Racetrack width and car relative position extraction using an image-based segmentation model

OSKAR JOHANNES PIIBAR, University of Twente, The Netherlands

Computer vision is widely used and researched in the general automotive and racing industry; however, it is almost never used alone, but assisted by other sensors such as GPS and LiDAR for better environment perception. This research relies fully on computer vision to extract spatial data from the simulation (sim) racing environment Assetto Corsa (AC), where direct access to track dimensions is unavailable. This work addresses the challenge of extracting such data visually to reduce the dependency of reinforcement learning agents on hard-coded track information and improve their ability to generalise to tracks it has never seen. A light-weight image-based segmentation model was developed to detect the track and its road boundary points. The model relies on a histogram-based approach that scans the binary image horizontally to find white pixel value peaks corresponding to road markings. The output of that model was used to calculate the Euclidean distance between the boundary points. The results are compared to ground truth data using metrics such as R-squared, MAPE, Pearson's correlation coefficient, and residual analysis. The results: MAPE of 15.91%, R^2 of 0.1164, Pearson's of 0.917 and a mean residual of 1.76 meters demonstrate that the model performs adequately when extracting spatial data from the the AC sim racing environment relying only on image-based input.

Additional Key Words and Phrases: sim racing, image-based segmentation, racetrack detection, assetto corsa, computer vision, width estimation, relative positioning.

1 INTRODUCTION

Simulation racing (sim racing) offers a controlled and cost-effective environment to develop and test autonomous driving techniques in a lifelike setting. Simulators such as Assetto Corsa (AC)[1] are still used in the sim racing community due to their customizability [7], realistic physics engines, and integration of the Python programming language [21]. These properties allow researchers to conduct repeatable experiments under conditions that closely resemble real-world racing, bridging the gap between simulation and reality while avoiding the risks and expenses of on-track testing.

Precise racetrack geometry data, particularly track width, is crucial in both simulated and real racing. This information is often used for the generation of an optimal racing line and strategy planning for competitive racing [9, 26]. Typically, the required track dimensions are provided by predefined track maps or the game environment itself [14], and by specialized sensors such as LiDAR and GPS in real-world scenarios [26]. Even advanced autonomous racing agents and reinforcement learning (RL) drivers usually incorporate prior knowledge of track boundaries. For example, one approach uses virtual detection lines around the car to measure its distance from the edges of the track [2], second, shapes the vehicle's reward function to encourage driving near the centre of the track [4], another extracts the racetrack boundary data before the learning processes begins [21]. Although some simulation environments, including AC, allow data collection during driving sessions [3, 15], direct access to track width or boundary data is not available through standard APIs or interfaces. As a result, extracting the dimensions of the track from visual data becomes a good alternative option.

This reliance on hard-coded track boundaries highlights a gap in the current state of sim racing: no existing autonomous sim racing agent relies purely on visual perception for understanding the track layout. In practice, simulators such as AC have the track limits built into the game logic (laps are flagged invalid upon leaving the track with more than two wheels) [3], so agents have not had to visually identify the road boundaries. This situation poses a limitation: an agent dependent on predefined track data may struggle to adapt to new or modified tracks without additional information. A purely vision-based method of perceiving the track could remove this dependency and improve an agent's ability to generalise to tracks it has never seen, since all necessary spatial cues would be learned from camera images. This would offer a significant advantage as the agent can extract road data without requiring prior knowledge of the circuit. This motivation forms the basis for the approach explored in this thesis.

The goal of this research is to extract the width of the racetrack, as well as the relative position of the car on the track, using only screen captures from the simulator as input. The approach uses an image-based segmentation model that identifies the drivable track area in AC screenshots. This method is novel in the context of sim racing because it is entirely dependant on the segmented road region produced by the model. This output serves as the basis for determining the dimension of the track and the lateral position of the vehicle, without relying on built-in track data or external sensors. With this approach, the project investigates the extent to which visual data alone can measure spatial information in sim racing context. In the implementation, the output of the segmentation model will be post-processed to compute the track width, and to locate the car relative to the track boundaries. These visually derived measurements will then be compared with ground-truth data from the simulator to evaluate their accuracy. To assess the effectiveness of the image-based model, the results will be evaluated using accuracy metrics, further discussed in the Results section.

2 RESEARCH DESIGN

Despite many advances in autonomous racing, current sim racing AI drivers are highly reliant on *a priori* track information rather than pure visual feedback. This dependency means that existing agents lack the ability to 'truly see' and interpret new tracks on their own, limiting their adaptability. The problem addressed in this research is the absence of an autonomous racing method that can

[1] Assetto Corsa Official Website

Author's address: Oskar Johannes Piibar, o.j.piibar@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

TScIT 43, July 4, 2025, Enschede, The Netherlands

derive essential track parameters using only image data. In other words, no known sim racing agent has been shown to extract race track dimensions and vehicle position exclusively from vision, without external track data. To fill this gap, this research focuses on answering the following research questions.

RQ 1: How can data be extracted from Assetto Corsa with respect to the dimensions of the track and the relative position of the car, using image recognition algorithms?
RQ 2: With what accuracy can spatial data be extracted from the Assetto Corsa using image-based algorithms?

To answer these questions, sub-research questions are defined for both:

SRQ 1.1: What are the available tools to extract track dimension data from Assetto Corsa?
SRQ 1.2: Which tools should be preferred when dealing with an image-based segmentation model?
SRQ 2.1: What are the accuracy metrics to which the gathered data should be compared?
SRQ 2.2: How accurately can an image-based segmentation model extract the width of the racetrack?
SRQ 2.3: How well can an image recognition algorithm detect the relative position of the car on the racetrack?

These sub-questions divide the core problem into two aspects: the first focuses on the available tools and the most suitable options for image-based data extraction, and the second on determining how accurate are the results of the model.

3 STATE OF THE ART

Early approaches to lane and track detection relied on classical computer vision techniques. For example, many systems detected lane boundaries using colour segmentation, low-level edge detectors, intensity histograms, Canny edge detection, and geometric transforms such as Hough lines [5, 18] or a combination of these. These lightweight methods are computationally efficient and can perform the required tasks in real-time, making them a good option for embedded use. A common pipeline is to apply an inverse perspective mapping (bird's eye view transformation) to the road images, given that the images are from the perspective of a forward-looking dashboard camera, then use a thresholding or colour segmentation to distinguish the road surface from boundaries [18]. Although intuitive and fast, purely feature-based algorithms are easily confused by shadows, non-lane markings, or missing boundary lines [18]. In the context of sim racing, classical techniques remain effective, since the track boundaries are often visually distinct from the surroundings, but they struggle with variations in environments settings, such as lighting, texture, track curvature, and new tracks.

3.1 Deep Learning and Segmentation Approaches

More recent work uses deep neural networks for track segmentation. Convolutional Neural Networks (CNN), more specifically UNet and SegNet [10], can classify each pixel as road or non-road with high accuracy. However, these models, including the latest YOLOv8 [24] require large memory limits, which become a barrier when deploying them to devices with limited resources [10]. Lightweight CNN models, such as ERFNet [20], have emerged to balance speed and accuracy, to demonstrate that real-time performance is possible without heavy hardware.

In the context of autonomous and sim racing, where speeds are high, there are often no clear lane markings, and images become blurry [10], specific datasets, such as RoRaTrack [10], and solutions are required for road segmentation. A study [10] developed a new RaceGAN model, which used a combination of a WeBACNN [10] and a generative adversarial network (GAN), composed of a generative and discriminative model [12], for the detection of the race track. WeBACNN targets specifically racing scenarios and treats track detection as a vision-based task [10], demonstrating the feasibility of a camera-only approach. This acts as a motivating factor to consider image-based approaches as alternatives to LiDAR or precise maps. Additionally, cameras are not only cheaper than LiDAR sensors, but struggle less in cases where there are no bounding walls for the laser to bounce off from [10].

3.2 Gap in Existing Research

As highlighted in the previous sections, there are classical and machine learning (ML) road segmentation approaches that have been widely researched. However, an area that has barely been studied is collecting spatial data from the segmented areas. In an urban context, a study was conducted by *Seo et al.* [22], where the geometry of the road was measured using images captured by a monocular camera placed on the dashboard of the car. In sim and racing context the tracks are hard coded in the game [3], available through predefined track maps [14], or gathered with a combination of GPS and LiDAR sensors [26]. Therefore, pure vision-based approaches to find race track spatial data have not been studied.

3.3 Preferred Approaches and Tools

To allow a fast response time by the model and to support RL agents in generalising to tracks it has never seen, a light-weight approach is preferred. Therefore, the OpenCV-based pipeline described by Bouziane and Rami [5], or a small and efficient ML model such as SegNet or ERFNet [20] are good choices for real-time road segmentation. After fast road detection, spatial features can be extracted using the Euclidean distance [16], which is a good option, as images are two-dimensional.

4 METHODOLOGY

In this section, the methodology of the practical part will be discussed. This covers the specifics of the steps taken to capture the AC images, how these images were preprocessed, the tools used for road detection, and how the width and relative positioning of the car was calculated.

4.1 Environment Setup

The images of the racetrack were collected using the top-down viewpoint to avoid perspective distortion, which occurs when the camera is angled rather than pointing straight at the object [6]. To

Table 1. Camera Settings

Setting	Value	Units
Distance	0.000000	Meters
Height	14.00000	Meters
Pitch	0.00000	Radians

achieve this point of view, some game systems (Appendix A) had to be altered to the values shown in Table 1. The `DISTANCE` parameter refers to the horizontal offset behind the car, measured in meters (`DISTANCE=0` is exactly at the car's centre, `DISTANCE=`y, where $y > 0$, y meters behind the car). The `HEIGHT` parameter, measured in meters, determines the vertical distance from the reference point of the vehicle. The value 14.0 was selected by practical testing on the racetrack to ensure that both boundary lines can be seen on most of the racetrack. The only exception is the section where the pit lane merges with the main track, where one of the sidelines may not be visible. Lastly, the `PITCH` parameter refers to the vertical camera orientation in radians, ranging from $[-\pi, \pi]$, where $\pm\pi$ is straight up, $\pm\frac{\pi}{2}$ is straight ahead (forward, backward) and 0 straight down.

4.2 Data Gathering

After initialization of the environment with the desired settings, images were captured (Figure 1) and some environment data of the location of the car were documented. Initially, the images were intended to be captured using the built-in AC *takeScreenshot* function, which proved to be unreliable. As an alternative, an external Python script using the *pyautogui* library was implemented, which captured screenshots over a regular interval. Each of the taken images was titled with a timestamp, representing the number of milliseconds passed since the Unix epoch (00:00:00 Coordinated Universal Time (UTC) January 1, 1970) [23]. Parallel and following the data

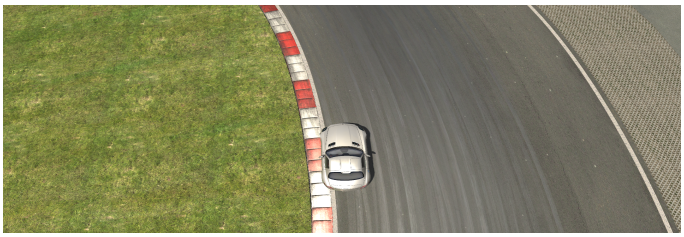

Fig. 1. One of the collected images from the data gathering.

gathering tool developed by Assies [3], a custom Python app was integrated into AC to record the coordinates of the car (x, y) and its relative position on the lap (ranging from 0 at the start to 1 at the finish), using the same timestamp as in the image taken. The z-coordinate was not included in the collected coordinates of the car, as the ground truth data – discussed in the results section – treats the racetrack as a flat two-dimensional surface.

4.3 Image Preprocessing

The gathered images were next preprocessed before any racetrack width or relative position of the car could be extracted from them. For that, a pipeline was developed that combines several threshold-based and filtering techniques. Firstly, a template matching algorithm [13] (Appendix D) was used that for each picture identified the car on the input image based on a series of close-up pictures. The generated car mask, highlighted with the blue rectangle in Figure (2), was then removed from the image to reduce additional noise. Afterwards, the background of the images was targeted. Usually most of the racetracks are surrounded either by grass, gravel, or asphalt, which allows to filter the images on specific colour boundaries. For that, the images were first transformed into an HSV format (Hue, Saturation, Value) to better segment the colours. The yellow (gravel) and green (grass) colour masks were applied, isolating the likely road surface.

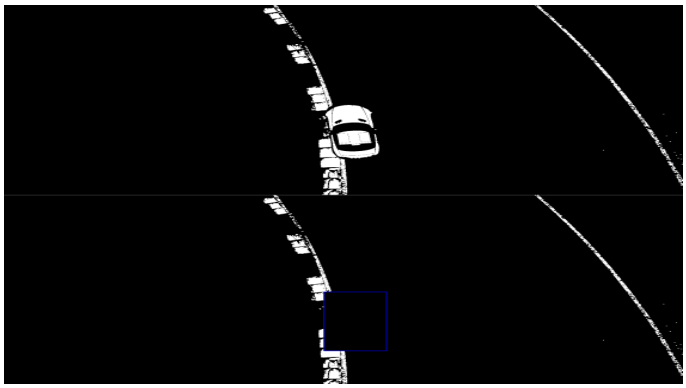

Fig. 2. Binary image before and after the car mask has been removed from the image.

Next, the filtered image is converted to greyscale, and the global intensity threshold is applied to highlight high-contrast features, such as lane markings and curbs. The remaining binary image (black-and-white), where the pixels have a value of 0 or 255, is multiplied together with the previously computed road masks to ensure that only the road areas remain.

Finally, to reduce unnecessary noise that appeared after applying a combination of filters, a morphological opening operation was applied using a 3 x 3 kernel. Morphological opening is an operation that includes an erosion that removes small objects (isolated bright pixels) and a dilation that attempts to restore the shape of the remaining objects [11]. The operation involves sliding a 3 x 3 kernel matrix, consisting of ones, over the binary image. At each location, the pixel value is set to 1 only if all the pixels under the kernel are 1 for the erosion step; and if any of the pixels under the kernel are 1 for the dilation step. Mathematically, for a binary image I and a kernel S, erosion and dilation are defined as in the following equations (1), (2) (pages 641 and 643 in the book by Gonzalez [11]):

$$\text{Erosion: } I \ominus S = \{z \mid S_z \subseteq I\}, \tag{1}$$

$$\text{Dilation: } I \oplus S = \{z \mid (\hat{S})_z \cap I \neq \emptyset\}, \tag{2}$$

where S_z is the translation of the kernel S by z, and $\hat{S}$ denotes the reflection of S. The morphological opening (3) is defined in the book [11] (page 647) as:

$$I \circ S = (I \ominus S) \oplus S \tag{3}$$

4.4 Racetrack Boundary and Road Detection

Following image pre-processing, a lightweight road detection algorithm was applied to the input images to identify the racetrack and its boundaries. First, a horizontal histogram was computed [8], which sums the intensities of the pixels along the vertical axis (code in GitHub repository in Appendix D). The peaks in the histogram are the concentrations of white pixels, which usually correspond to the lane markings (Figure 3). Instead of locating peaks, the detection

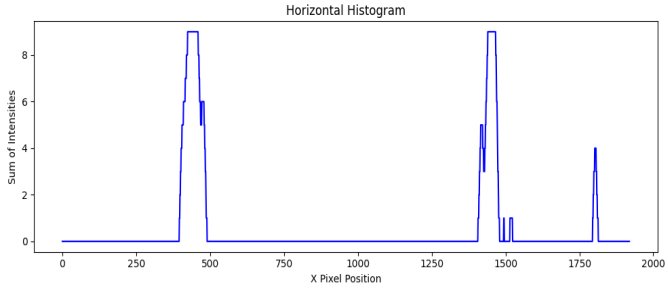

Fig. 3. Horizontal pixel intensity histogram extracted from the centre region of the detected car bounding box.

method searches for flat areas of low white-pixel concentration that are wide enough to be classified as road sections (Figure 3). A width check is performed to ensure that not all flat areas bounded by white lines (the area between 1500-1750 pixels in Figure (3)) could be considered road sections. In AC the road is always bounded by white lines from both sides. Therefore, it is more foolproof to search for two peaks on either side of the flat area, because racetrack surroundings such as sand or other white lines can also appear as sharp peaks on the histogram. The peaks separated by a flat area that is large enough were defined as the edges of the road and displayed on the original image for visual confirmation (Figure 4).

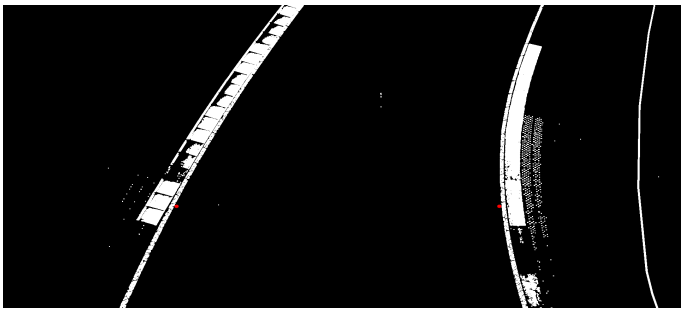

Fig. 4. Found track boundaries (red dots) displayed on the binary image for visual confirmation.

4.5 Meter-to-Pixel Ratio Calculation

To be able to compare the pixel measurements with the ground truth values (in metric measurement), two different pixel-to-meter ratios [17] were calculated.

The first ratio was calculated by dividing the true road width of the first image by the pixel width measured in the same image. This approach ensured that the first measured metric road width is exactly the same as the ground truth. This ratio value was later used to transfer all the other measured pixel widths to metric units.

The second approach was developed to be able to more easily generalise to new tracks, without having to align the first measured and truth data points. It relied on the key assumption that AC models cars with realistic physical dimensions. The real-life width of the 2014 Mercedes-Benz SLS AMG, used for this research, was used [25]. The in-game width of the car was found using the template matching algorithm (Appendix D) that returns the bounding box of the car. The real-life width of the car was divided by the width of the bounding box, resulting in a meter to pixel ratio (4).

$$\text{ratio}_{meter-to-pixel} = \frac{\text{width}_{real}}{\text{width}_{px}} \tag{4}$$

To ensure the precision of the meter-to-pixel ratio, a consistency check was performed. For that, a control image was captured from a camera height of 11 meters. Similarly, the template matching algorithm was used to determine the car's pixel width in the control image.

Next, a camera height ratio was calculated between the height of the driving image and the control image; and a car width pixel ratio between the two pixel widths. To assess the precision of the meter-to-pixel ratio, the normalized difference between the two ratios was calculated (5). If the difference is below a statistically significant threshold of 0.05, the calculated meter-to-pixel ratio is considered reliable.

$$\text{confidence} = \frac{\text{ratio}_{\text{heights}} - \text{ratio}_{\text{pixel widths}}}{\text{ratio}_{\text{heights}}} < 0.05 \tag{5}$$

4.6 Width and Relative Distance Calculation

After obtaining the road boundaries and calculating the meter-to-pixel ratio from the previous steps, the final stage involved calculating the width of the racetrack and the relative position of the car.

The road width was calculated using the Euclidean distance (6) [16] and the resulting pixel values were transferred to meters using both calculated meter-to-pixel ratios.

$$d = \sqrt{(\text{left\_x} - \text{right\_x})^2 + (\text{left\_y} - \text{right\_y})^2} \tag{6}$$

The relative distance to the road boundaries was measured from the centre of the car found with the previously described template matching algorithm (Appendix D). Similarly, to the calculation of the width of the road, the Euclidean distance (6) [16] was used to get metric relative distance values.

5 RESULTS

This section discusses the metrics used to assess the measured results and the gathering process of the ground truth data. Finally, it assesses how well the data performed according to the chosen metrics.

5.1 Assessment Metrics

The measured road width is assessed using the following metrics: Mean Absolute Percentage Error (MAPE) (7), R-squared (8), and Pearson's correlation coefficient (9) (m - measured, t - truth).

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\text{measured} - \text{truth}}{\text{truth}}\right| \times 100\% \tag{7}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\text{measured}_i - \text{truth}_i)^2}{\sum_{i=1}^{n}(\text{measured}_i - \text{mean}_{truth})^2} \tag{8}$$

$$r = \frac{\sum_{i=1}^{n}(m_i - \text{mean}_m)(t_i - \text{mean}_t)}{\sqrt{\sum_{i=1}^{n}(m_i - \text{mean}_m)^2} \cdot \sqrt{\sum_{i=1}^{n}(t_i - \text{mean}_t)^2}} \tag{9}$$

MAPE was chosen because it allows for direct comparison of measurements regardless of the true width of the road. However, MAPE does not cover the direction of the error, for which an additional residual analysis is performed. This will help to assess whether the model over- or underestimates the measured values compared to true values. Moreover, it helps to visualise outliers and give a better context for the other assessment metrics.

Secondly, the R-squared (which can be negative, but usually ranges from 0 to 1) was chosen because it highlights how well the measured data explains the variances in the truth data. A value of 0 indicates that the model does not perform better than predicting the mean of the true values, while values closer to 1 indicate a better fit between the predictions and true values.

Lastly, Pearson's correlation (ranging from -1 to 1) was used to assess the strength and relation of the linear relationship between the predicted and true road widths. A value close to 1 indicates a strong positive linear relationship, 0 indicates no linear relationship, and negative values reflect negative linear relationships.

The relative distance will be compared to the measured road width. More specifically, whether the sum of the two relative distances differs from the width of the road.

5.2 Ground Truth Data

To evaluate the performance and reliability of the developed pipeline, a comparison was made between the extracted measurements and ground truth data. The true data was collected using a reinforcement learning script [21] that maps the limits of the racetrack and the ideal race line and writes the data in a *.csv* file with the following data columns: left border x and y, right border x and y, and race line x and y. The width of the track was calculated using the Euclidean distance (6) on the coordinates of the left and right borders.

The ground truth data had to be mapped to the measurements captured by the Python app. Each data entry collected during gameplay included the car's coordinates and its relative lap position (ranging from 0 to 1). To enable a direct comparison, the exported ground truth data - consisting of approximately 3300 evenly spaced entries along the 5148-meter track - were first augmented with a normalized track position column. This was achieved by assigning each row a linearly spaced value between 0 and 1, representing its relative position along the track. With this normalization, the two datasets could be mapped by selecting, for each captured data point, the ground truth entry with the closest normalized track position.

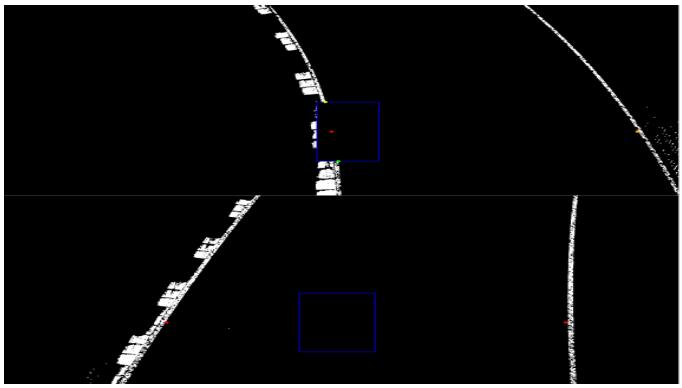

Fig. 5. Two binary images output from the model.

5.3 Model Performance

In total, 907 images were collected. Before analysing the data, some images were removed due to improper screen capture or due to gantries blocking the view of the car (5 images). The resulting images were binary images with the found car mask outlined with a blue rectangle, the edge of the found road marked in green, blue, and orange dots, if the car is on a white line, and red dots, if the car was not on a white line (Figure 5).

5.3.1 Mean Absolute Percentage Error (MAPE). The MAPE values were calculated and inspected for each of the collected data entries. Before the total MAPE was calculated, the data was filtered to exclude images with a MAPE value greater than 0.45 (total of 8 images). This was done to eliminate outliers and have more consistent results.

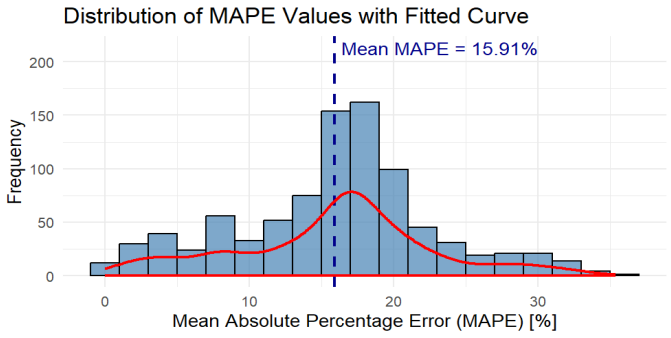

Fig. 6. Distribution of MAPE values.

The model performed adequately in terms of MAPE, with an average error of 15.91% (Figure 6). The most frequent absolute percentage error was in the range of 17 – 19% with approximately 160 measurements. The distribution is slightly skewed to the left, indicating that there were more measurements, where the MAPE values were lower, from the viewpoint of the peak bin.

The results calculated with the meter-to-pixel ratio, found by dividing the real-life width of the car by the pixel width of the car, were worse by 7.27% with a MAPE value of 23.28%. There were very

few errors below the 5% margin, with most errors in the 21% – 23% range (Figure 7). In addition, the distribution is more skewed to the right, having more errors in the range 23% – 33%.

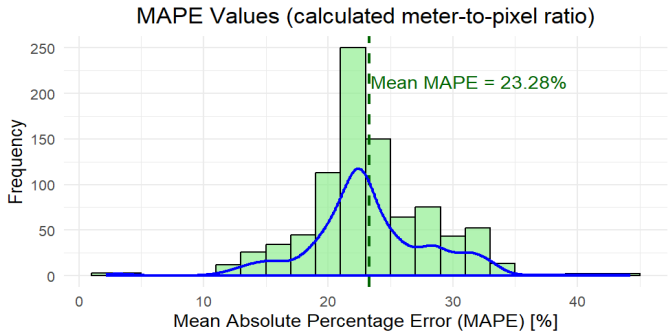

Fig. 7. MAPE values (calculated meter-to-pixel ratio).

Looking at the different types of roads (straight, left and right turns) and the MAPE values on the separate road segments, a similar trend can be seen to the overall MAPE values. The average MAPE values of the groups were quite similar to the general one, with the lowest being the right turn group with an average of 15.3% (Figure 8). The straight road section had an MAPE score of 15.6% and the right turn group 17.6%. It can also be seen that the road width measured during right turns has the most predictions with an MAPE less than 10%. However, for all three groups the peak, where most errors were measured, was around the 16% – 17% range.

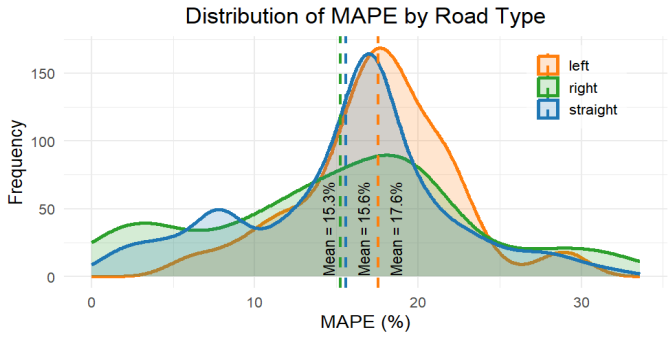

Fig. 8. MAPE values per road type.

5.3.2 R-squared. Similarly, for the MAPE the R-squared was calculated on the whole data set and later for the three different road segments on the race track. For all road widths collected, the value of R-squared was 0.1164, indicating a weak ability of the measured data to explain the variance in the truth data. The results of meter-to-pixel ratio for R-squared were worse than with the chosen ratio variable, having a score of −0.9505.

For the three sections of the road, the results differed from the general ones, as can be seen in Figure (9). The areas of the right turn had an adequate value of 0.453, the left turn section had a value of 0.046 and the straight section scored −0.499 for the R-squared. Summing all of these values roughly equals the R-square score for the whole data set.

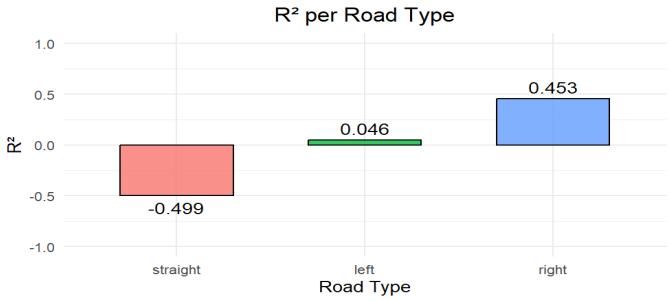

Fig. 9. R-squared per road type.

5.3.3 Pearson's Correlation Coefficient and Residual Analysis. The outcome 0.9166, with close to zero confidence (p-value) (Figure 10), from Pearson's correlation analysis, indicates a strong linear relationship for the predicted data. The other meter-to-pixel ratio achieved the same Pearson's correlation score. The same can be seen for the three sections of the road: straight 0.8969, right 0.9353, left 0.9718. However, without a residual analysis, Pearson's correlation coefficient is not very informative by itself. In Figure (10) the green line is the true width of the road and the blue dots are the measured values. The race track is mostly 9 – 15 meters wide, and most often 10.5 meters wide, which can be seen in the histogram of the true road width in Appendix B. There are significantly less track areas with the road being 8 – 9 and 15 – 22 meters wide. Most of the measured road is between 10.5 – 17.5 meters wide, with fewer measurements in the range 8 – 10 and 17 – 25 meters (Appendix C).

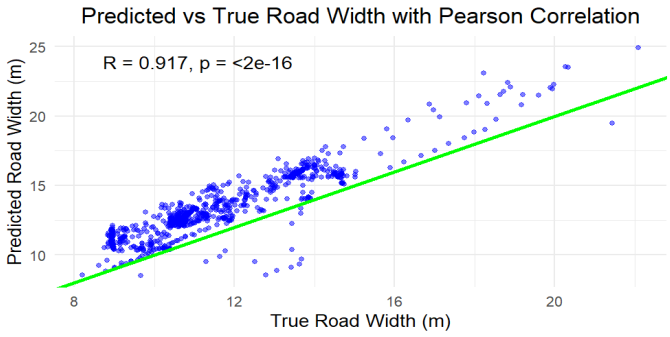

Fig. 10. True vs Predicted width with Pearson's correlation.

The residuals are the vertical offset from the green line to a specific blue dot, which can be both negative and positive. In Figure (10) it can be seen that the model is mainly overestimating the width of the road. This is verified in Figure (11), as most of the residuals are positive, which means higher measured values than the true ones. The most errors, around 275 of them, are in the range of 2–2.5 meters. There are few measurements that are smaller than the ground-truth value and few that are off by more than 4 meters. The histogram in

Figure (11) is slightly left-skewed, with more residuals concentrated below the peak bin.

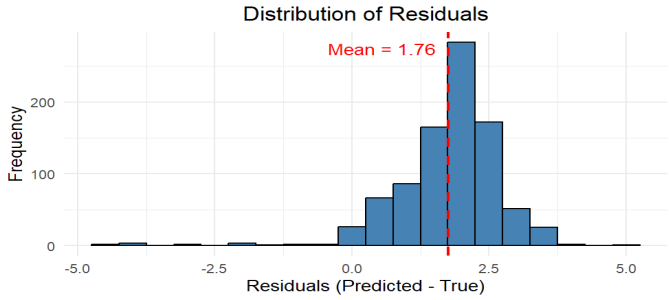

Fig. 11. True vs Predicted residuals histogram.

5.3.4 Relative Distance Measurement Performance. The relative distance measurements were mainly equal to the measured road width; therefore, the residuals had a zero value (Figure 12). There was an outlier that differed from the width of the road by 17 meters, which was removed to reduce noise. Some relative distances differed from the measured road width by $2-4$ meters.

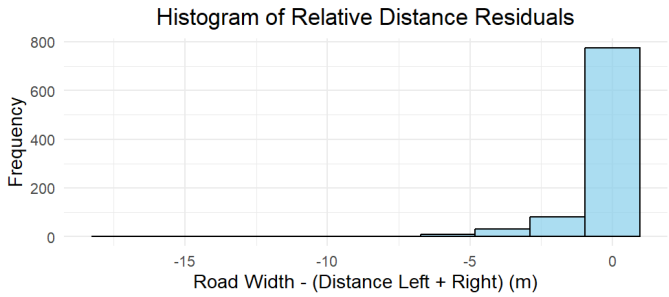

Fig. 12. Road width - Relative distance residuals histogram.

5.3.5 Simple Machine Learning models. To test whether ML techniques could improve road width prediction accuracy, two lightweight models were implemented: Classification and Regression Tree (CART) [19] approach without depth limitation and a shallow neural network (NN), consisting of four 32 neuron layers, with ReLU activation [1]. These models were chosen because they can support real-time feedback and are not computationally heavy. Both models received the measured road width and the car coordinates (x, y) as input and received the true values as a target variable. An 0.8/0.2 training-test split was used.

Both models performed well and improved the measured data assessment metrics, with the CART method having a MAPE of 2.51%, R^2 of 0.7722, and Pearson's correlation of 0.8938. The NN had the following results: MAPE of 4.99%, R^2 of 0.7926, and Pearson's correlation of 0.9224.

The better results are also seen in the residual analysis (Figure 13), where most of the residuals (measured - true) are around 0 for both models. The NN had more residuals near the range 0.5 meters, however, had less outliers compared to the CART method, which had the largest outlier around 8.5 meters (Figure 13).

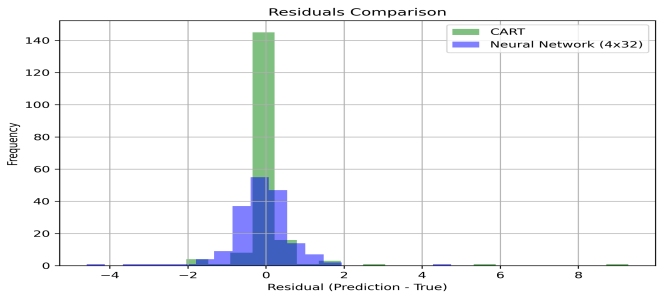

Fig. 13. The residuals of two machine learning models (Prediction - True).

6 DISCUSSION

This section analyses the results more in depth and gives a reasoning for the found values. In addition, the implications on the use cases, limitations, and challenges of the approach taken are discussed.

6.1 Interpretation of Results

The measured mean absolute error is adequate, with an average of 15.91% (Figure 6), however, in the simulation racing context, where small errors can have big consequences, the error can be considered rather large. Tying this value to the average value of the residuals (predicted - true value), the model on average overestimates the road width by 1.76 meters, with the most frequent overestimation occurring around 2 meters (Figure 11), which is approximately the width of an additional car. This suggests a systematic bias that could lead to unsafe decisions if used directly by an RL agent, if it assumes that there is more space on the road than there is actually.

The Pearson correlation coefficient of 0.9166 (Figure 10) indicates a strong linear relationship between predicted and true road widths, even in less reliable segments. This shows that the model is capable of tracking changes in the true values, even if its scale or offset is off. Interestingly, when using a meter-to-pixel ratio between the in-game calculated value and the ground truth-derived value, the model achieved substantially better results. MAPE of 4.91%, R^2 of 0.82, and Pearson's correlation remained the same, indicating a great relation between the measured and truth values. This gives a reason to believe that the ratio variable used may not have been ideal. However, this approach cannot be used in practise, because arbitrarily choosing a ratio variable between the two calculated ones does not have theoretical or empirical justification.

The R-squared for the overall dataset is low and negative for straight-road segments. This suggests that while the predictions increase with the true values (as captured by Pearson's correlation), they deviate in measure. The negative value for the straight areas suggests a poor fit and that the model performs worse than predicting the mean of true values for that section. This discrepancy is most likely due to a consistent bias in the predictions and due to outliers that more heavily influence the metric. For the right turn section, the model had the highest R^2 value of 0.453 (Figure 9) as the edge

cases (driving on the line or off the road) were less tested on these sections, indicating adequate core functioning of the model. In general, the R^2 score indicates that the model captures some structure but lacks the precision required for strong regression performance.

The results achieved with the ML models suggest that additional use of simple supervised learning with the used approach can compensate for systematic errors in the measurements, since the residuals in Figure (11) are consistent.

6.2 Limitations

There are some limitations that affect the performance of the model. Firstly, the capturing of the images is done by an external script (not in-game), which causes small discrepancies between the ground truth and measured values. Taking and saving a screenshot takes around 50 ms, during which the car keeps moving, and the position of the car and the truth value are mapped after the image has been saved. Secondly, there might be a small difference between the indices used for relative positioning on the race track and the normalised ground truth values. However, the width of the road does not change suddenly; therefore, the two discrepancies should barely influence the predictions.

The model is light-weight and does not use any ML, which can be a limiting factor for detecting the car when the environment settings are changed (weather and the circuit). Additionally, the model, due to its architecture, sometimes struggles with detecting the track boundaries in edge cases, such as driving on the line or when the car is completely off the race track.

The limiting factor in using ML models to improve the predicted road width is that it requires truth data, which are not always available.

6.3 Implications on Use Cases

The model shows promising baseline results; however, for real-time use, for example in RL agents, consistency and reliability under varying conditions is required. In its current form, the model could serve as a perception module for making a rough initial scan of the race track or to be used in constrained environments for initial road segmentation. However, the varying accuracy across road segments currently limits the use for open and unconstrained RL environments.

To be integrated into unconstrained environments, the model should be more accurate and robust, but also predictable in the areas where it fails. Understanding weaknesses of the model is necessary to integrate uncertainty estimates or fall-back logic into the pipeline. This would help the agent learn to navigate imperfect information rather than relying on ideal estimates. Implementing this would increase the generalisability of the model and make it more suitable for real-world decision-making challenges.

7 CONCLUSION

This thesis explored whether the width of the road and the relative position of the car can be extracted from images using a light-weight segmentation model. Real-time image capture from AC and a custom processing pipeline demonstrated that spatial data can be derived only based on visual input.

There are different tools, such as ML models, neural networks; and light-weight options, based on canny edge detection, sliding windows and histogram-based edge detection, available to extract track dimension data from AC. However, in the context of RL agents that depend on real-time feedback, light-weight methods tend to perform better because of shorter processing times and lower computational overhead. To support faster decision-making, a lightweight model was chosen that uses colour-based binary filtering and, afterward, a histogram-based approach to segment the racetrack and detect the road boundaries.

The results gathered were compared against the ground truth values using the following metrics: MAPE, R-squared, Pearson's correlation coefficient, and a residual analysis. These four metrics provided a good overview of the model's performance and gave valuable insight into its shortcomings. The measured pixel results were converted to metric units using a calculated meter-to-pixel ratio.

The predicted results had the following scores for the assessment metrics: MAPE of 15.91%, R^2 of 0.1164, Pearson's correlation of 0.9166, and the mean of the residuals was 1.76 meters. These results suggest that the model does an adequate job of measuring the width of the road only from image-based input. However, the low R-squared value with a combination of the Pearson correlation coefficient indicates that while the model is capable of tracking and adapting to the changes in the true values, it is constantly off by a margin.

Implementing simple supervised learning ML models (CART and NN) improves predicted road width measurements and achieves substantially better results with the following assessment metric scores: MAPE of 2.51%, R^2 of 0.7722, Pearson's correlation of 0.8938 for the CART model; and MAPE of 4.99%, R^2 of 0.7926, Pearson's correlation of 0.9224 for the NN.

The relative distance was measured in meters from the centre of the car to both boundary points of the road. It was assessed by looking at the residuals that remained when deducting the sum of the two relative distances from the measured road width. Approximately 800 of the total 900 measurements had zero residuals and the rest had residuals up to 5 meters. This indicates that the model performed well to measure the relative distance.

8 FUTURE WORK

Although the model performed adequately, there are aspects that require improvement, additional functionalities to be added, and other approaches to be tested. Firstly, a more accurate and consistent method for deriving the meter-to-pixel ratio should be explored. Secondly, using a chase camera setting instead of a top-down view could help to anticipate the upcoming road sections and make it more suitable to be used by an RL agent. Furthermore, basing road and car detection mechanisms on ML methods could help to increase the robustness, especially in complex, more noisy scenes. Evaluating the system within RL agents is a next step in assessing its usefulness in learning-based environments. Lastly, the model could benefit from faster performance to ensure usability in time-constrained applications.

REFERENCES

[1] Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375* (2018). https://arxiv.org/abs/1803.08375

[2] B. Amutha, Kadiyala Ramana, Gaurav Dhiman, G. Ashok, V. Bhaskar, A. Sharma, Gurjot Singh Gaba, Mehedi Masud, and Javed F. Al-Amri. 2021. Multimedia Concepts on Object Detection and Recognition with F1 Car Simulation Using Convolutional Layers. *Wireless Communications and Mobile Computing* 2021 (2021), 1–21. https://doi.org/10.1155/2021/5543720

[3] Daan Assies. 2021. Developing a Smart Telemetry Feedback System for Sim Racing. https://essay.utwente.nl/87322/. Accessed: 2025-04-30.

[4] Jean-Baptiste Authier-Carcelen and Rubina Zadourian. 2024. A Driving Model in the Realistic 3D Game Trackmania Using Deep Reinforcement Learning. https://doi.org/10.20944/preprints202409.0778.v1. Accessed: 2025-04-30.

[5] Amal Bouziane and Salma Rami. 2025. Real-time lane detection system using OpenCV and Kalman filter-based path prediction. *International Journal of Automobile Engineering* 6, 1 (2025), 52–58. https://www.mechanicaljournals.com/ijae/article/51/6-1-9-305.pdf

[6] C. Chun, S. S. Xie, S. C. Wong, and K. Choi. 2013. Generalized camera calibration model for trapezoidal patterns on the road. *Optical Engineering* 52, 1 (2013), 017006–017006. https://doi.org/10.1117/1.OE.52.1.017006

[7] Sergio Hernández de Frutos and Manuel Castro. 2021. Assessing Sim Racing Software for Low-Cost Driving Simulator to Road Geometric Research. *Transportation Research Procedia* 58 (2021), 575–582. https://doi.org/10.1016/j.trpro.2021.11.076

[8] Vighnesh Devane, Ganesh Sahane, Hritish Khairmode, and Gaurav Datkhile. 2021. Lane Detection Techniques using Image Processing. In *ITM Web of Conferences*, Vol. 40. EDP Sciences, 03011. https://doi.org/10.1051/itmconf/20214003011

[9] Gonçalo Faria and F. Pimentel Da Mota. 2024. Autonomous Vehicle Racing - Perception. https://repositorio-aberto.up.pt/handle/10216/162669. Accessed: 2025-04-30.

[10] Shreya Ghosh, Yi-Huan Chen, Ching-Hsiang Huang, Abu Shafin Mohammad Mahdee Jameel, Chien Chou Ho, Aly El Gamal, and Samuel Labi. 2025. A Racing Dataset and Baseline Model for Track Detection in Autonomous Racing. *arXiv preprint arXiv:2502.14068* (2025). https://arxiv.org/abs/2502.14068

[11] Rafael C. Gonzalez and Richard E. Woods. 2018. *Digital Image Processing* (4th ed.). Pearson.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, Vol. 27. 2672–2680.

[13] Nazanin Sadat Hashemi, Roya Babaie Aghdam, Atieh Sadat Bayat Ghiasi, and Parastoo Fatemi. 2016. Template Matching Advances and Applications in Image Analysis. *arXiv preprint arXiv:1610.07231* (23 Oct 2016). https://arxiv.org/pdf/1610.07231

[14] Fazilat Hojaji, Adam J. Toth, John M. Joyce, and Mark J. Campbell. 2024. AI-enabled Prediction of Sim Racing Performance Using Telemetry Data. *Computers in Human Behavior Reports* 14 (2024), 100414. https://doi.org/10.1016/j.chbr.2024.100414

[15] I. I. Husyeva, I. Navas-Delgado, and José García-Nieto. 2025. Data-Driven Approaches for Efficient Vehicle Driving Analysis: A Survey. *Journal of Sensor and Actuator Networks* 14, 3 (2025), 52–52. https://doi.org/10.3390/jsan14030052

[16] Umer Idrees. 2016. CHAPTER 4––Methods for Measuring Distance in Images. Academia.edu. https://www.academia.edu/29248316/CHAPTER_4_4_METHODS_FOR_MEASURING_DISTANCE_IN_IMAGES#loswp-work-container Published October 18, 2016.

[17] S. Kampati, U. Yashwan Goud, and V. Reddy. 2025. Deepvision: Real-Time Object Detection and Dimension Estimation Using Yolov4. *International Journal of Future Multidisciplinary Research (IJFMR)* 7 (2025), Article ID 250343751. https://doi.org/10.1234/ijfmr.2025.250343751

[18] H. Khan, A. Rafaqat, A. Hassan, A. Ali, W. Kazmi, and A. Zaheer. 2020. Lane detection using lane boundary marker network with road geometry constraints. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2160–2169. https://openaccess.thecvf.com/content_WACV_2020/papers/Khan_Lane_detection_using_lane_boundary_marker_network_with_road_geometry_WACV_2020_paper.pdf

[19] Wei-Yin Loh. 2011. Classification and Regression Trees. In *International Encyclopedia of Statistical Science*, Miodrag Lovric (Ed.). Springer, 196–201. https://www.researchgate.net/publication/227658748_Classification_and_Regression_Trees

[20] Yangyang Qiu, Guoan Xu, Guangwei Gao, Zhenhua Guo, Yi Yu, and Chia-Wen Lin. 2024. Efficient Semantic Segmentation via Lightweight Multiple-Information Interaction Network. *arXiv preprint arXiv:2410.02224* (2024). arXiv:2410.02224 [cs.CV] https://arxiv.org/abs/2410.02224 Submitted October 3, 2024.

[21] Adrian Remonda, Nicklas Hansen, Ayoub Raji, Nicola Musiu, Marko Bertogna, Eduardo Veas, and Xiaolong Wang. 2024. A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data. *arXiv preprint arXiv:2407.16680* (2024). https://arxiv.org/abs/2407.16680

[22] Y.-W. Seo and R. Rajkumar. n.d.. *Use of a Monocular Camera to Analyze a Ground Vehicle's Lateral Movements for Reliable Autonomous City Driving*. Technical Report PPNIV-13-YWSeo. Carnegie Mellon University. https://www.ri.cmu.edu/pub_files/ppniv-13-ywseo.pdf Retrieved June 29, 2025.

[23] The Open Group 2024. *The Open Group Base Specifications Issue 8, Section 4.19*. The Open Group. Accessed: 2025-06-29.

[24] R. Varghese and M. Sambath. 2024. Yolov8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In *Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. IEEE, 1–6.

[25] Wikipedia Contributors. 2025. Mercedes-Benz SLS AMG. https://en.wikipedia.org/wiki/Mercedes-Benz_SLS_AMG. Accessed June 28, 2025.

[26] Jasmina Zubača, Michael Stolz, and Daniel Watzenig. 2020. Smooth Reference Line Generation for a Race Track with Gates based on Defined Borders. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. 604–609. https://doi.org/10.1109/IV47402.2020.9304722

A CAMERA SETTING SPECIFICATIONS

To achieve the desired top-down view, the *chase_cam.ini* file, which is located in the *steamapps/common/assettocorsa/system/cfg* folder, had to be altered. More precisely, the variable '[*CHASE*_0]' was altered to the values mentioned in Table (1).

B TRUE ROAD WIDTH DISTRIBUTION HISTOGRAM

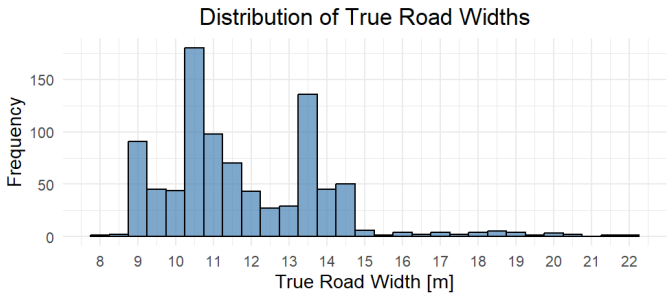

Fig. 14. True road width distribution histogram.

C MEASURED ROAD WIDTH DISTRIBUTION HISTOGRAM

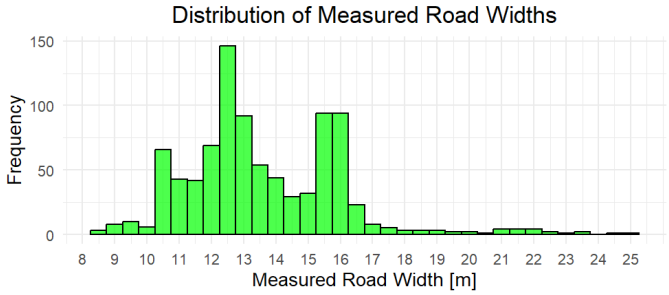

Fig. 15. Measured road width distribution histogram.

D GITHUB CODE REPOSITORY

Below is the link to the GitHub repository that contains the code used for the practical part of the research. Link to GitHub repository