are the pixel coordinates in the template image [13].

$$R(u,v) = \frac{\sum_{i=0}^{w-1}\sum_{j=0}^{h-1}\left(I(u+i,v+j)-\bar{I}\right)\left(T(i,j)-\bar{T}\right)}{\sqrt{\sum_{i=0}^{w-1}\sum_{j=0}^{h-1}\left(I(u+i,v+j)-\bar{I}\right)^2}\sqrt{\sum_{i=0}^{w-1}\sum_{j=0}^{h-1}\left(T(i,j)-\bar{T}\right)^2}} \tag{2.9}$$

2.6 Back Projection from 2D to 3D

Given a 2D point $\mathbf{x} = (u, v)$ in an image, its corresponding 3D point must lie along a ray originating from the center of the camera and passing through the image plane. This relationship can be expressed using the inverse projection equation:

$$\mathbf{X} = sK^{-1}\tilde{\mathbf{x}} \tag{2.10}$$

Here $\mathbf{X} = (X, Y, Z)$ is the 3D point in camera coordinates, s is the depth scale factor that determines the exact position along the ray, K is the intrinsic camera matrix and $\tilde{\mathbf{x}} = (u, v, 1)^\top$ is the homogeneous 2D image coordinate.

The intrinsic camera matrix K is typically defined as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

where f_x and f_y are the focal lengths in pixels and (c_x, c_y) represents the principal point of the camera.

If the depth Z of the point is known, the 3D coordinates can be explicitly recovered using:

$$X = \frac{(u - c_x)Z}{f_x}, \quad Y = \frac{(v - c_y)Z}{f_y}, \quad Z = Z. \tag{2.12}$$

By incorporating depth information, it becomes possible to accurately reconstruct the spatial position of objects from 2D image data using the equations in 2.12 [14].

2.7 Rotation of 3D Points

A 3D point $\mathbf{X} = (X, Y, Z)$ can rotate around the origin using a rotation matrix R, which preserves its distance from the origin while changing its orientation in space. The rotated point is given by:

$$\mathbf{X}' = R\mathbf{X} \tag{2.13}$$

where $\mathbf{X}' = (X', Y', Z')$ is the rotated 3D point and R is an orthonormal matrix 3×3 representing the rotation [15].

The individual rotation matrices are defined as follows:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.14}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.15}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

Here:

- ϕ: roll is rotation around the x-axis,
- θ: pitch is rotation around the y-axis,
- ψ: yaw is rotation around the z-axis.

The total rotation matrix is constructed by applying the rotations in the order roll–pitch–yaw:

$$R = R_x(\phi)R_y(\theta)R_z(\psi) \tag{2.17}$$

This ordering is the Tait-Bryan convention for extrinsic rotation. As matrix multiplication is not commutative, changing the order of rotations leads to a different final orientation [16].

2.8 Optimization With the BFGS Algorithm

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is an iterative optimization algorithm used to minimize smooth and differentiable functions. It belongs to the class of quasi-Newton methods, which approximate second-order information of the function without explicitly computing the Hessian matrix. Instead, BFGS constructs an estimate of the inverse Hessian using successive gradient evaluations.

At each iteration, the parameter vector $\mathbf{v}$ is updated using the search direction $\mathbf{p}_k$ and the step size α_k:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{p}_k. \tag{2.18}$$

The search direction is computed as

$$\mathbf{p}_k = -B_k^{-1} \nabla f(\mathbf{v}_k), \tag{2.19}$$

where B_k^{-1} is an approximation of the inverse Hessian matrix of the function $f(\mathbf{v})$.

The core of the BFGS method is the iterative update of B_k^{-1}. Given the gradient difference

$$\mathbf{y}_k = \nabla f(\mathbf{v}_{k+1}) - \nabla f(\mathbf{v}_k) \tag{2.20}$$

and the step vector;

$$\mathbf{s}_k = \mathbf{v}_{k+1} - \mathbf{v}_k, \tag{2.21}$$

The inverse Hessian approximation is updated as follows:

$$B_{k+1}^{-1} = \left(\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^\top\right) B_k^{-1} \left(\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^\top\right) + \rho_k \mathbf{s}_k \mathbf{s}_k^\top, \tag{2.22}$$

where

$$\rho_k = \frac{1}{\mathbf{y}_k^\top \mathbf{s}_k}. \tag{2.23}$$

This update ensures that the Hessian approximation remains positive definite, preserving the descent properties of the method [17].

The step size α_k is typically determined by using a line search to satisfy the Wolfe conditions, ensuring a sufficient decrease in the function value and maintaining a suitable curvature condition.

3

Methods

In order to find the extrinsic rotation angles for the camera, the algorithm follows some steps that will be described in depth below. The method is the same overall for all five cameras, but with some minor adjustments between the pinhole camera and the fisheye cameras due to the different camera types. Those adjustments will also be described below.

3.1 Acquisition of Simulated Images

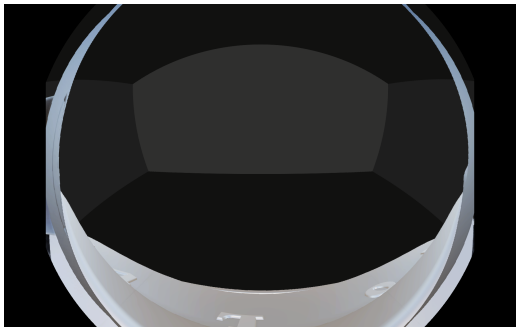
To evaluate the algorithm, a large set of images with known camera orientations was required for all cameras. This was achieved using a simulation tool capable of generating synthetic images from a detailed 3D model of the specific vehicle. The simulation environment included accurate representations of both the geometry of the car and the exterior cameras. The virtual cameras were modeled to match the optical characteristics of the real cameras, including the focal length, FOV, and lens distortion parameters.

Each virtual camera was precisely positioned and oriented according to its real world mounting location and direction on the vehicle. This ensured that the simulated setup closely mirrored the actual sensor configuration. Using the simulation tool, images were captured from each camera with controlled rotational deviations.

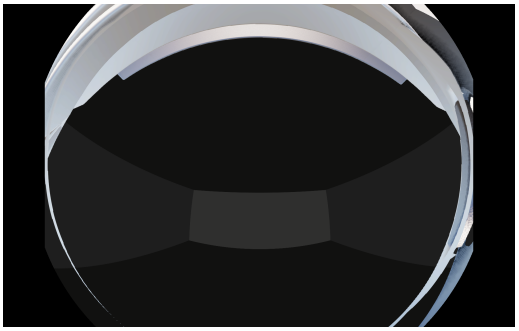
When constructing the dataset, several practical constraints were considered. First, image capture in the simulation environment had to be performed manually, which made the process time consuming. Secondly, the intended use case for the calibration is to ensure that the camera has been mounted on the car within a specific range, measured in terms of the norm of the axis-angle representation. As a result, there was no need to include larger rotational offsets in the test set.

Taking these constraints into account, the cameras were rotated around each of the three principal axes, roll, pitch, and yaw, in discrete steps of $-3°, -2°, -1°, 0°, 1°, 2°$, and $3°$. This resulted in a total of $7^3 = 343$ unique camera orientations for each of the five cameras, yielding 343 simulated images per camera in which the camera had been rotated from its nominal pose.

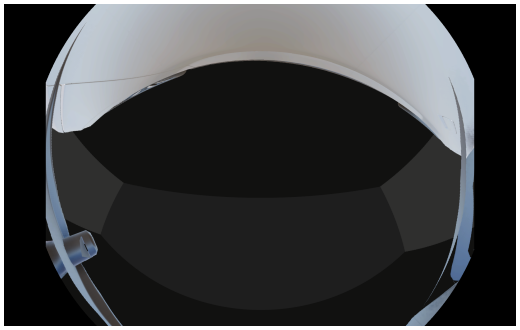
The nominal simulated images can be seen in Figures 3.1 and 3.2.



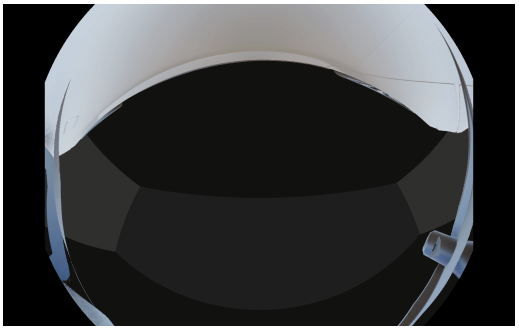
(a) The figure shows the nominal fisheye back camera image from the simulation.



(b) The figure shows the nominal fisheye front camera image from the simulation.



(c) The figure shows the nominal fisheye left camera image from the simulation.



(d) The figure shows the nominal fisheye right camera image from the simulation.

Figure 3.1: The four fisheye camera's images from the simulation tool.

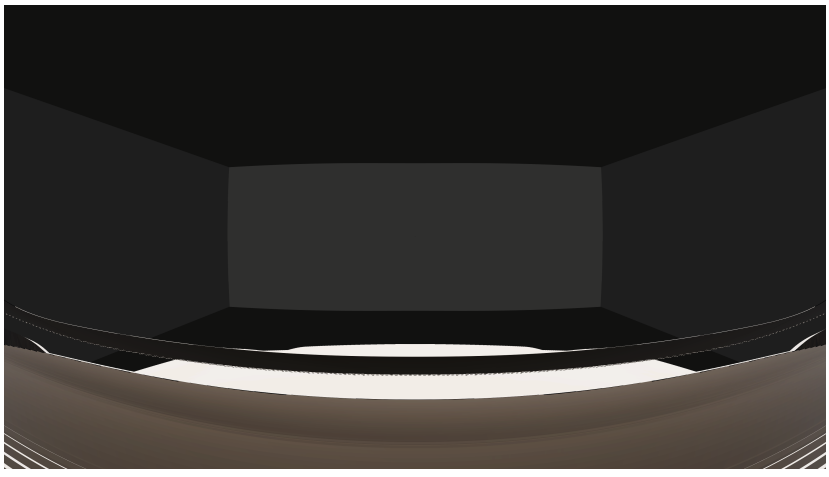


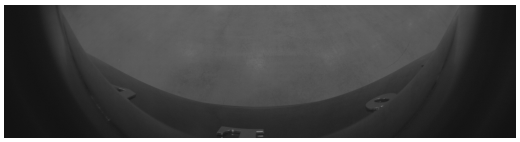
Figure 3.2: The figure shows the nominal pinhole front camera image from the simulation.

3.2 Acquisition of Real Images

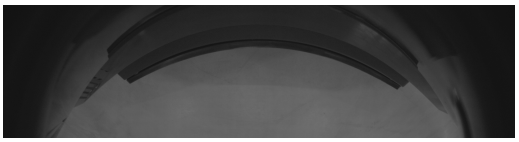
Using images taken in the room for the current factory calibration was considered, as this would have provided a large dataset of real images with known rotation deviations. However, when inspecting those images, it was discovered that a vast majority of them had reflections on the car body, which interfered with the point identification step. These reflections come from the lines and chessboard target that the current factory calibration uses. There were also some other lines and poles that interfered as they were positioned at the same place as our chosen reference points. Another reason that makes it difficult to use old images from the current factory calibration is that the algorithm needs each camera's intrinsic parameters to achieve the best possible accuracy. As these parameters are not normally necessary to save, there is no way one could find the intrinsic parameters for many of the images. This meant that such images could not be used to collect the dataset for real images.

Based on this, this project needed to find another way of collecting real images. Volvo Cars provided the opportunity to borrow a few cars of the desired model. To avoid lines or reflection interfering with the point identification, the images were taken by driving the cars indoors to a suitable spot and then taking the images. The spot should have a clean floor without any lines or other potential disturbing objects close to the car body. The images were taken indoors to represent the environment in the factory to the best of our ability. It would also be possible to run the car body calibration outside, but as the factory is indoors, it was decided to take all the images indoors. By reading out the intrinsic parameters the project could run the algorithm on the images taken and also compare them to the rotation angles from the current factory calibration method.

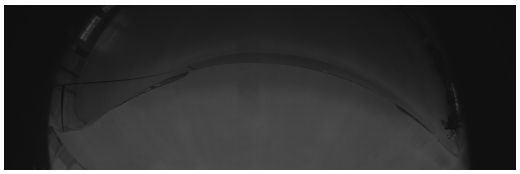
The real images taken can be seen below in Figures 3.3 and 3.4.



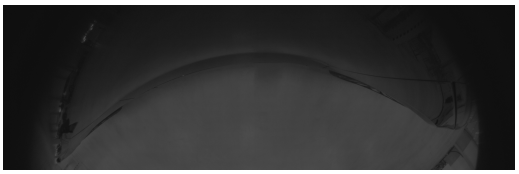
(a) The image shows the fisheye back camera with the image taken in a real car. The image has been cropped.



(b) The image shows the fisheye front camera with the image taken in a real car. The image has been cropped.



(c) The image shows the fisheye left camera with the image taken in a real car. The image has been cropped.



(d) The image shows the fisheye right camera with the image taken in a real car. The image has been cropped.

Figure 3.3: The real images taken in a car for all four fisheye cameras.

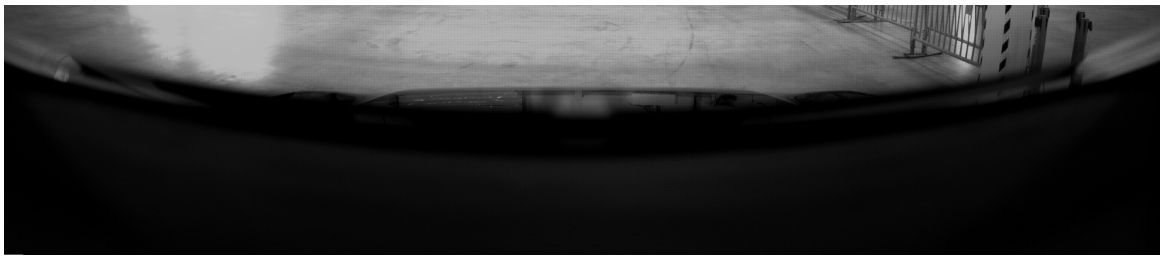


Figure 3.4: The image shows the pinhole front camera with the image taken in a real car. The image has been cropped.

As one can see in Figures 3.3 and 3.4 the images are very dark and not as clear. This is because the images are optimized for the current factory calibration station, while this project took the images in a normal indoor environment. For this thesis it was unfortunately not possible to get the original images, therefore the project needed to adjust the car body algorithm to work on the dark images. This step made the next step, the creations of masks much more difficult and time consuming for the thesis. Especially the fisheye back camera caused some problems because of the small letters which have some narrow details important for the template matching. In the future if Volvo Cars wishes to use the car body algorithm, it will be possible to use the original images, which will make the algorithm easier to use and more stable.

3.3 Point Extraction from Images

The next step was to get the 3D coordinates of the reference points from an image where the camera has been rotated. This section will go over how these coordinates were obtained.

3.3.1 Edge detection

Binary masks were created from the input images to highlight specific visual features on the car body. These masks helped make key structures more visible by emphasizing edges and filtering out irrelevant parts of the image.

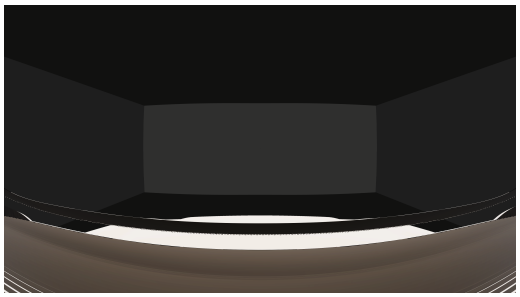
The mask creation process followed three main steps, all implemented using OpenCV:

3.3.1.1 Blur

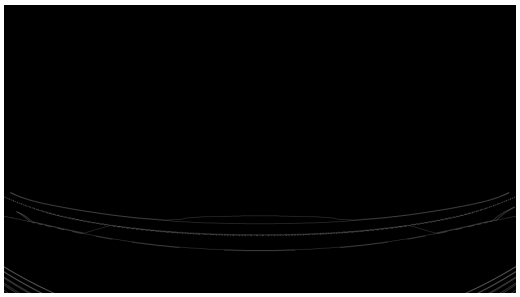
To reduce image noise and small variations, a Gaussian blur was applied to each region of interest. A kernel size of 5×5 and a standard deviation (σ) of 1.5 were used for the Gaussian filter. This preprocessing step smoothed the image, making it easier to detect the main edges in the next stage.

3.3.1.2 Canny Edge Detection

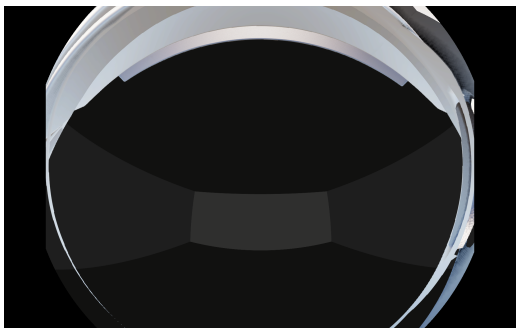
To create the masks, Canny edge detection was used to outline the edges, see Figure 3.5. This method identifies regions with strong intensity changes and produces a binary image in which edges appear as white lines. Different threshold values were tested to ensure that the desired feature was clearly defined while minimizing the inclusion of unrelated edges.



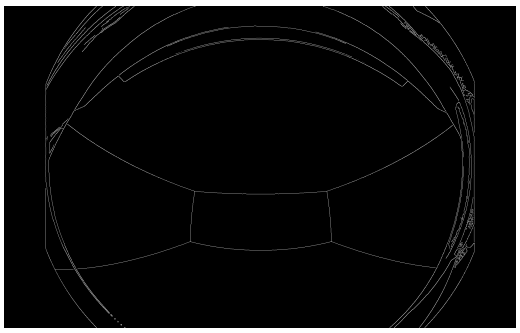
(a) Simulated image captured with the pinhole front camera.



(b) Created mask after applying canny edge detection.



(c) Simulated image captured with the fisheye front camera.

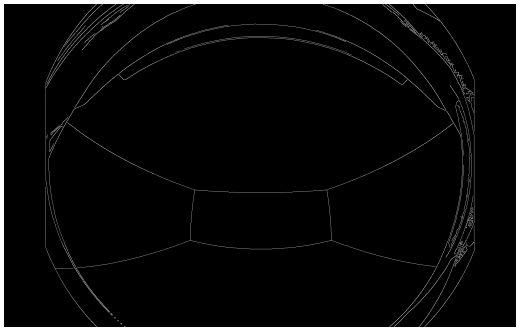


(d) Created mask after applying canny edge detection.

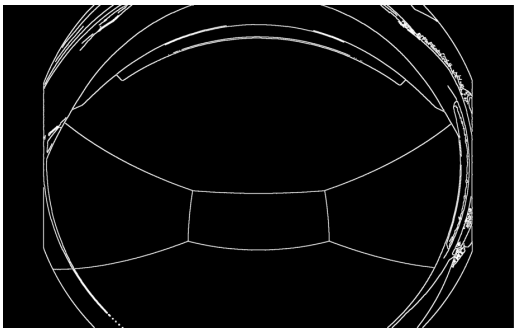
Figure 3.5: The input image taken by the pinhole front camera, (a), and fisheye front camera, (c), as well as the masks created after applying canny edge detection, (b) and (d).

3.3.1.3 Dilation

The resulting edge image was then processed with a dilation operation, which expanded the white areas and made the edges thicker. This step increased the similarity between masks from different images, even when slight variations were present due to camera rotation. Although dilation slightly reduced the sharpness of the edges, it made the method more stable by helping to ensure that the same point on the feature was detected each time. A visualization of this step can be seen in Figure 3.6 where the edges of the car body have been made thicker and possible minor gaps in the white lines have been filled in.



a) The mask for the fisheye front camera



b) The created mask for the nominal fisheye front camera with dilated edges.

Figure 3.6: The before and after the dilation step where the white lines becomes thicker.

After dilation, the final binary masks were complete. These masks, with clear outlines of the selected visual features, were then used in the next stages of the calibration process.

3.3.2 Selection of Image Points and Templates

In order to reach the best possible accuracy, one needs to be clever in selecting reference points. Testing showed that points that marked an intersection, curve or other distinct feature worked the best, as opposed to taking an arbitrary point on a straight line. The reason behind this is that the algorithm needs to find the exact same point in the rotated image. On a straight line, it is impossible to ensure that the points are in the correct position. In Figure 3.7 one can see the optimal points chosen for the fisheye cameras. For the fisheye front camera the intersection between the number plate holder and the car body was chosen. For the left and right fisheye cameras, the intersection point between the tire and the car body was chosen. Finally, for the fisheye back camera, two intersection points between the letters and the car body were chosen to act as reference points. One can also see red rectangles around the optimal points. This region of interest will work as a template and find the same shape of that region in the rotated image.

The choice of optimal points and templates is made manually and needs to be done once for every new camera. Once done, the same optimal points are always used to decide the deviated angles.

(a) Nominal fisheye back camera. Green points are the optimal points. Red rectangles are the template regions.

(b) Nominal fisheye left camera. Green points are the optimal points. Red rectangles are the template regions.

(c) Nominal fisheye right camera. Green points are the optimal points. Red rectangles are the template regions.

(d) Nominal fisheye front camera. Green points are the optimal points. Red rectangles are the template regions.

Figure 3.7: Selected optimal points and template regions across all four fisheye cameras.

In Figure 3.8 the area that was used as a template is shown along with the optimal points. As the hood is mostly a straight line, the only points that were deemed feasible were at the end of the slight slope the hood makes.

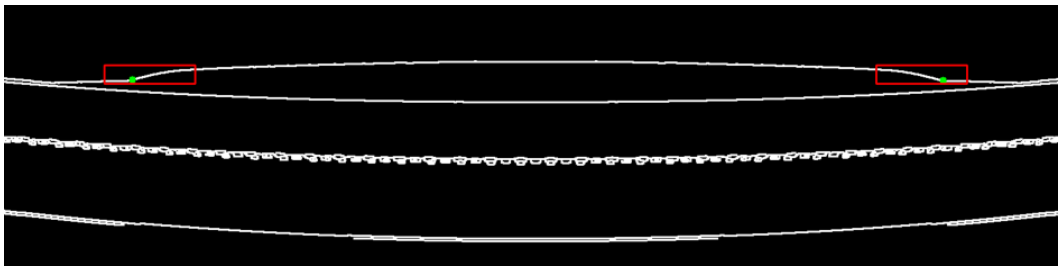


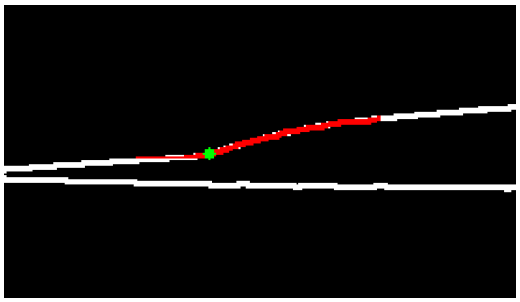
Figure 3.8: The image shows the mask of the pinhole front camera where the white line is the hood. The red rectangles shows the part that works as templates with the green points being the optimal points.

3.3.3 Identification of Image Points

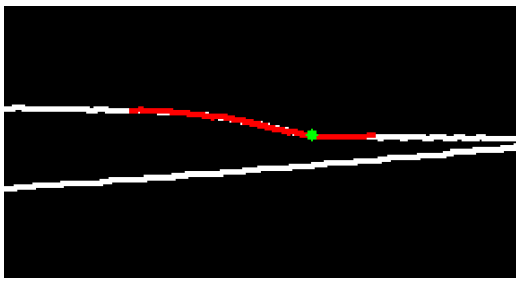
When the mask has been created for both the non-rotated and the rotated case, the next step is to find the rotated reference points on the car body.

Firstly, a so called template of the car body was cut out from the non-rotated mask. The part of the car body that is used as a template is arbitrary. In this project, different parts were tested. It was discovered that the templates that gave the highest accuracy were small templates with clear features of the car body. For example, for

the pinhole front camera the algorithm uses that the hood goes down. The templates are then laid on top of the rotated image. By translating and rotating the template, the algorithm finds the best match in which the template and the rotated car body are perfectly aligned. In Figures 3.9 and 3.10 one can see how the templates are finding the correct positions in the images.

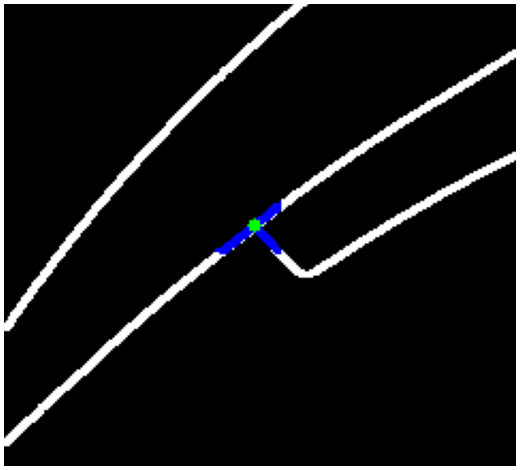


a) The alignment for the pinhole front camera of the left part of the hood.

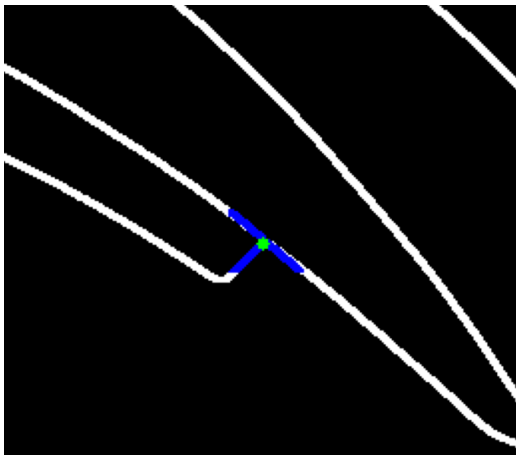


b) The alignment for the pinhole front camera for right part of the hood

Figure 3.9: The two images shows the alignment for both templates. The red line are the templates from the red rectangles and the white line is the edge of the hood for a rotated image.



a) The alignment for the left part of the number plate holder.



b) The alignment for the right part of the number plate holder.

Figure 3.10: The two images shows the alignment for both templates for the fisheye front camera. The blue line are the templates from the red rectangles and the white line is the edge of the car body and the number plate holder for a rotated image.

One might ask why the use of only some points and not all white pixels in the template? Although one might think that more points improves accuracy, this project showed that the accuracy did not change significantly. The templates used for this project have several hundred pixels and while that is small compared to

a whole image, it was decided to only use two points to improve the speed of the algorithm. As the accuracy did not improve using more points, there is no downside of using two points, but one gets a huge positive aspect when the time it takes for the algorithm to run improves.

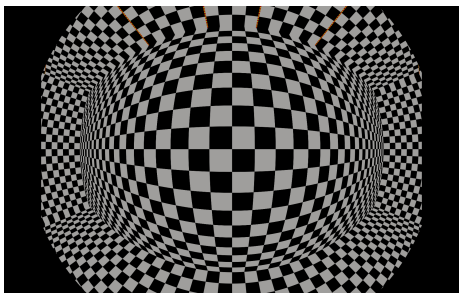
3.3.4 Undistortion of Image Points

After identifying the reference points on the car body, the corresponding pixel coordinates were undistorted. This meant repositioning the pixels to their correct locations as they would appear in an image without lens distortion.

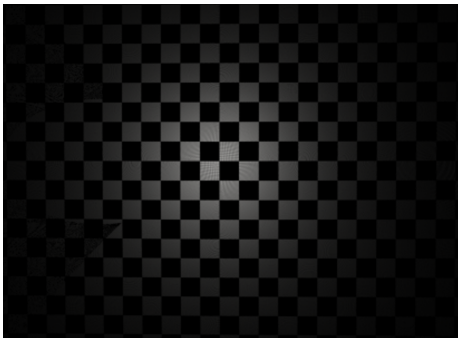
The two types of cameras used on the car introduce different levels of distortion. The pinhole camera has a slight but significant radial distortion and a small tangential distortion, whereas the fisheye camera introduces substantial radial distortion. In both cases, the radial distortion follows a barrel distortion pattern.

To correct for these distortions, the camera suppliers provide intrinsic parameters, including distortion coefficients and a distortion model specific to each camera. This model incorporates a polynomial function that describes the distortion characteristics. By applying this model, the feature points were accurately undistorted.

In Figure 3.11 one can see the undistortion of the fisheye cameras. Although the method only undistorts the points in order to make the algorithm faster, this is proof that the undistortion formula works. As we move farther away from the middle to the edges of the image, the undistorted pixels are placed farther away from each other. As the fisheye FOV is greater than 180°, the pixels toward the edge lie behind the camera. In the undistorted image one can only see those pixels by zooming in, otherwise it just looks black. Therefore, Figure 3.11 b) has been cropped to only show the undistorted chessboard pattern that one can actually see.



a) An image taken with the fisheye camera in a chessboard environment.



b) The image created by applying the undistortion function.

Figure 3.11: Before and after the undistortion function has been applied. One can see that the bent lines are now straight.

3.3.5 3D Projection of Image Points

After one has obtained the pixel coordinates for the points in the image, the next step is to project the points into 3D space. As described in section 2.6, if one knows the focal length and the distance in z, one can calculate the 3D position of the points. When the camera has an extrinsic rotation, the distance z to a point will change. The algorithm therefore calculates the rotated z-distance using the fact that the absolute distance to the point will not change when the camera rotates.

3.4 Nominal Point Extraction from CAD Model

In order to know where the nominal points chosen are in 3D coordinates, the project considered the Computer-Aided Design (CAD) model of the car. From the CAD model one could see the light rays of the cameras and from there one chose the wanted reference points. These reference points will serve as the set of nominal points. It is extremely important to choose the exact same points as in Section 3.3.2 to get the correct 3D coordinates. One now has the 3D coordinates of the optimal points in the camera coordinate system. These points will later be used to decide the rotation angles when compared to the rotated points.

3.5 Estimation of Rotation Angles

After the 3D reconstruction, two sets of points in space were obtained, the first being the rotated 3D points, as explained above. The second set is the nominal 3D points. These were obtained from the CAD model of the car where the 3D coordinates of the chosen points could be obtained. Since camera rotation does not affect the actual position of the 3D points, any difference in the positions of the two point sets must result from a camera rotation. This means that the spatial difference between the sets can be used to estimate the extrinsic rotation of the camera.

To estimate this rotation, the rotated point set was rotated in 3D using a combination of roll, pitch, and yaw rotations. These were implemented using rotation matrices as described in Section 2.7. The objective was to find the rotation angles that aligned the rotated points as closely as possible to the nominal points.

The problem was formulated as an optimization problem, in which the goal was to minimize the distance between the corresponding points in the two sets. Each pair of corresponding points, for example, the left and right reference points, was consistently matched between images. The cost function was defined as the sum of squared Euclidean distances between the rotated reference points and the target points:

$$f(\phi, \theta, \psi) = \sum_{i=1}^{n} |R(\phi, \theta, \psi)\mathbf{X}_i^{\text{ref}} - \mathbf{X}_i^{\text{rot}}|^2 \tag{3.1}$$

Here, $\mathbf{X}_i^{\text{ref}}$ and $\mathbf{X}_i^{\text{rot}}$ are the i-th points in the reference and rotated sets, respectively, and $R(\phi, \theta, \psi)$ is the combined rotation matrix as described in Equation (5) in Section 2.7.

To solve the optimization problem, the BFGS algorithm was used, as explained in Section 2.8. This method is suitable for smooth functions and uses gradient information to efficiently approximate the inverse Hessian. The optimization always started with an initial guess of 0° deviation from the nominal orientation of the camera for all three angles and was run iteratively to find the rotation that best aligned the point sets.

When the rotation of the three angles was calculated, it was used to calculate the length of the axis-angle representation vector to which it corresponded, which gave the value of the combined rotation from the nominal orientation.

4

Results and Analysis

In this section, the results are presented. Firstly, the results for the simulated images for all five cameras will be shown. Furthermore, results from images taken with real cameras and a comparison with the current calibration method will be presented. This section will also include an analysis of the results obtained.

4.1 Performance of Algorithm on Simulated Images

This section presents the performance of the algorithm across five different simulated camera configurations, evaluated on a dataset of 343 simulated images per camera. The performance is analyzed in terms of five statistical error metrics: maximum absolute error, mean absolute error (MAE), root mean square error (RMSE), standard deviation, and median error. The complete error data for the five cameras are visualized in graphs in Appendix A.

4.1.1 Error Statistics for the Car Body Algorithm

Table 4.1 presents the maximum absolute error for each camera and rotation axis. Here combined is the norm of the axis-angle representation vector. Notably, the pinhole front camera achieves the lowest maximum errors in yaw and pitch, with pitch yielding the lowest overall maximum error at 0.0676°. Meanwhile, the fisheye front camera records the lowest maximum errors in roll and combined rotation. On the other hand, the fisheye right camera exhibits the highest maximum error, at 0.279° in yaw, which is still relatively small considering the dataset includes combined rotations up to five degrees.

Table 4.1: Maximum absolute error for the five different cameras, measured in degrees.

Camera	Roll (°)	Pitch (°)	Yaw (°)	Combined (°)
Pinhole Front Camera	0.229	0.0676	0.121	0.165
Fisheye Front Camera	0.195	0.149	0.136	0.160
Fisheye Back Camera	0.236	0.185	0.262	0.237
Fisheye Right Camera	0.225	0.178	0.279	0.218
Fisheye Left Camera	0.231	0.133	0.273	0.202

Table 4.2 shows the MAE, offering insight into the average performance of the algorithm. Again, the pinhole front camera performs best, with the lowest MAE in pitch at 0.0237°, reinforcing the trend seen in maximum error. In contrast, the fisheye back camera has the highest MAE in yaw, at 0.0896°. This pattern, which consistently has lower pitch errors and higher yaw errors, is observed in all cameras, indicating a potential structural advantage in pitch estimation.

Table 4.2: Mean absolute error (MAE) for the five different cameras, measured in degrees.

Camera	Roll (°)	Pitch (°)	Yaw (°)	Combined (°)
Pinhole Front Camera	0.0758	0.0237	0.0398	0.0497
Fisheye Front Camera	0.0699	0.0448	0.0450	0.0444
Fisheye Back Camera	0.0651	0.0520	0.0896	0.0659
Fisheye Right Camera	0.0697	0.0616	0.0813	0.0555
Fisheye Left Camera	0.0676	0.0445	0.0875	0.0560

The standard deviation of the absolute error, shown in Table 4.3, closely follows the trend in MAE. Typically, the standard deviation is around 70% of the MAE, suggesting that the errors are widely spread rather than tightly clustered. This implies a broad but stable error distribution, which means that the max error is not driven by a few isolated outliers, but by a relatively wide range of values.

Table 4.3: Standard deviation of the absolute error for the five different cameras, measured in degrees.

Camera	Roll (°)	Pitch (°)	Yaw (°)	Combined (°)
Pinhole Front Camera	0.0525	0.0167	0.0273	0.0338
Fisheye Front Camera	0.0457	0.0306	0.0319	0.0319
Fisheye Back Camera	0.0466	0.0364	0.0559	0.0493
Fisheye Right Camera	0.0503	0.0372	0.0565	0.0410
Fisheye Left Camera	0.0504	0.0321	0.0601	0.0424

4.1.2 Outlier Influence

Table 4.4 presents the RMSE, which penalizes larger errors more heavily than the MAE. The relative difference between RMSE and MAE provides insight into the presence and size of outliers. Across all camera configurations, the RMSE is consistently 15-25% higher than MAE, indicating that although some outliers exist, they are not dramatically worse than the average case.

The greatest increase is observed for combined rotation in the fisheye back camera, where RMSE is 25% greater than MAE (0.0822° compared to 0.0659°). The smallest increase is for the fisheye right camera (17% for pitch), reinforcing the idea that the error distribution is generally stable across the dataset.

Table 4.4: Root mean square error (RMSE) for the five different cameras, measured in degrees.

Camera	Roll (°)	Pitch (°)	Yaw (°)	Combined (°)
Pinhole Front Camera	0.0921	0.0290	0.0483	0.0600
Fisheye Front Camera	0.0835	0.0543	0.0551	0.0546
Fisheye Back Camera	0.0800	0.0635	0.106	0.0822
Fisheye Right Camera	0.0860	0.0719	0.0989	0.0689
Fisheye Left Camera	0.0842	0.0549	0.106	0.0702

The stable ratio between RMSE and MAE, in combination with a wide standard deviation and moderate maximum errors, suggests that the algorithm does not have catastrophic failure cases and performs predictably across the test set.

4.1.3 Error Skewness

Table 4.5 shows the median error values for each camera and rotation axis. The median helps to reveal whether there is any consistent bias in the algorithm's predictions. One clear trend is that the median error for roll and yaw is negative across all cameras. This suggests that the algorithm tends to overestimate these angles slightly, since a negative signed error means the predicted angle is larger than the true one.

Table 4.5: Median error for the five different cameras, measured in degrees.

Camera	Roll (°)	Pitch (°)	Yaw (°)	Combined (°)
Pinhole Front Camera	-0.0455	0.00113	-0.00131	0.0162
Fisheye Front Camera	-0.0236	-0.0254	-0.0149	-0.00736
Fisheye Back Camera	-0.0256	-0.00440	-0.0815	-0.0421
Fisheye Right Camera	-0.0221	0.0585	-0.0631	-0.0180
Fisheye Left Camera	-0.0180	0.0157	-0.0747	-0.00814

In several cases, the error distributions are not symmetric. One way to see this is when the absolute value of the median error is greater than the standard deviation, which indicates a skewed distribution. This happens, for example, in the yaw errors for the back, left, and right fisheye cameras, and also in the roll error for the right fisheye camera.

The most skewed case is found in the pitch error of the right fisheye camera, illustrated in Figure 4.1. The error values for all 343 rotated images are sorted and plotted, and a trendline is fitted. The line clearly shows that the distribution is skewed and that most errors are positive. To center the data more symmetrically, a constant offset of 0.0636° could be subtracted from each value.

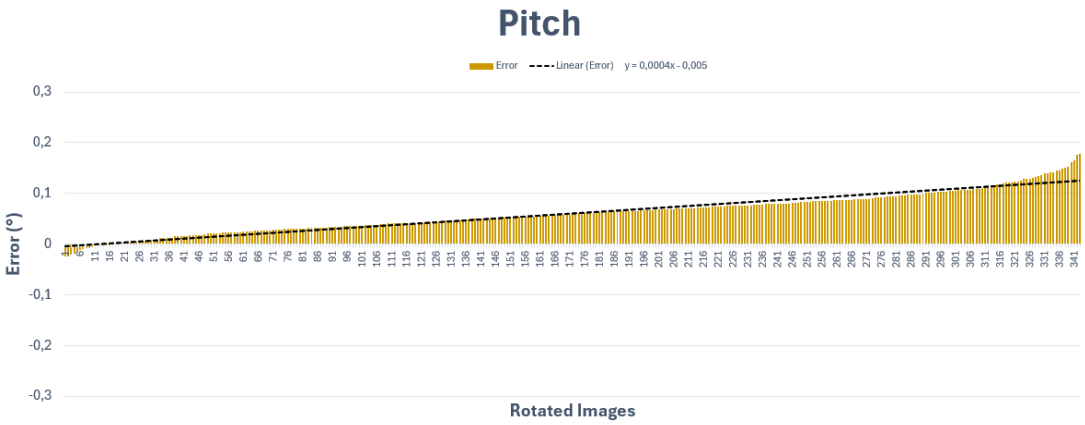


Figure 4.1: Pitch error for 343 rotated images from the fisheye right camera. Errors are sorted in increasing order. The fitted line reveals a strong positive skew in the distribution.

For comparison, Figure 4.2 shows the pitch error of the pinhole front camera, which had one of the lowest median values. The distribution here is much more balanced, though still slightly skewed. In this case, a very small offset of 0.00505° would be enough to make the distribution symmetric.

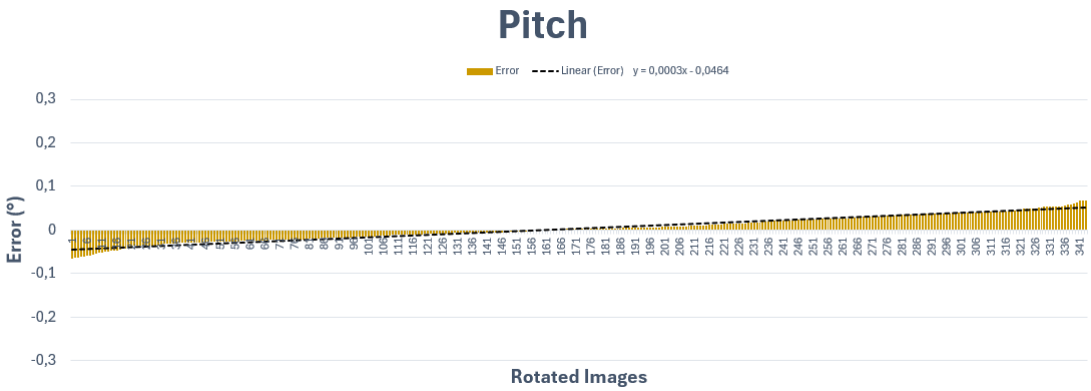


Figure 4.2: Pitch error for 343 rotated images from the pinhole front camera. Errors are sorted in increasing order. The trendline shows only a slight skew compared to the fisheye right camera.

To test whether correcting for skew improves performance, a simple post-processing step was applied where the calculated bias (i.e., the offset needed to center the distribution) was subtracted from each prediction. This had a noticeable effect: for the right fisheye camera, the MAE decreased from 0.0616° to 0.0313°. For the pinhole front camera, the correction had almost no effect, with the MAE changing from 0.0237° to 0.0239°.

This suggests that adding a bias correction step could improve the algorithm's accuracy, especially in cases with strong skew. With a larger dataset, the bias for each case could be estimated more precisely, allowing for even better corrections and potentially further reducing the error.

4.1.4 Sorting Dependencies

Each image in the dataset is created by rotating the camera around the roll, pitch, and yaw axes. For each axis, the rotation varies independently from $-3°$ to $+3°$ in steps of $1°$, resulting in 7 discrete values per axis. Since all combinations are evaluated, this results in $7^3 = 343$ unique image orientations per camera.

Although each image corresponds to a unique triplet of rotation values, (roll, pitch, yaw), the order in which these triplets are processed or visualized can vary. Specifically, the images can be sorted according to the values of roll, pitch, and yaw in any of six possible axis orders.

This sorting order affects how the data appear when plotted, especially when errors

are visualized along the x-axis according to the image index. For example, sorting in the order roll, pitch, yaw means:

- Roll changes slowest.
- Pitch changes faster.
- Yaw changes fastest.

The sequence begins with $(\text{roll}, \text{pitch}, \text{yaw}) = (-3°, -3°, -3°)$ and proceeds with the yaw angle increasing first, then pitch, and finally roll. This is conceptually similar to how nested loops work in programming, where the last variable changes fastest.

This sorting becomes relevant when analyzing how the algorithm's prediction errors are distributed across the dataset. In some cases, specific combinations of the sorting order and error axis reveal systematic trends, where the error is not randomly distributed but instead varies consistently with one of the rotation parameters.

A clear example of such a dependency is observed for the roll error of the pinhole front camera when the images are sorted by roll, pitch, yaw. As shown in Figure 4.3, the error follows a strongly linear trend with a coefficient of determination $R^2 = 0.8499$. The non-horizontal trendline suggests that the roll error is systematically influenced by the yaw value, which is the last and therefore fastest changing variable in this sorting order.

Since yaw ranges from $-3°$ to $+3°$, and is responsible for the horizontal movement throughout the graph, the tilt of the trendline indicates that a correction term proportional to the yaw value could reduce this bias and flatten the trendline.

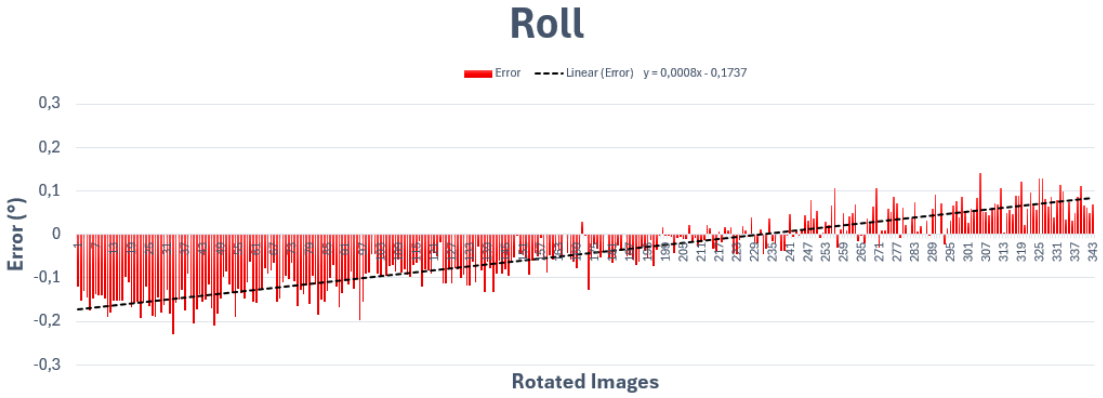


Figure 4.3: Roll error for 343 rotated images captured with the pinhole front camera. The images are sorted by roll, pitch, then yaw. The tilted trendline indicates a dependency on yaw, with $R^2 = 0.8499$.

Another similar trend is seen in the pitch error for the same camera when the images are sorted by roll, yaw, pitch. As seen in Figure 4.4, the error again follows a clear linear pattern, with $R^2 = 0.814$. Here, the tilt in the trendline suggests a dependency on the pitch value, again the last sorted (and thus fastest changing) variable in this configuration. As before, introducing a correction term that accounts for pitch could potentially eliminate this systematic bias.

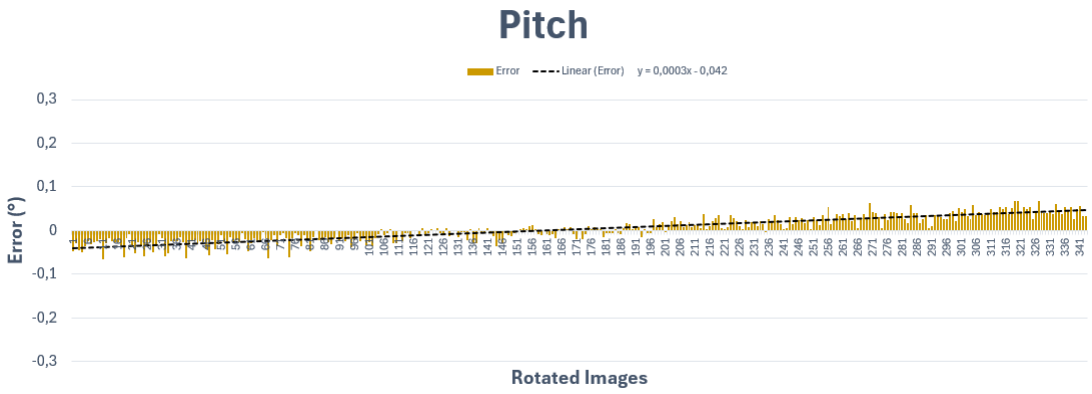


Figure 4.4: Pitch error for 343 rotated images captured with the pinhole front camera. The images are sorted by roll, yaw, then pitch. The trendline indicates a dependency on pitch, with $R^2 = 0.814$.

These findings show that, for certain camera angle configurations, the prediction error is not entirely random, but rather exhibits a structural dependency on one of the rotation parameters, depending on how the data are sorted. This suggests that the model's predictions could be improved by incorporating a post-processing step that corrects for these systematic trends, effectively flattening the error curves by compensating for the dependent variable.

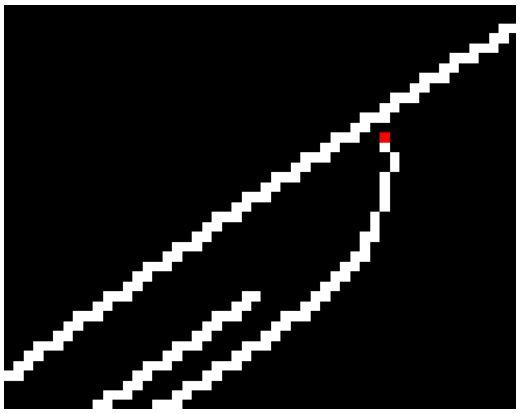
Such corrections could further reduce the MAE and improve robustness. Future work could explore this correction step more rigorously by fitting and validating such adjustments on a larger dataset.

4.1.5 Template Matching Misalignment

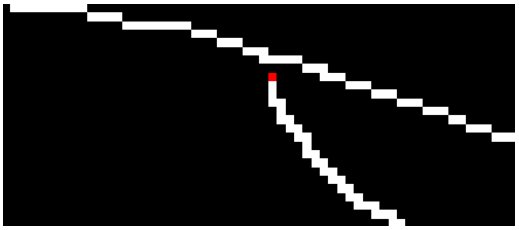
Upon investigating the causes behind the largest observed errors, one specific flaw in the algorithm was identified. The template matching step does not always locate the exact intended reference point. Instead, it sometimes selects a nearby pixel, typically just a few pixels away, which can result in significant rotation errors.

A clear example of this can be seen in the image that yielded the highest absolute error in the entire dataset, listed in Table 4.1. This was the yaw error for the right fisheye camera for the image rotated at $(\text{roll}, \text{pitch}, \text{yaw}) = (3°, 0°, 3°)$. The predicted values in this case were $(3.225°, -0.0937°, 3.279°)$, showing a large deviation from the ground truth.

Figure 4.5 illustrates the intended reference points for the right fisheye camera, visualized in red on both the left and right mask templates. These are the pixel locations the algorithm is expected to match.



a) The left template and optimal point.

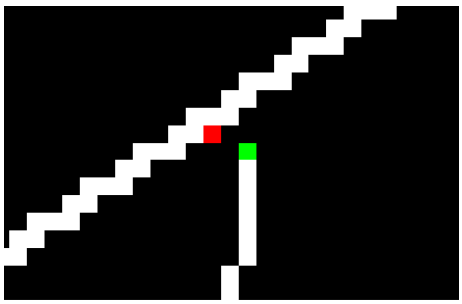


b) The right template and the optimal point.

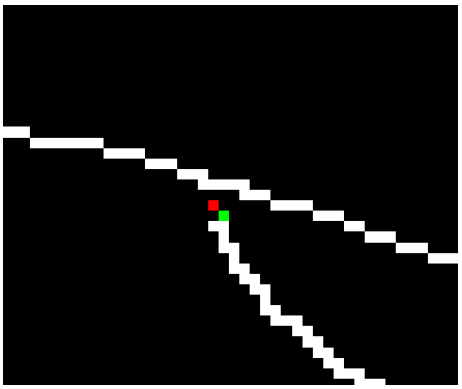
Figure 4.5: The templates used for the right fisheye camera. The intended reference points are marked in red.

Figure 4.6 shows a zoomed-in view of the actual points chosen by the algorithm (in red) and the manually corrected positions (in green). In this particular example, the selected points are offset by just 1–2 pixels from the intended targets.

When the template matching was manually corrected to use the intended green points instead, the output changed to $(\text{roll}, \text{pitch}, \text{yaw}) = (3.075°, 0.0855°, 3.0617°)$ a considerable improvement in all three angles.



a) Left mask with the algorithm's chosen point (red) and corrected point (green).



b) Right mask with the algorithm's chosen point (red) and corrected point (green).

Figure 4.6: Zoomed-in masks of the right fisheye camera showing misalignment between the chosen (red) and intended (green) reference points.

Although this misalignment was only shown here in one example, it highlights a potential source of systematic error. Some more examples with high error were tested and showed the same results. If the template matching consistently selects a pixel slightly to one side of the correct reference point, this could introduce a directional bias across many images. This may partly explain the consistent patterns seen in Section 4.1.3 and Section 4.1.4.

To determine whether this misalignment is a widespread problem, more analysis is needed. This would involve checking both low-error and high-error cases across all cameras to see whether the offset from the intended reference point correlates with the prediction error.

If such misalignments are a significant contributor to observed errors, two possible improvements could be explored. The first is to refine the template matching to increase precision. The second is to add a post-processing step to adjust or validate the selected reference point.

However, the current implementation, which typically deviates by only a few pixels in the worst case, performs reasonably well overall and represents a solid starting point. Future refinements could further reduce the error by addressing this problem.

4.2 Performance of Algorithm on Real Images

In Figure 4.7 one can see the calibration results of both the current factory calibration method as well as the car body algorithm developed in this thesis. Although some rotation angles are close, some angles are significantly different between the two methods. As there does not exist any accurate method to determine the true camera rotation, one can not say which method has the correct values.

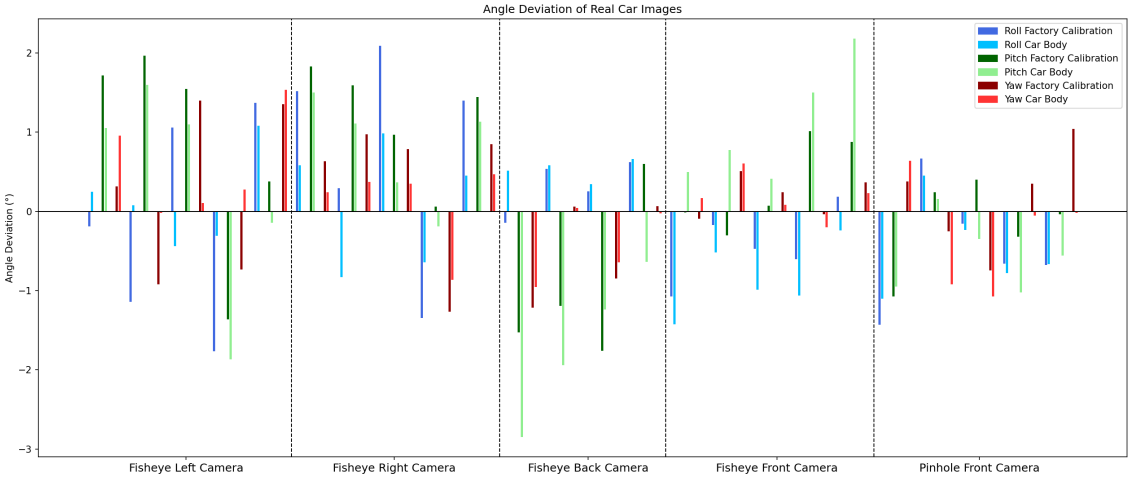


Figure 4.7: The calibration results for both the car body method and the current factory calibration method in roll, pitch and yaw. Five different cars of the same model were used. Note that the fisheye back camera has one less result due to the fact that one car did not have a valid factory calibration for that camera.

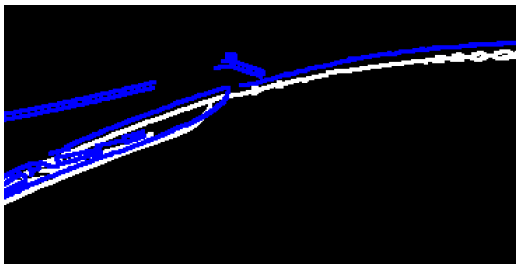
If one studies the result for the fisheye front camera in Figure 4.7 one can see a pattern. The roll angles for the car body method are for all cars around $-0.4°$ compared to the factory calibration. The pitch angles also show such a pattern with the angles being around $0.7°$ larger than the factory calibration. More testing would be needed to confirm that this is the case for other cars, but it is an interesting pattern that one could study more. However, it is impossible to determine which method is wrong, as there is no way to determine the actual camera orientation.

4.2.1 Validation of the Calibration Results for Real Images

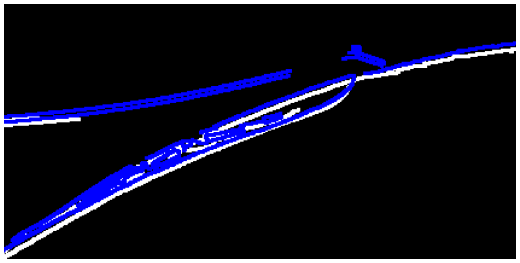
For simulated images, one knows how the camera is rotated and can compare that to the algorithm's result. However, for images taken in real life that is not possible. It is impossible to know how accurate one's calibration method is in real life. In Figure 4.7 the results for the car body calibration were compared to the current calibration method. The current factory calibration method has an error below $0.1°$ per rotation angle and the car body calibration method has a maximum absolute error below $0.3°$ for all cameras. When the error of both methods are added together, there is quite a huge span the actual rotation could be. This is of course not desired

as one wants to know how accurate the algorithm is in real life and a high accuracy of the calibration method is crucial.

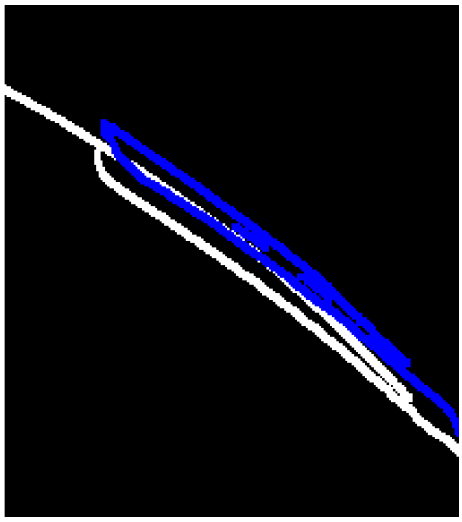
By overlaying a simulated image where the rotation of the camera was the same as the results from the calibration method, one could see if the edges overlapped. This can be seen in Figure 4.8, while both overlays show an unwanted difference, the car body algorithm gives a closer alignment than the factory calibration. Although this is not a complete proof, it gives an indication of what the true rotation angles are.



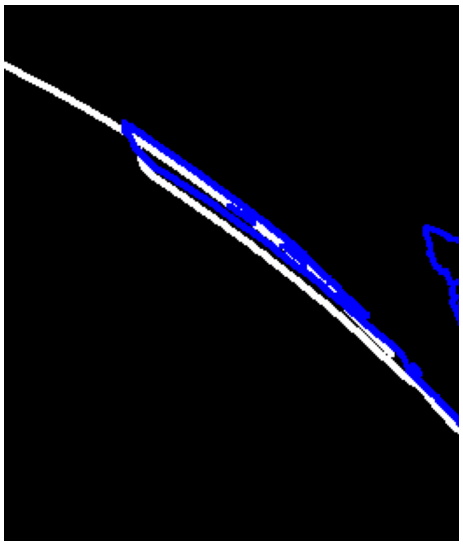
(a) An overlay of the left wheel comparing a simulated image with the factory calibration angles to a real image.



(b) An overlay of the left wheel comparing a simulated image with the car body angles to a real image.



(c) An overlay of the right wheel comparing a simulated image with the factory calibration angles to a real image.



(d) An overlay of the right wheel comparing a simulated image with the car body angles to a real image.

Figure 4.8: An overlay for the fisheye left camera. The blue line are the mask from the real image with the white line being the mask for the simulated image. The simulated image has been rotated accordingly to the result for the current factory calibration (a) and c))and for the car body algorithm (b) and d)).

4.2.2 The Dependency on Intrinsic Parameters

One potential source of error are the intrinsic parameters that are given. As each real camera will have different principal point and distortion coefficients, these are taken as input to the algorithm. If those values are slightly incorrect, they will have an effect on the final result. Testing was carried out about how much the parameters affected the result. However, the distortion coefficients showed to not have a great effect on the final result. All of the cameras used in this project had similar distortion coefficients and the small difference between the constants is not enough to significantly impact the final result. However, the principal point had a larger effect, but only for the fisheye cameras, since the pinhole camera does not have a huge distortion. The principal point decides the distance to the reference point which will affect the undistortion step as explained in Section 2.4. As distortion has a huge impact on the fisheye cameras, it is expected that the final result will be impacted. As discussed in Section 4.1.5, a few pixels make a significant difference in the final result, which means that the wrong principal point could potentially cause an error in the algorithm.

4.2.3 Repeatability Error

To have a stable and reliable calibration method, it is important that the same car gets the same rotation angles if the calibration is done twice by repeating the process. For the current factory calibration method, this works well for the same station. If one drives the car into the calibration station, starts the method and then drive out and repeat the process, the method will give the same rotation angles. This is the same for the car body calibration method. As the parts on the car body do not move, the binary masks will be the same and the calibration give the same rotation angles. The car body algorithm has one more advantage, namely that the calibration can be run in different places but still yield the same results. The only thing one might need to change is the threshold values for the canny edge detection, but this step is of course only necessary if there is a difference in lighting. The fact that the car body method has a very high repeatability shows the strength of the algorithm and proves its robustness and stability.

4.2.4 Impact of Manufacturing Variations on Accuracy

In an ideal world, every part of the car body is mounted in the exact same position, but in reality this is not the case. As Volvo Cars are producing thousands of cars every day, one has to assume that some parts will be mounted marginally different. Although this difference does not affect the customer, our algorithm needs pixel precision to deliver the best possible accuracy. In Table 4.6 one can see how small mounting errors would affect the car body calibration. These values are estimates of how much mounting error a real car could have. Of the fisheye cameras, only the fisheye front camera was tested, as the other fisheye cameras are expected to give approximately the same values.

Table 4.6: The error difference for the nominal image when there is a mounting error. All of the data in this table was obtained in the simulation tool by moving the objects.

Camera	Horizontal Offset (mm)	Vertical Offset (mm)	Roll (°)	Pitch (°)	Yaw (°)
Fisheye Front Camera	0	0	0.0577	0.0481	0.0134
Fisheye Front Camera	-1	-1	0.0581	0.0415	0.0176
Fisheye Front Camera	-1	1	-0.2129	0.0441	-0.0743
Fisheye Front Camera	1	-1	0.3285	0.0516	0.1002
Fisheye Front Camera	1	1	-0.0173	-0.0099	-0.0571
Pinhole Front Camera	0	0	0.0018	0.0027	0.0012
Pinhole Front Camera	-1	-5	0.0021	-0.1500	-0.0303
Pinhole Front Camera	-1	5	0.0437	0.1556	0.1032
Pinhole Front Camera	1	-5	-0.0021	-0.1500	0.0211
Pinhole Front Camera	1	5	-0.0415	0.1556	-0.1054

As one can see, the possible error that steams from mounting error of the car is not huge. For the pinhole front camera the pitch angle has an error of around $\pm 0.15°$ when the hood has a mounting error of 0.5 mm.

4.3 Optimal Point Selection for Minimizing the Error

In theory, it does not matter which optimal points are chosen as long as the rotated points are placed on the same place on the car body for the rotated image. However, in practice it is impossible to detect an exact point on a straight line. There needs to be some kind of corner, intersection point, or other distinct feature in order for the algorithm to detect the exact same point every time. As mentioned in section 4.1.5, only a few pixels of misalignment can cause an error, meaning that it is of highest importance to make the identification of the point precise to avoid any errors.

This project tested several different locations of the points for the different cameras. Although in theory it should not matter, the algorithm seems to have a higher accuracy by placing a point both to the left and to the right of the image center. This project also tried using more points than two for each image, but that did not yield a better result. There is a finite number of clear and distinct feature points for each image and there is therefore a greater risk of not having an exact identification of the rotated points if one chooses too many points. Therefore, the best results in this project were obtained by using only two points.

The project also experimented with different template sizes. Although a bigger template makes the algorithm more stable, it has a negative impact on the accuracy.

With the dilation step, the algorithm is already stable for all realistic rotation angles. A smaller template increases the accuracy as the matching should be focused around the chosen feature point. A larger template matches too much with other parts of the car body. A good rule of thumb is to have as small a template as possible while still keeping the algorithm stable.

As one can see in section 4.1.1 the fisheye front camera had the lowest error out of the four fisheye cameras. This is expected since that camera had the best feature point with the intersection between the car body and the number plate holder. One can also see from section 4.1.1 that the pinhole front camera had a very low value for maximum absolute error in pitch. This is most likely due to the fact that the template is almost a straight line, which makes the pitch angle easy to determine. On the other hand, roll and yaw angles become more difficult to determine because the template does not have a clear intersection or corner point.

If possible, it is also better to have points close to the center. This is because the distortion is lower in the middle, which means that the differences between neighboring pixels are not huge after they are undistorted. However, in the edges of the image, only one pixel difference can have a hugely different location after the undistortion step. If the identification of the rotated point is wrong with one or two pixels, it is better to have it in the middle to avoid a larger error. Of course, the car body will always be in the edges of the image, but the closer to the middle the reference points are, the better.

To sum it all up, the algorithm works the best when a specific corner or intersection point can be used and a point should be placed on both sides of the image center in order for the algorithm to have the highest possible accuracy.

4.4 Future Work

This section will discuss different improvements of the car body algorithm that one would find interesting to investigate.

4.4.1 Improvement of Identification for Rotated Points

As discussed in section 4.1.5, the placement of the rotated points is essential as well as the matching of the template. In order to further improve the accuracy of the algorithm, one could improve the current method or use other computer vision techniques to find the feature point. If one is able to find the correct point with pixel accuracy every time, no matter the orientation of the camera, then the car body algorithm will most likely deliver the same accuracy as the current factory calibration method for all possible rotation angles.

4.4.2 Using a Neural Network

One idea that this project decided not to use was to use a neural network to decide the rotation angles. A neural network could be trained to decide the extrinsic rotation of the camera. The drawback is the huge dataset that is necessary in order to train the neural network. For a new car model, there does not exist any such dataset that one could collect. One possible solution could be to use the simulation tool that this project used in order to create images of all possible angles that the camera can be orientated. In order for this method to work with high accuracy, it would require that the simulation model represents the reality without any offset or errors. For now, one cannot automatically take several images for different angles, which means that one would need to create their dataset manually. Of course, in the future one has the possibility of implementing a function which can create the necessary dataset automatically. An idea would be to use a neural network to identify the reference points on the car body. A neural network could potentially detect the feature point with a higher accuracy than the template matching. However, the drawback of needing to create a huge dataset made this thesis choose another way, but this would be an interesting approach as the neural network will have to use the car body to decide the cameras orientation.

5

Conclusion

This thesis has successfully developed a calibration method using the car body to find reference points. The method works well for all cameras on the car model that was tested, both the pinhole camera and all fisheye cameras work with the same method. For the pinhole camera, an accuracy within 0.25° was reached for all angles. For the fisheye cameras an accuracy within 0.3° was obtained for all angles, but a difference between the different cameras was observed. The fisheye cameras are believed to have a lower accuracy as a result of their high distortion in the edges of the image and because they have a lower resolution. Testing showed that much of the magnitude of the error was due to the misalignment of the template. Although the misalignment was small, it is believed to have created much of the observed error. If the identification of the reference points could be improved in the future, it is believed that the car body algorithm could reach a maximum absolute error of below 0.1° for all cameras. The problem in the template matching could also explain the error skewness which was a phenomenon in some cameras. Despite this, the algorithm was able to successfully determine the orientation with a high accuracy.

The car body algorithm was also tested on five different real cars of the same model. The rotation angles determined by the car body algorithm were compared to the factory calibration method. In some cases both methods aligned, while for some angles there was a significant difference. To try to determine which method was correct, a real image was overlayed with a simulated image rotated according to the rotation angles for each respective calibration method. Here, the car body algorithm was much closer to aligning than the factory calibration. Therefore, while the results provide valuable insight into the behavior of both methods, the absence of a ground truth for the camera orientation means that one cannot declare one method superior with absolute certainty.

The current factory calibration uses optimal targets in the middle of the image, but despite using the car body this method was able to reach almost the same accuracy. Many of the drawbacks with the current factory calibration method have been eliminated with the car body calibration. For example, the car body method is much faster as well as being cheap and easy to maintain. If Volvo Cars in the future wants to implement new cameras on the car, the current factory calibration method

would need the station to be rebuilt in all factories around the world, but the car body algorithm can easily implement new cameras as well as new car models. It only requires that some part of the car body with two clear feature points are visible in the camera image.

Bibliography

[1] "Camera Calibration." OpenCV. [Online]. Available: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (accessed: May 23, 2025).

[2] Peter Corke. "Angle-axis representation of rotation in 3D." Robot Acedamy. [Online Video]. Available: https://robotacademy.net.au/lesson/angle-axis-representation-of-rotation-in-3d (accessed: May 23, 2025).

[3] "About." OpenCV. [Online]. Available: https://opencv.org/about/ (accessed: May 23, 2025).

[4] Nwe Zin Oo. "The Improvement of 1D Gaussian Blur Filter using AVX and OpenMP." 2022 22nd International Conference on Control, Automation and Systems (ICCAS). [Online]. Available: https://doi.org/10.23919/ICCAS55662.2022.10003739 (accessed: May 23, 2025).

[5] Robert Fisher, Simon Perkins, Ashley Walker and Erik Wolfart. "Canny Edge Detector." The University of Edinburgh. [Online Video]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm (accessed: May 23, 2025).

[6] "Canny Edge Detection." OpenCV. [Online]. Available: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (accessed: May 23, 2025).

[7] "Morphological Transformations." OpenCV. [Online]. Available: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html (accessed: May 23, 2025).

[8] “CalibrateCamera() in OpenCV in Python,” GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/calibratecamera-opencv-in-python/ (accessed: May 26, 2025).

[9] “Distortion (optics),” Wikipedia, The Free Encyclopedia. [Online]. Available:

https://en.wikipedia.org/wiki/Distortion_(optics) (accessed: May 26, 2025).

[10] Minjung Lee, Hyungtae Kim, and Joonki Paik. "Correction of Barrel Distortion in Fisheye Lens Images Using Image-Based Estimation of Distortion Parameters." IEEE Access, vol. 7, pp. 45723–45733, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2908451 (accessed: May 23, 2025).

[11] J. Kannala and S. S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 8, pp. 1335–1340, 2006. [Online]. Available: https://doi.org/10.1109/TPAMI.2006.153 (accessed: May 23, 2025).

[12] Jiachuan Yu, Zhisheng Zhang, Han Sun, Zhijie Xia, and Haiying Wen. "Reevaluating the Underlying Radial Symmetry Assumption of Camera Distortion." IEEE Transactions on Instrumentation and Measurement, vol. 73, pp. 1–10, 2024. [Online]. Available: https://doi.org/10.1109/TIM.2024.3451594 (accessed: May 23, 2025).

[13] "Template Matching." OpenCV. [Online]. Available: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html (accessed: May 23, 2025).

[14] Gaojian Ding, Tianfei Chen, Lijun Sun, and Pengxiang Fan. "High Precision Camera Calibration Method Based on Full Camera Model." 2024 36th Chinese Control and Decision Conference (CCDC), pp. 4218–4223, 2024. [Online]. Available: https://doi.org/10.1109/CCDC62350.2024.10588332 (accessed: May 23, 2025).

[15] Grzegorz Stepien, Jozef Sanecki, Andrzej Klewski, and Ewa Zalas. "Method of Parameter Reduction in the Transformation of Oblique Photographs and Proposal of Its Implementation in Unmanned Aerial Systems." 2016 Baltic Geodetic Congress (BGC Geomatics), pp. 171–175, 2016. [Online]. Available: https://doi.org/10.1109/BGC.Geomatics.2016.38 (accessed: May 23, 2025).

[16] Indriarto Yuniantoro, Ridwan Gunawan, and Rudy Setiabudy. "The pqr-coordinate in the mapping matrices model of Kim-Akagi on power transformation based on Euler angle rotation method." 2013 International Conference on QiR, pp. 121–126, 2013. [Online]. Available: https://doi.org/10.1109/QiR.2013.6632549 (accessed: May 23, 2025).

[17] Adrian Lam. "BFGS in a Nutshell: An Introduction to Quasi-Newton Methods." towards data science, Nov. 26, 2020. [Online]. Available: https://towardsdatascience.com/bfgs-in-a-nutshell-an-introduction-to-quasi-newton-methods-21b0e13ee504/ (accessed: May 23, 2025).

A

Algorithm Error

The algorithm error, meaning the measured rotation subtracted to the actual rotation, visualized for all cameras with the rotated images sorted in the order roll, pitch, yaw as described in Section 4.1.4.

A.1 Error Pinhole Front Camera

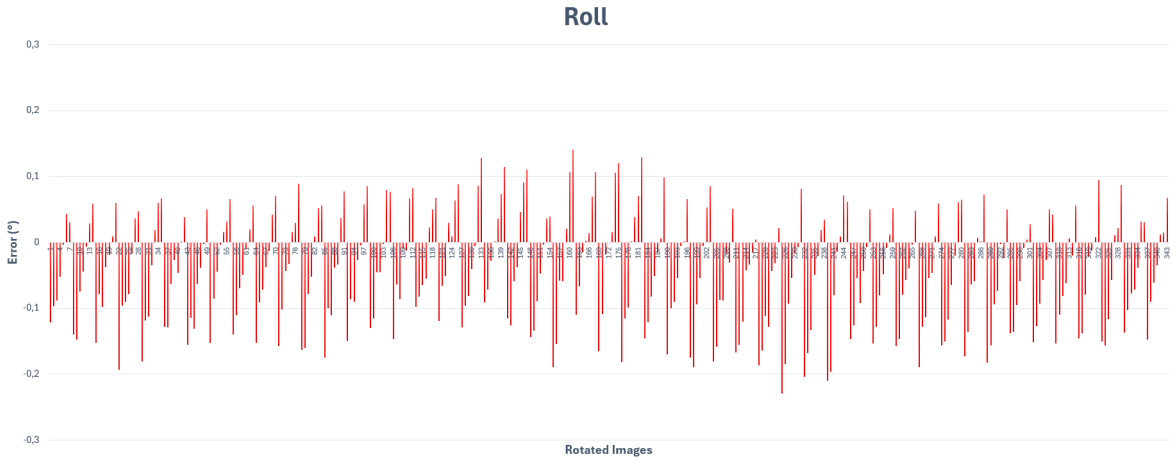


Figure A.1: Roll error for 343 rotated images captured with the pinhole front camera.

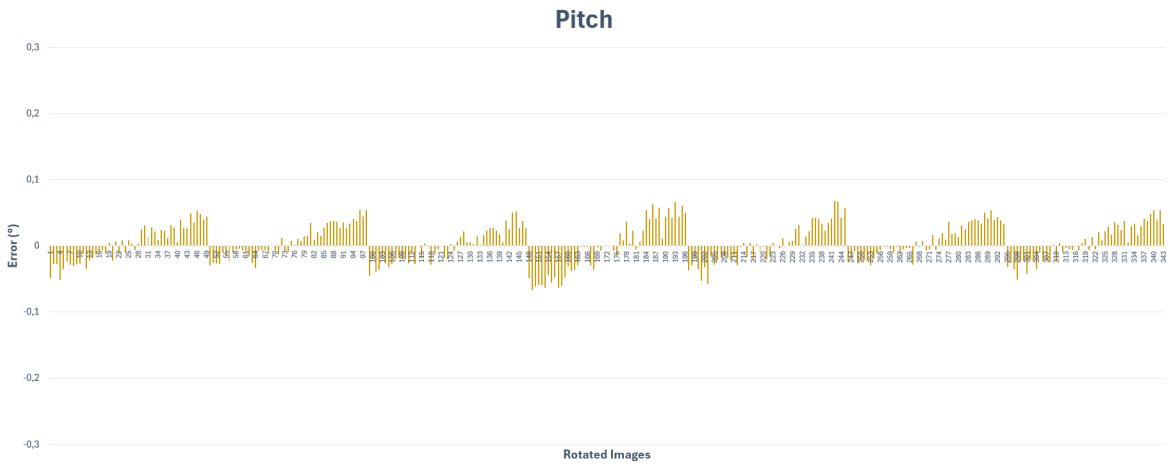


Figure A.2: Pitch error for 343 rotated images captured with the pinhole front camera.

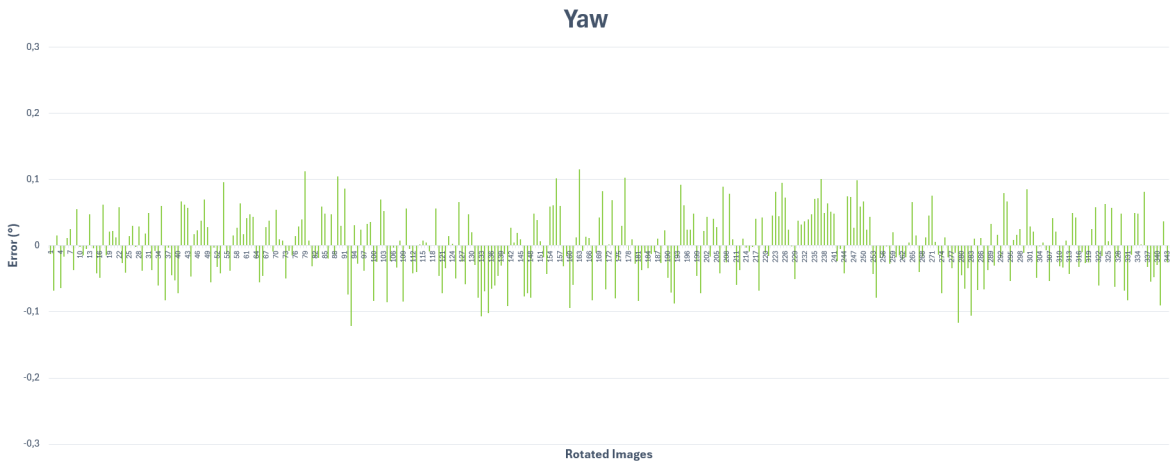


Figure A.3: Yaw error for 343 rotated images captured with the pinhole front camera.

A.2 Error Fisheye Front Camera

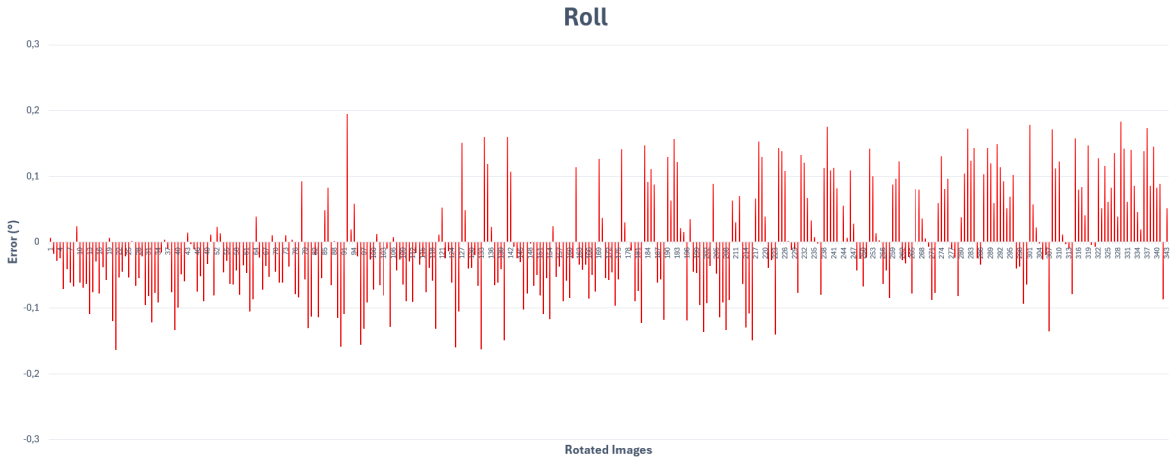


Figure A.4: Roll error for 343 rotated images captured with the fisheye front camera.

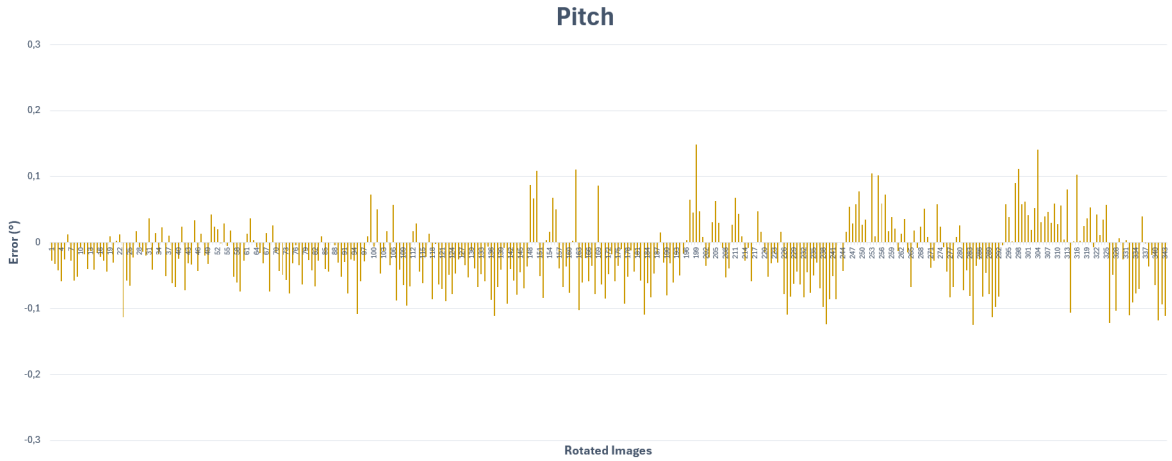


Figure A.5: Pitch error for 343 rotated images captured with the fisheye front camera.

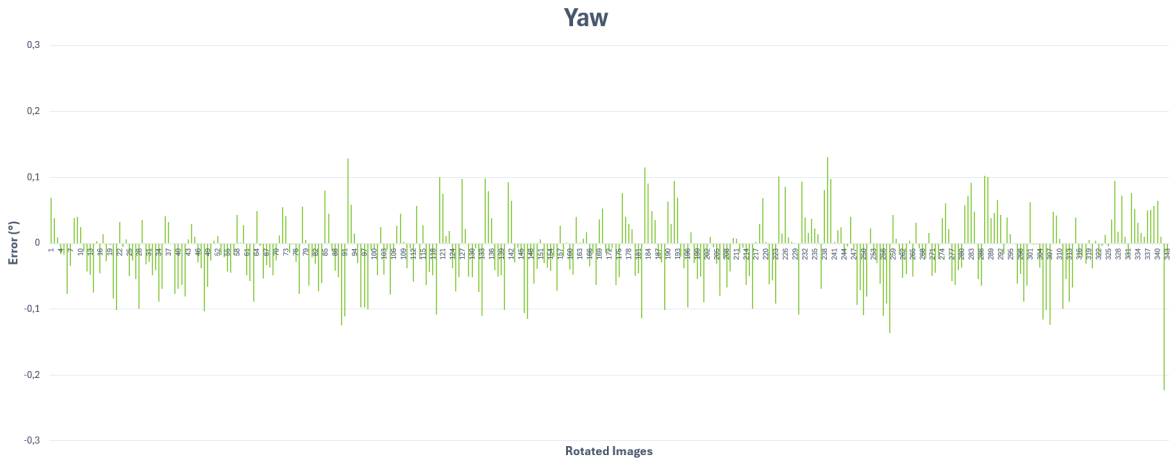


Figure A.6: Yaw error for 343 rotated images captured with the fisheye front camera.

A.3 Error Fisheye Back Camera

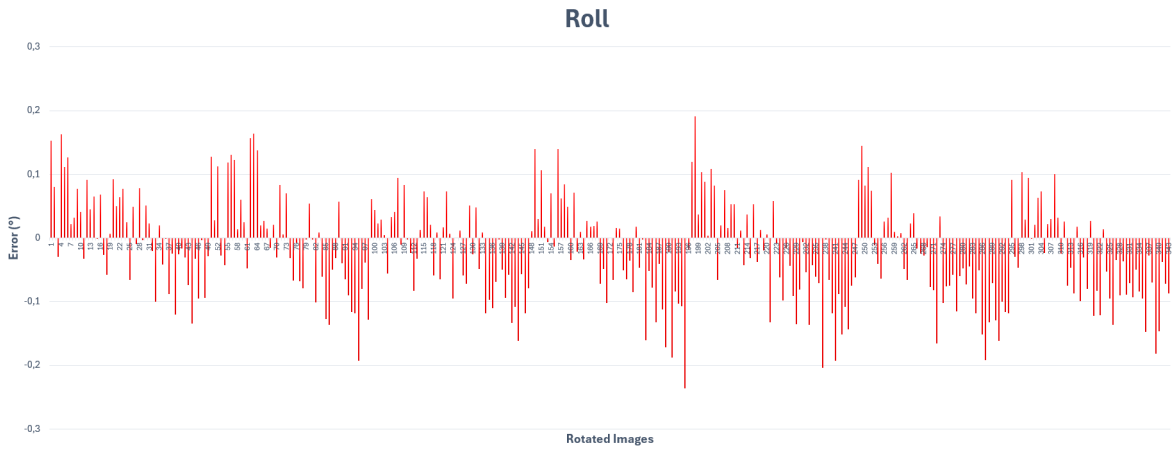


Figure A.7: Roll error for 343 rotated images captured with the fisheye back camera.

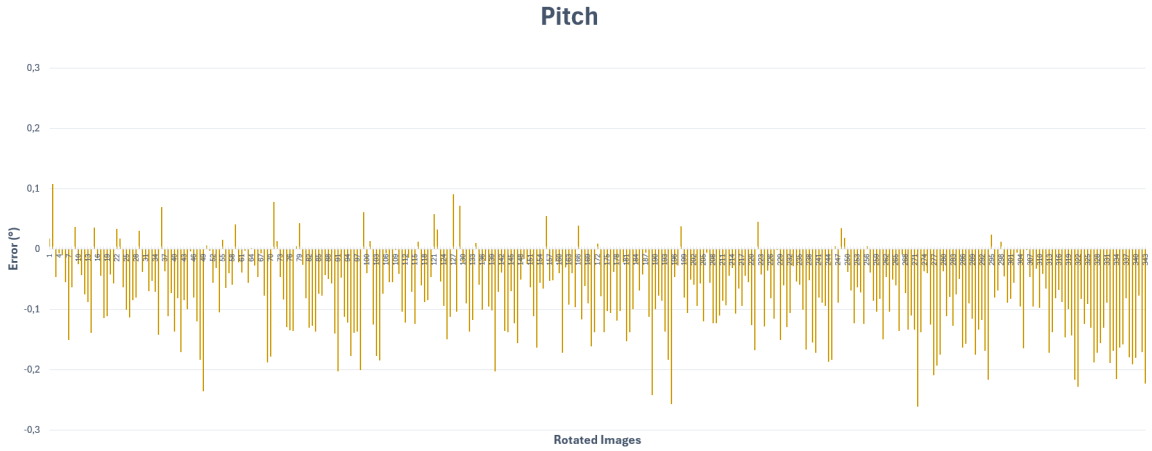


Figure A.8: Pitch error for 343 rotated images captured with the fisheye back camera.

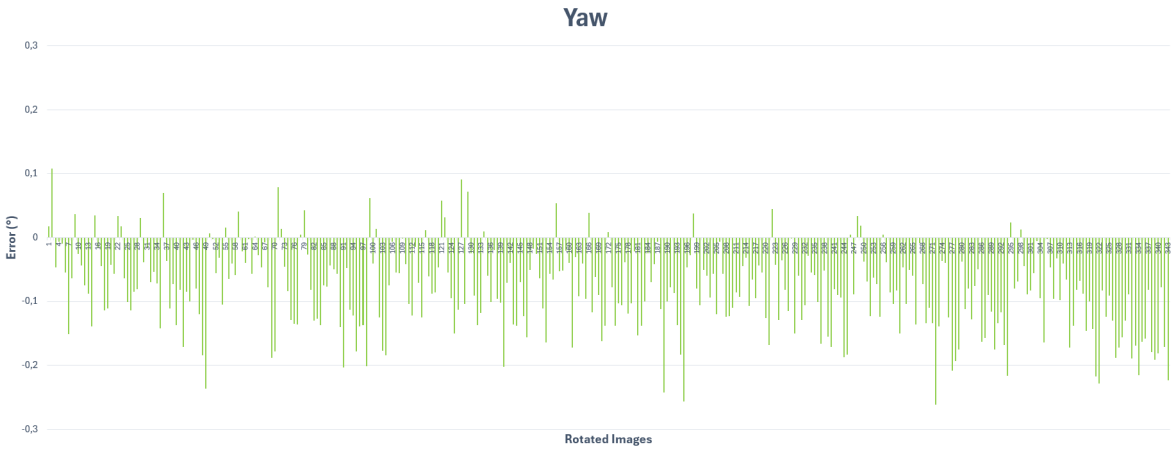


Figure A.9: Yaw error for 343 rotated images captured with the fisheye back camera.

A.4 Error Fisheye Right Camera

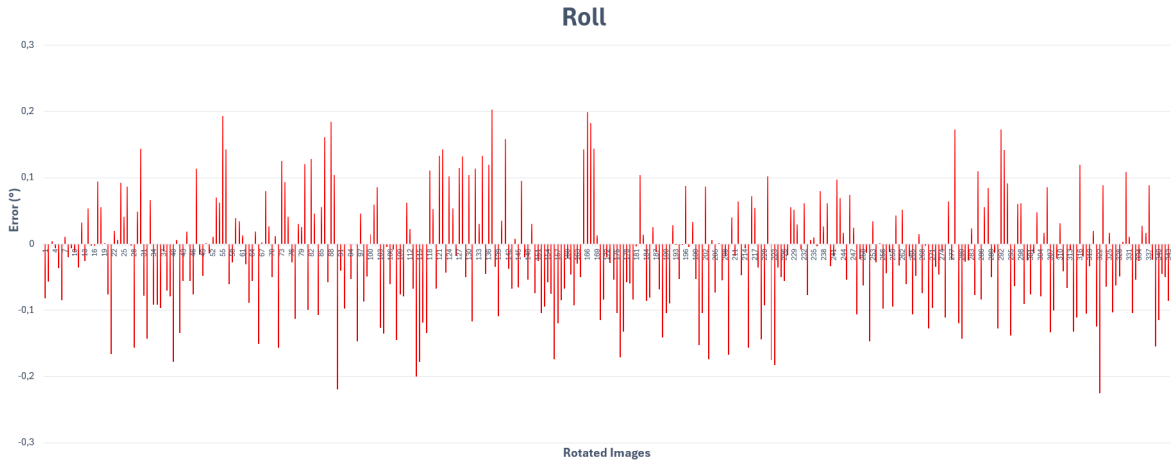


Figure A.10: Roll error for 343 rotated images captured with the fisheye right camera.

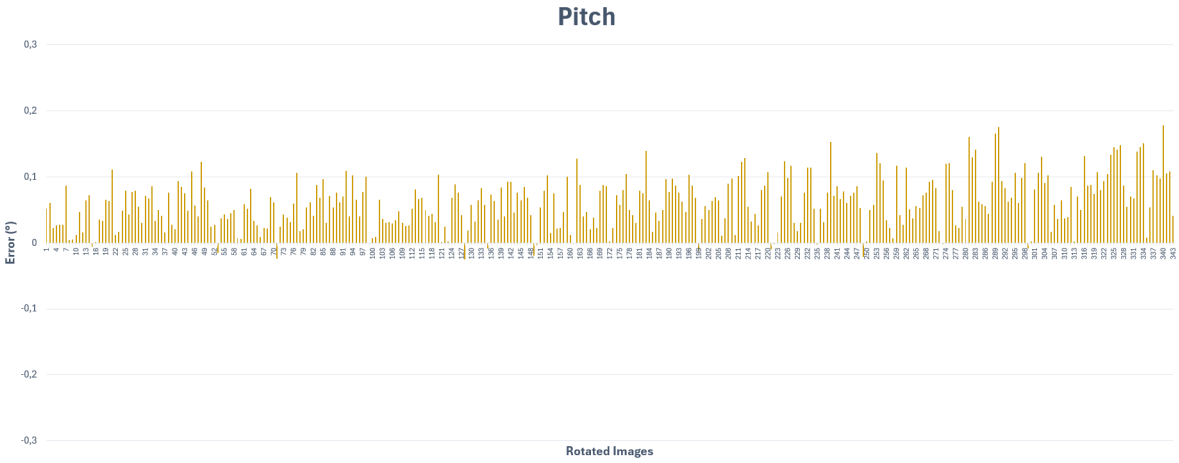


Figure A.11: Pitch error for 343 rotated images captured with the fisheye right camera.

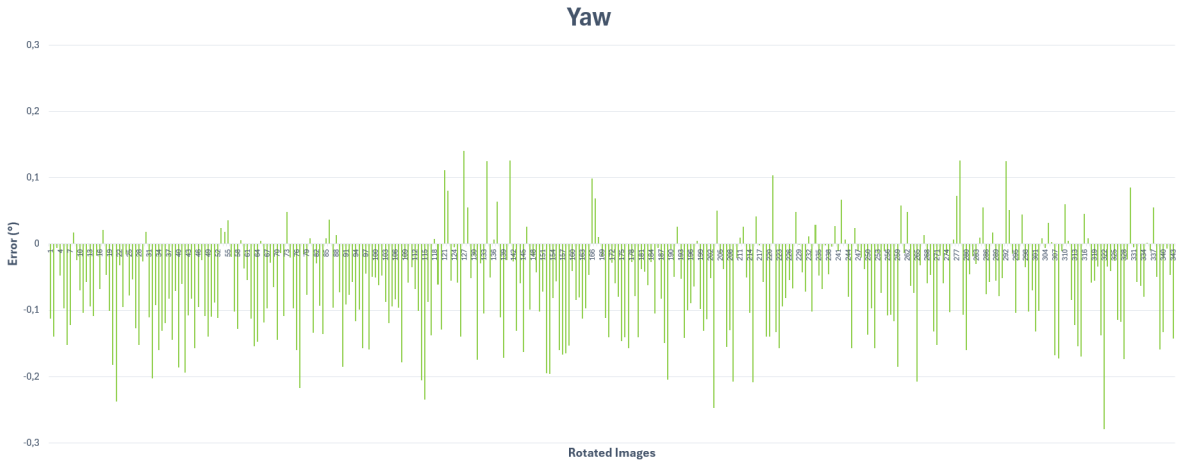


Figure A.12: Yaw error for 343 rotated images captured with the fisheye right camera.

A.5 Error Fisheye Left Camera

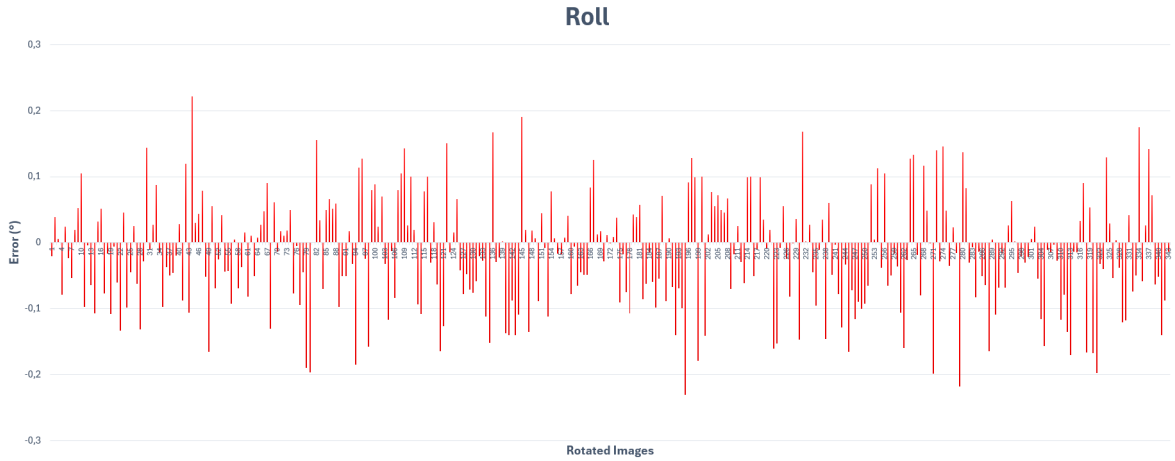


Figure A.13: Roll error for 343 rotated images captured with the fisheye left camera.

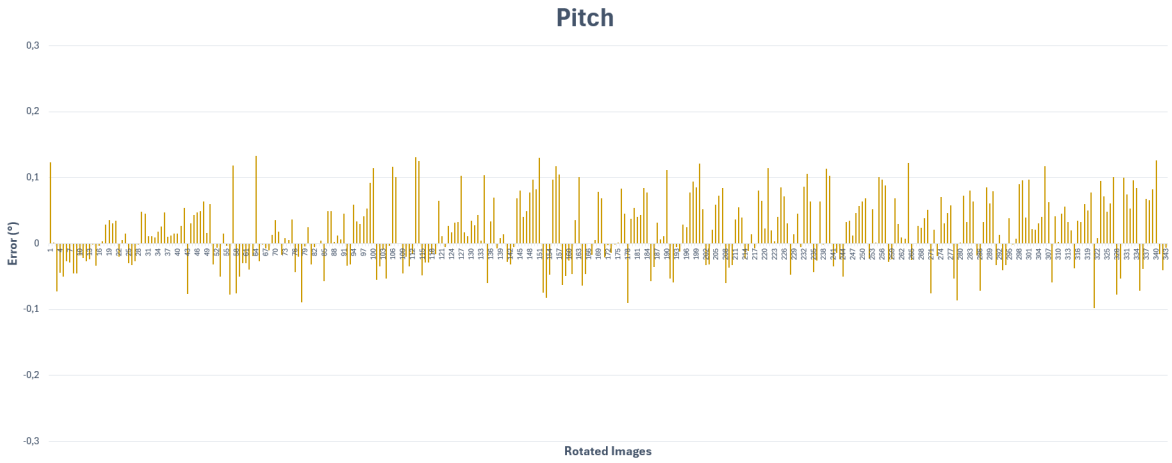


Figure A.14: Pitch error for 343 rotated images captured with the fisheye left camera.

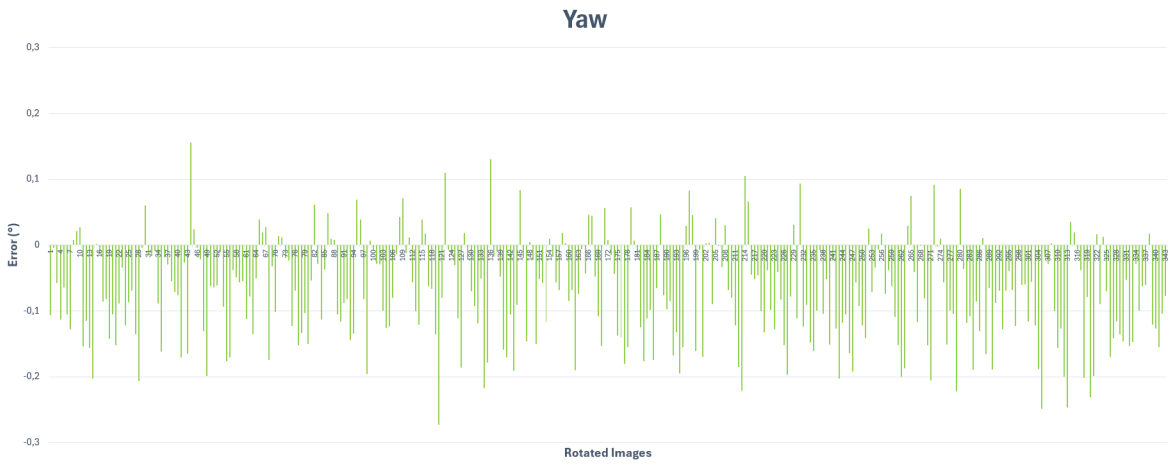


Figure A.15: Yaw error for 343 rotated images captured with the fisheye left camera.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY