CHALMERS
UNIVERSITY OF TECHNOLOGY

Threat assessment from naturalistic video-data: How to detect, classify, and estimate the position of multiple road users from cameras.

Master's thesis in Mobility Engineering

Luhan Fang
Yahui Wu

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's Thesis in Mobility Engineering

Threat assessment from naturalistic video-data: How to detect, classify, and estimate the position of multiple road users from cameras.

Luhan Fang
Yahui Wu

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology
Gothenburg, Sweden 2024

Threat assessment from naturalistic video-data: How to detect, classify, and estimate the position of multiple road users from cameras.
Master's thesis in Mobility Engineering
Luhan Fang
Yahui Wu

Supervisor: Rahul Rajendra Pai & Alexander Rasch
Examiner: Marco Dozza

Master's Thesis 2024
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Application of the threat assessment pipeline on the naturalistic data.

Typeset in LaTeX
Gothenburg, Sweden 2024

Threat assessment from naturalistic video-data: How to detect, classify, and estimate the position of multiple road users from cameras.
Master's thesis in Mobility Engineering
Luhan Fang
Yahui Wu
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology

Abstract

Each year, over one million people die in traffic-related crashes, with more than half involving vulnerable road users (VRUs) such as pedestrians, cyclists, and e-scooterists. To mitigate the crashes, it is important to understand the causation mechanisms of such crashes and analyze the interactions and behaviors of VRUs. Using the knowledge of these crash mechanisms and rider behavior, threat assesment algorithms can be developed to identify anomaly in rider behavior and likelihood of a crash. Naturalistic data has proven to be an effective tool in traffic safety research to identify the road user interaction and crash mechanisms. However, as the data tends to be large, traditionally the safety critical events have been identified based on kinematic trigger. However, in some safety critical scenarios, the vehicles may not exhibit any change in kinematics, which often leads to their exclusion from analysis. Given the sheer volume of data, manually reviewing each video requires significant time and resources making it unviable. With the development of computer vision, investigating the video footage can be very effective. Therefore, by using computer vision methods, we can accurately identify different road users within the footage. Once a road user is identified, machine learning models can be utilized to convert the 2D position on the video frame into real-world positions and kinematics of the road users. Threat assessment algorithms can use these positions and kinematics to determine safety-critical scenarios. This approach makes identification of safety critical scenarios using videos feasible in addition to making efficient use of the collected data.
This thesis proposes a system that first utilizes the version 7 of You Only Look Once (YOLOv7) model to conduct object detection, providing the bounding boxes for target road users, including pedestrians, cars, cyclists, e-scooterists, and other micromobility vehicles such as bicycles and e-scooters. Then, machine learning regressors are used, taking the features of bounding boxes from YOLOv7 as input, to estimate the positions of the target objects. Finally, a threat assessment algorithm, based on the range and time gap between the ego vehicle and the target object, is used to identify whether the scenario in the video footage is a critical event or not. The object detection model can generate detection results for pedestrians, cars, cyclists, e-scooterists, bicycles, and e-scooters with a mean Average Precision (mAP) of 0.893 for all classes combined. Among all the selected machine learning regressors for position estimation, the RandomForestRegressor has the highest R^2 score, exceeding 0.9. With the positional information of the road users, the threat assessment

algorithm can apply safety metrics to detect critical events.

Keywords: VRU, Micromobility, Computer Vision, Position Estimation, Threat Assessment

Acknowledgements

We would like to extend our heartfelt thanks to our examiner, Marco Dozza, for his time and effort in evaluating our thesis. His critical assessment and thoughtful comments have significantly contributed to improving the quality of this work. We would also like to express our deepest gratitude to our supervisors, Alexander Rasch and Rahul Rajendra Pai, for their invaluable guidance, continuous support, and encouragement throughout the duration of this research. Their expertise and insights have been instrumental in shaping the direction and outcome of this thesis. We are sincerely grateful for their patience, mentorship, and the countless hours they dedicated to reviewing our work and providing constructive feedback.

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

This work was sponsored by the Drive Sweden project "MicroVision: Development, Testing, and Demonstration of a Real-Time Support System for Electric Vehicle Riders" (2023-01047).

We would like to thank Voi Technology AB for offering the e-scooter that enabled the data collection in this thesis. We also thank the team at Chalmers REVERE for lending us the LiDAR equipment for data collection.

Luhan Fang, Yahui Wu, Gothenburg, May 2024

List of Acronyms

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

ADAS	Advanced Driving Assistance System
ADS	Automated Driving System
AP	Average Precision
AEP	Automotive Engineering Project
CNN	Convolutional Neural Network
CSP	Cross Stage Partial
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ETTC	Enhanced Time to Collision
FPN	Feature Pyramid Networks
FOV	Field of View
GPS	Global Positioning System
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
MSE	Mean Squared Error
MAP	Mean Average Precision
MLP	Multilayer Perceptron
NAISS	National Academic Infrastructure for Supercomputing in Sweden
NMS	Non-Maximum Suppression
ROS	Robot Operating System
RTS	Rauch-Tung-Striebel
SCE	Safety Critical Event
SPP	Spatial Pyramid Pooling
TTC	Time to Collision
VRU	Vulnerable Road User
WHO	World Health Organization
YOLO	You Only Look Once

Contents

List of Figures

List of Tables

1
Introduction

Every year, around 1.3 million people are killed globally in traffic-related crashes and between 20 to 50 million people sustain non-fatal injuries according to the World Health Organization (WHO) [1]. In more than half of the annual fatal, groups of people called vulnerable road users (VRUs) [2] are involved. In the context of safety on the road, certain groups mainly include pedestrians, cyclists, and lately, there has been an increased number of e-scooterists on our roads. In short, a road user who belongs to any of these categories is referred to as a vulnerable road user in this thesis. To understand the crashes and mitigate the injuries, it is essential to study the behavior of VRUs and other road users in critical situations. Naturalistic data, which captures real-world interactions and behaviors, is particularly valuable for this purpose. Such data enables researchers to analyze the causation mechanisms of crashes and near-crash events, understand the dynamics of road-user interactions, and develop effective safety measures. Additionally, investigations are underway to explore on how new technologies like computer vision which enables to discover information from the situation represented in a image can be used in real traffic situations to avoid or lower the severity of crashes in the future. Computer vision methods often rely on high quality image dataset, which also states the importance and necessaries of naturalistic data in the traffic context.
This thesis is the continuation of Bharti's master thesis "Estimating road-user position from a camera: a machine learning approach to enable safety applications" [3] and Fang et al.'s automotive engineering project (AEP) [4] "Identifying Critical Interactions Among Road Users from Video Data". Bharti successfully implemented pedestrian detection as well as distance prediction. During the AEP, Fang et al. extended the scope of the models to enable detection of other road users including cyclists, e-scooterists and cars and their position estimation. However, the above models did not enable detection and position estimation of stationary e-scooters. Additionally, using learning-based models to detect e-scooterists and cyclists depended on the collaboration with rule-based models, which used the intersection of the bounding box of a person and a cycle to identify cyclists and e-scooterists. This was based on identifying the pedestrian and then looking for an e-scooter close to the person. Combination of several models increased the overall system size and constrained the potential for real-time detection. Even with the development of neural networks capable of achieving object detection in an end-to-end manner, the model is still highly dependent on the quantity and quality of the dataset. Furthermore, most open-source large-scale object detection datasets lack data on VRUs, such as e-scooterists and cyclists. Therefore, the work of this thesis will focus on forming a dataset that covers VRUs, including e-scooterists and cyclists, as well as some

stationary objects commonly seen in traffic, like e-scooters and bicycles. Moreover, a threat assessment algorithm using the position information of road users will be added to identify safety-critical events (SCE).
Object detection models utilize images collected from monocular camera can only generate 2D detection results. In most of the traffic scenario, 3D detection containing depth information of the target object such as the position is crucial for understanding whether the current scenario is critical to the ego vehicle or not. Learning the position information directly from the 2D object detection model is very difficult. However, there have been several machine learning methods which can utilize the bounding box as features to predict position with the position estimation being regarded as a regression task.
Once the position is estimated, estimating threat assesment can be done to identify potential near crashes and unsafe riding behavior in general.

1.1 Aim of the study

Despite the maturity of object detection based on on-board sensors and associated threat assessment algorithms in the field of autonomous driving, low-cost solutions with a focus on VRUs based on monocular cameras are rarely explored. The aim of this thesis is to develop a system to predict threats based on camera footage with a specific focus on micromobility vehicles and riders by using computer vision to identify different road users and vehicles mentioned above and using machine learning methods to predict threats. The methodology can be applied to naturalistic data to identify near-crash events and other interaction of road user with e-scooterists.
The first goal of this thesis is to collect and label a dataset that includes different VRUs and micromobility vehicles. The second goal is modifying and training object detection models on the expanded dataset. The third goal of the thesis is using machine learning models to estimate the position of the road users based on the bounding box parameters. The fourth goal is applying Kalman filter to filter outliers in the detection. The last goal is using a critical event detection algorithm to identify the near crashes and threats to e-scooter riders using naturalistic data.

1.2 Background Work

1.2.1 Data collection

With the development of computer vision, more and more datasets for object detection has emerged over the past few years. While there are plenty of datasets available for pedestrians and bicycles detection, such as COCO dataset [5], there are few datasets for VRUs such as e-scooterists, cyclists and micromobility vehicles such as e-scooters. Moreover, there is a lack of datasets recording interactions among these VRUs, which can be crucial for understanding crash causation mechanisms. Therefore, in recent years, research targeting this gap have been proposed.
Prabu et al. [6] proposed a wearable data collection system, integrating LiDAR, cameras, and GPS, operating through the Robot Operating System (ROS), to address

limitations related to fixed collection methods and to capture e-scooterists' behaviors. Bharti [3] introduced a pedestrians dataset by utilizing an e-scooter equipped with a monocular fish-eye camera, LiDAR, and a data logger based on a Raspberry Pi board. This setup was employed to capture pedestrians walking in front of the camera, while the LiDAR generated point clouds containing kinematic information used as the ground truth for position estimation. In the AEP [4], the e-scooter used for data collection was equipped in the same manner as the one utilized in Bharti's thesis [3]. The range of target objects to be detected was expanded to include cars, cyclists, e-scooterists in addition to pedestrians. Moreover, García-Venegas et al. [7] introduced a cyclists datasets and a Convolutional Neural Network (CNN) model which can not only detect the cyclists but also identify their orientation.

1.2.2 Road Users Detection

The two-stage object detection algorithms like R-CNN [8], Fast R-CNN [9], and Faster R-CNN [10] can generally reach a higher accuracy on detecting objects at the cost of sacrificing the detection speed. With the introduction of one-stage object detection like You Only Look Once (YOLO), the detection task can be completed faster, which is more suitable for detecting object in real-time.
The e-scooterists detection algorithm in naturalistic scenes proposed by Apurv et al. [11] was achieved by combining YOLOv3 with MobileNetV2. They used YOLOv3 pre-trained on the MS COCO dataset to detect persons, then extended the bounding box to encompass potential e-scooter regions beneath the feet, feeding this information into MobileNetV2 for binary classification to identify pedestrians and e-scooterists. Hoa Nguyen et al. [12] introduced an e-scooter and its rider detection framework where the detection is implemented by using a transfer learning strategy to train a pre-trained YOLOv3 model on the collected dataset having e-scooter and its rider. To improve the e-scooterist detection method based on YOLO collaborating with another binary classification model proposed by Apurv et al., especially in scenarios where the e-scooterist is occluded by other objects, Shane Gilroy et al [13]. introduced an algorithm to enhance the calculation of the bounding box extension, specifying two algorithms based on the aspect ratio for extending the bounding box height for e-scooterists, stretching it to 5 times its original size if occluded (aspect ratio <= 1:2.5), and 1.5 times the original height if not occluded (aspect ratio > 1:2.5). In the AEP [4], the YOLOv7 model collaborates with a series of rules based on the Intersection over Union (IOU) between the detected person and bicycle, as well as the frame information extracted from the JSON file which stores the detection results generated by YOLOv7, to determine whether a person appearing in one frame with a bicycle should be detected as a cyclist or not.

2
Theory

This chapter will present and explain the different theoretical aspects used during this thesis.

2.1 Deep learning for object detection

2.1.1 Convolutional neural networks

Convolutional neural network (CNN) is a kind of deep learning model that is designed to process and analyze visual data. The typical structure of a CNN includes convolutional layers, pooling layers, and fully connected layers. Due to their ability to learn complex features as well as make accurate classifications, they are widely used in the field of computer vision.

2.1.2 YOLO

In this thesis, version 7 of YOLO i.e. YOLOv7 [14] model trained on customized data is used to achieve object detection for different road users. YOLOv7 is a state-of-the-art object detection algorithm. YOLOv7 introduces several improvements and optimizations over its predecessors, which make it accurate and well-suited for real-time applications. It employs a powerful backbone network, such as CSPDarknet53 [15] to extract high-level features from input images efficiently. It also utilizes a multi-scale detection strategy to detect objects at different resolutions within a single network architecture, which improves the detection of objects at various sizes and scales. The following is the network structure of YOLOv7:

1. Backbone: YOLOv7 utilizes CSPDarknet53 as its backbone network, which is an improved version of the Darknet series. The CSP (Cross Stage Partial) structure integrates cross-stage partial connections into Darknet to enhance feature propagation efficiency and network performance.
2. Neck: YOLOv7 employs SPP (Spatial Pyramid Pooling) structure as its neck component. SPP allows the network to capture contextual information of targets at different scales, thereby improving detection performance.
3. Head: The detection head of YOLOv7 consists of multiple Feature Pyramid Networks (FPN) and multilevel predictions. FPN is used to extract feature maps of different scales from the backbone network, while multilevel predictions are utilized to predict bounding boxes, confidence, and class information at different scales.

4. Anchor Boxes: YOLOv7 uses predefined anchor boxes, which are employed to detect targets at different scales. YOLOv7 adjusts these anchor boxes through regression to adapt to targets of different scales and shapes.
5. Output: The output of YOLOv7 consists of a set of bounding boxes, each associated with a target and containing class information and confidence scores. These bounding boxes undergo non-maximum suppression (NMS) to remove highly overlapped boxes, generating the final detection results.

The training process of YOLOv7 is designed to be easily customizable and adaptable so that it is possible to train the model with the dataset collected from the naturalistic data, making the model more adaptable to real-world usage scenarios.

2.1.3 Evaluation metric

To evaluate the performance of the object detection model, mAP (mean Average Precision) is employed. It provides a comprehensive measure of how well an object detection model identifies objects in an image across different categories.

$$\text{mAP} = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{2.1}$$

where C is the number of categories, AP is the average precision of each category. To compute AP, the first step is to create a precision-recall curve. Next, the precision-recall curve is interpolated to create a smooth curve. Finally, the area under this interpolated precision-recall curve is computed, which gives the AP. Higher mAP values indicate better detection performance, with 1.0 being perfect precision and recall for all classes.

In object detection tasks, mAP is often combined with IoU (Intersection over Union), which measures the overlap between the predicted bounding box and the ground truth bounding box. For example, mAP@0.5 means that mAP is calculated under the condition that a detection is considered correct if the IoU between the predicted bounding box and the ground truth bounding box is at least 50%.

2.2 Machine learning models

The distance estimation is a regression task which predicts the position of the target relative to the ego vehicle in the real world based on the geometric features of the bounding box from the object detection model. The following regression models are examples that were used in this thesis.

1. Linear Regression: It is one of the simplest and most widely used regression algorithms. It models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. In simple linear regression, there is only one independent variable, while in multiple linear regression, there are multiple independent variables. The goal of linear regression is to find the best-fitting line that minimizes the vertical distances between the observed data points and the line (the residual errors). This line can then be used to make predictions for new data points. Linear regression is computationally efficient, interpretable, and provides insights

into the relationships between variables, but it assumes a linear relationship between the independent and dependent variables, which may not always be the case.[16]

2. Decision Tree Regressor: It is a type of supervised learning algorithm used for regression tasks. It works by partitioning the feature space into a set of non-overlapping regions and fitting a simple model (like a constant) to each region. The model predicts the target variable by traversing the tree from the root to a leaf node, where each internal node represents a decision based on the value of a feature, and each leaf node represents the predicted value. Decision trees are intuitive, easy to interpret, and can handle both numerical and categorical data. However, they are prone to overfitting, especially when the tree depth is not appropriately controlled.[17]
3. Random Forest Regressor: It is an ensemble learning method used for regression tasks. It operates by constructing multiple decision trees during training and outputs the mean prediction of the individual trees for regression. Each tree in the forest is built using a random subset of the training data and a random subset of the features, which helps to reduce overfitting and improve generalization performance. Random forests are highly robust to noise and outliers in the data and can handle high-dimensional feature spaces effectively.[18]
4. K-Neighbors Regressor: It is a non-parametric algorithm used for regression tasks. It predicts the target variable for a new data point by averaging the target values of the k-nearest neighbors, where "k" is a user-defined parameter. The prediction is typically the mean (or weighted mean) of the target values of the k-nearest neighbors in the feature space. KNN regression is simple to implement and can capture complex patterns in the data. However, it suffers from high computational costs during prediction, especially for large datasets, and requires careful selection of the hyperparameter k.[19]
5. Multi-layer perceptron (MLP) Regressor: It is a type of artificial neural network used for regression tasks. It consists of multiple layers of nodes (neurons), including an input layer, one or more hidden layers, and an output layer. Each node in the hidden layers uses a nonlinear activation function to transform its input. During training, MLP regressors use backpropagation and gradient descent to minimize a loss function and adjust the weights of connections between neurons. MLPs can capture complex relationships in the data but may require careful tuning of hyperparameters like the number of layers, number of neurons per layer, and learning rate.[20]

2.3 Machine learning model evaluation metrics

2.3.1 Mean Absolute Error

Mean Absolute Error (MAE) is a metric used to evaluate the accuracy of a prediction model, particularly in the context of regression analysis. It measures the average

absolute difference between the predicted values and the actual values in a dataset.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the corresponding predicted value.

2.3.2 Mean Squared Error

Mean Squared Error (MSE) is another common metric used in regression analysis to evaluate the performance of a prediction model. Similar to MAE, it quantifies the discrepancy between the predicted values and the actual values in a dataset. However, MSE emphasizes larger errors more than smaller ones, due to the squaring operation.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.3}$$

where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the corresponding predicted value.

2.3.3 R-squared score

The R-squared (R^2) score, also known as the coefficient of determination, is a statistical measure used to assess how well a regression model fits the observed data. It provides insight into the proportion of the variance in the dependent variable (the variable being predicted) that is explained by the independent variables (the predictors) in the model. It ranges from 0 to 1, where a higher value indicates a better explanatory power of the model.

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.4}$$

where n is the number of samples, y_i is the actual value, $\hat{y}_i$ is the corresponding predicted value, and $\bar{y}$ is the mean of the actual values.

2.4 Filtering and outlier rejection algorithm

Due to the limited training data of the position estimation model and the complex environment in the naturalistic data, it is likely to have some errors in its estimation of road user positions. The use of filters outlier rejection algorithm allows correction for errors in position estimation at a certain degree. In addition, the filter can also act as an estimator, which, in combination with the process equations, enables the estimation of the velocity of the target based on its position.

2.4.1 Kalman filter

Kalman filter[21] is presented as a recursive algorithm designed to estimate the state of a linear dynamic system in the presence of noise, which has been used in numerous fields including control theory, signal processing, and navigation systems.
The Kalman filter operates through a two-step process: predict and update. In the prediction step, the algorithm forecasts the state of the system based on its previous state and any known control inputs. The update step then refines this prediction using noisy measurements, producing an optimal estimate of the current state by combining information from both the system dynamics and the measurement data so that an optimal estimation can be given.
The following is the mathematical expression of Kalman filter:

$$\begin{aligned}
\text{Predict:} \quad & \hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \\
& P_k^- = AP_{k-1}A^T + Q_k \\
\text{Update:} \quad & K_k = P_k^- H^T (HP_k^- H^T + R_k)^{-1} \\
& \hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\
& P_k = (I - K_k H)P_k^-
\end{aligned}$$

Where:

- $\hat{x}_k^-$ is the predicted state estimate at time step k.
- A is the state transition matrix.
- $\hat{x}_{k-1}$ is the state estimate at the previous time step $k-1$.
- B is the control-input matrix.
- u_{k-1} is the control input applied at time step $k-1$.
- P_k^- is the predicted error covariance matrix at time step k.
- Q_k is the process noise covariance matrix.
- K_k is the Kalman gain at time step k.
- H is the measurement matrix.
- R_k is the measurement noise covariance matrix.
- z_k is the measurement at time step k.
- $\hat{x}_k$ is the updated state estimate at time step k.
- P_k is the updated error covariance matrix at time step k.
- I is the identity matrix.

2.4.2 Rauch–Tung–Striebel smoother

The Rauch–Tung–Striebel (RTS) smoother [22] is a statistical method used in state-space modeling and signal processing. It is employed to estimate the past states of a dynamic system given noisy observations of its current and past states. RTS smoother is particularly useful when dealing with time series data and is an extension of the Kalman filter.
The RTS smoother works by incorporating information from future observations to refine the estimates of past states, taking advantage of the entire dataset available up to the current time point. By considering both past and future observations, the

RTS smoother provides improved estimates compared to the Kalman filter alone, especially in scenarios where observations are noisy or incomplete.
The mathematical expression of RTS smoother:

$$\hat{x}_{k|K} = \hat{x}_{k|k} + K_k(x_{k+1|K} - \hat{x}_{k+1|K}) \tag{2.5}$$

where $\hat{x}_{k|K}$ is the smoothed state estimate at time k, $\hat{x}_{k|k}$ is the filtered state estimate at time k, $x_{k+1|K}$ is the predicted state estimate at time $k+1$, and K_k is the smoothing gain at time k.

2.4.3 DBSCAN for outlier rejection

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [23] is a popular clustering algorithm in data mining and machine learning. The key idea behind DBSCAN is to group together closely packed points based on two parameters: epsilon (ϵ), which defines the radius within which to search for neighboring points, and min-samples, the minimum number of points required to form a dense region (a cluster).
DBSCAN categorizes points into three types:

1. Core points: These are points that have at leastmin-samples points (including themselves) within distance ϵ.
2. Border points: These points are within distance ϵ of a core point but do not have enough neighbors to be considered core points themselves.
3. Noise points (or outliers): Points that are neither core points nor border points.

The algorithm works by starting with an arbitrary point and expanding the cluster by adding neighboring points if they are core points. This process continues recursively until all reachable points are added to the cluster. Then, the algorithm moves to the next point that has not been assigned to any cluster and repeats the process until all points are either assigned to a cluster or marked as outlier.

2.5 Threat assessment algorithm

Typically, the naturalistic data is used to visually confirm the crashes or near crashes identified by the sensors installed in the vehicle. However, with the kinematic information derived from the position estimation model, it is possible to perform threat assessment on the naturalistic data in the domain of time, distance or a combination of both. The following are the potential threat assessment algorithms that could be used.

2.5.1 Range

Range is also called distance headway in SAE J2944 standard [24]. Distance headway, as defined in J2944, refers to the space or gap between vehicles traveling in the same lane, measured in terms of distance. This is a critical parameter in active safety systems, as it determines the safe following distance between vehicles.

2.5.2 Time gap

In SAE J2944, the time gap is typically expressed in seconds and represents the time interval between ego vehicle and the lead vehicle. It can be calculated according to the equation:

$$TG = \frac{D}{V_e} \tag{2.6}$$

where D is the distance between the same features of the two vehicles, for example the distance between the front bumpers of two cars, V_e is the speed of ego vehicle.

2.5.3 Time to collision (TTC)

TTC is a critical parameter measured and analyzed in evaluating the effectiveness of collision avoidance systems within ADAS and ADS. Time to Collision refers to the estimated time it would take for a vehicle to collide with an obstacle or another vehicle if no corrective action is taken. Compared to the time gap, the TTC takes the speed of the leading vehicle into account:

$$TTC = \frac{D}{V_e - V_t} \tag{2.7}$$

where D is the distance between the same features of the two vehicles, V_e is the speed of ego vehicle, V_t is the speed of leading vehicle.

3
Methods

This chapter will present the methodology used in this thesis. Figure 3.1 shows the overall methodology, and each section is described in detail below.

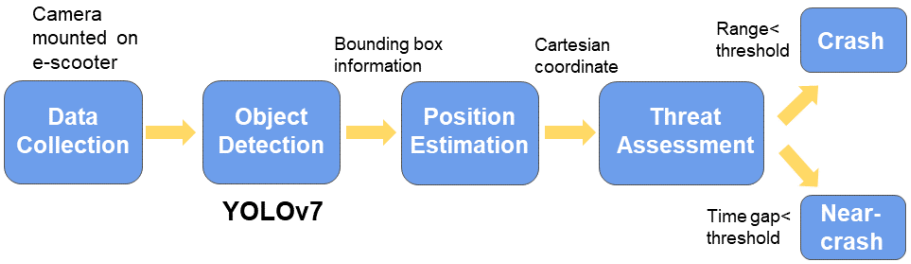
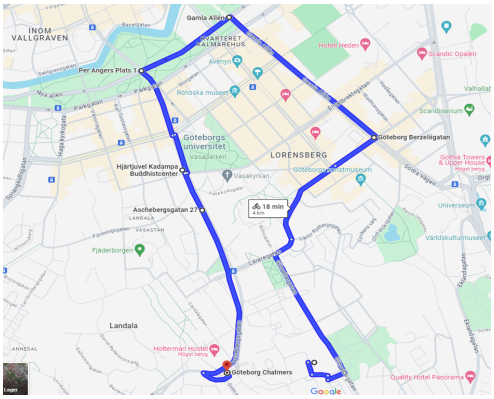


Figure 3.1: Schematic of the methodology pipeline.

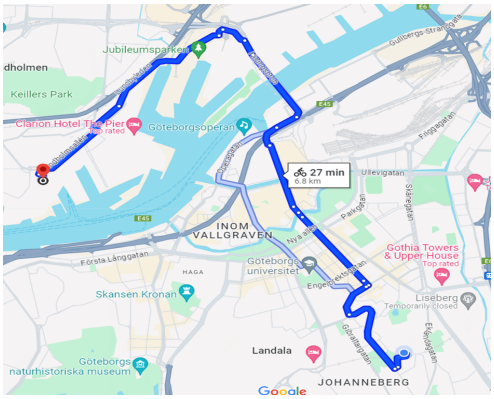
3.1 Data collection

The image data used to train the YOLOv7 model is collected by a monocular fish-eye camera which is the same camera used in Bharti's thesis [3] and AEP [4] with a resolution of 720*532, a field of view of 220° and a frame rate of 30 FPS.
The data was collected at multiple sites in Gothenburg, in order to ensure that the dataset covers as wide a range of traffic situations as possible. Figure 3.2 shows the routes planned to collect data. The first route is mainly along the city's main roads and also passes through the city center area. These areas have a high volume of traffic around 8 am and 4 pm and it's possible to capture a large number of cyclists, e-scooterists and cars.
The second and third routes are two different routes between Lindholmen and Johanneberg. The second route goes through Hisingsbron bridge, where many cyclists as well as some e-scooterists can be captured during rush hours, typically around 8.00 to 9.00 am and 4.00 pm.
The third route also connects the two campus.The e-scooter need to cross the river via a ferry and go through another main road, is a common route preferred by mi-

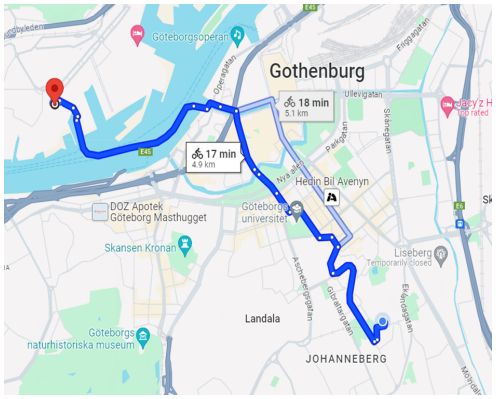
cromobility users.
For stationary objects, such as bicycles and e-scooters, they could easily be found at Chalmers university campuses in Johanneberg and Lindholmen.
To ensure the dataset had the least bias possible, care was taken to balance the number of samples for each category in the dataset.
There are 8083 images, encompassing 16722 road users for collected dataset. It was split into training, validation and test set with a ratio of 8:1:1.



(a) Route around city center.



(b) Route from campus Johanneberg to Lindholmen (via the bridge).



(c) Route from campus Johanneberg to Lindholmen (via the ferry).

Figure 3.2: The three plots show the data collection routes (generated by Google Maps).

Table 3.1: Number of labels of different classes.

Class	Number of labels
Person	4237
Car	4240
Cyclist	2005
Bicycle	1091
E-scooterist	3401
E-scooter	1748

3.2 Object detection and classification

3.2.1 Training data formation and data labelling

The customized training data of YOLOv7 includes images of the target objects and the corresponding annotations that denote the location and class of objects present in the image. There are six categories of objects in total, which include person, car, bicycle, cyclist, e-scooter and e-scooterist.
A standard YOLO annotation includes:

1. Class label: An integer index of the object class.
2. X_center: Normalized X coordinate of the center of the bounding box.
3. Y_center: Normalized Y coordinate of the center of the bounding box.
4. Width: Normalized width of the bounding box relative to the image dimensions.
5. Height: Normalized height of the bounding box relative to the image dimensions.

The training data followed the structure below, each image had a corresponding txt file where its annotation is saved.

```
./data
        ./data/Annotations
                ./data/Annotations/1.txt
                ./data/Annotations/2.txt
                ...
        ./data/images
                ./data/images/1.jpg
                ./data/images/2.jpg
                ...
```

In order to reduce some of the workload, a YOLOv7 model trained on several open-source datasets from the paper of García-Venegas et al. [7] and Roboflow [25] was first used to pre-annotate and get a base weight for further training. The YOLOv7 model combined with the pre-trained weight then was run on the collected dataset which includes the six categories as target. The model produced a rough estimate that had low overlap with the ground truth box and even resulted in false positives or false negatives. Therefore, it is necessary to perform post-processing on these rough detection results. Manual review and labeling was performed to proofread as well as correct the initial labels of the rough results from the base model to make the ground truth more accurate. An open-source graphical image annotation tool LabelImg [26] is utilized to manually label objects in images. It provides a simple and intuitive interface for annotating images with bounding boxes around objects of interest, along with labels specifying the class of each object. With this tool, the correction towards the rough detection can be simply done by dragging or redrawing the bounding box on the edited image. The entire process can be seen as a closed-loop annotation system with continuously incoming new datasets, as shown in Figure 3.3. Figure 3.4 illustrates some examples of the data collected and processed by the auto-labeling loop.

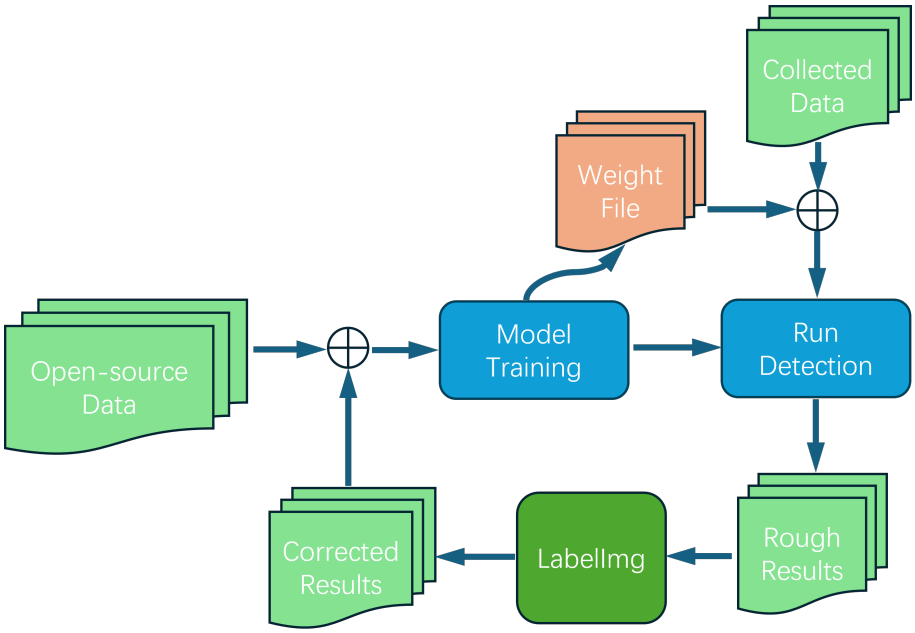


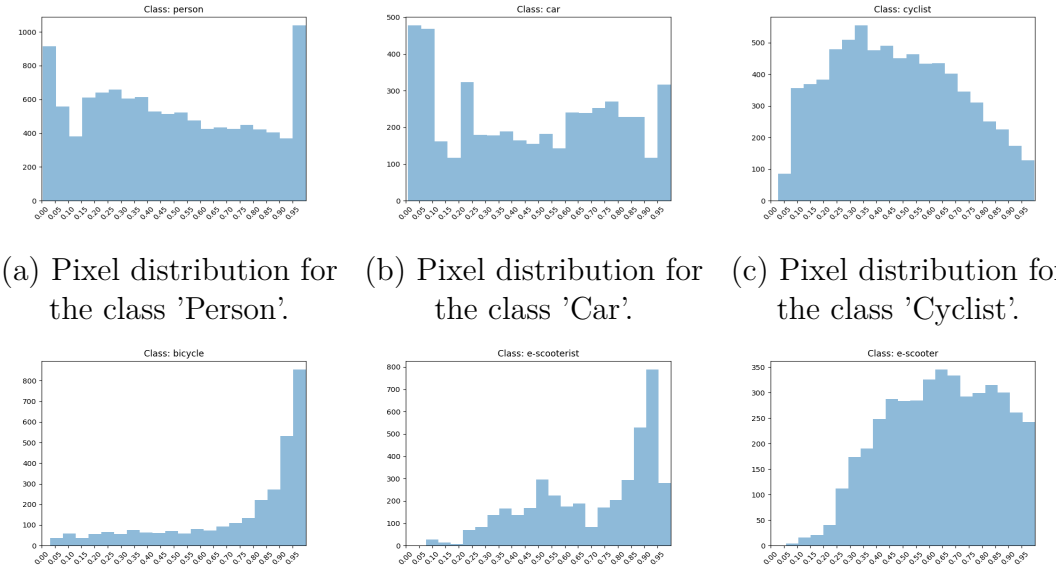
Figure 3.3: Auto-labeling loop.

Figure 3.4: Examples of labeled data

3.2.2 Object detection model training

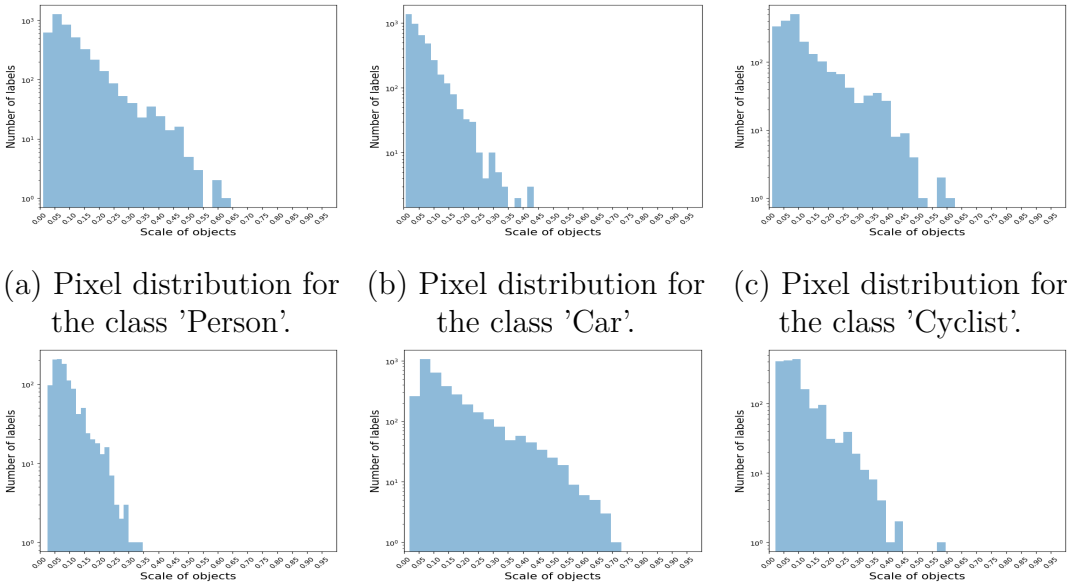
To ensure that the base model trained on the open-source dataset [25] can provide relatively accurate detection outcomes that we can rely on to label our collected

data, the dataset needs to have a distribution similar to that of the collected data. The distribution here were defined as the pixel count of the target object normalized by the total pixels in the image. From Figure 3.5, it can be seen that except for person and cars, the distributions of the other four object classes show a noticeable difference compared to the data collected from everyday driving—larger objects, which occupy more pixels, appear more frequently. In real traffic scenarios captured by the onboard camera, objects typically occupy less than 0.5, or even lower proportions of the camera frame as shown in Figure 3.6.
To minimize the impact of this difference, we used training parameters with an image size of 128 pixels during the training on open-source dataset to prevent the model from performing poorly on the collected dataset, and 640 pixels as image size for collected dataset.
The training and testing were all completed on an NVIDIA T4 GPU. The pre-trained model was not only used to auto-label the collected dataset but also to accelerate the training process. The model was trained for 100 epochs, with the first 3 epochs reserved for warm-up. The Adam optimizer was employed for training, with an initial learning rate of 1e-3, and the final learning rate dropped to 0.1 of the initial learning rate.



(a) Pixel distribution for the class 'Person'.
(b) Pixel distribution for the class 'Car'.
(c) Pixel distribution for the class 'Cyclist'.
(e) Pixel distribution for the class 'Bicycle'.
(f) Pixel distribution for the class 'E-scooterist'.
(g) Pixel distribution for the class 'E-scooter'.

Figure 3.5: Analysis of pixel distribution for the open-source data.



(a) Pixel distribution for the class 'Person'.

(b) Pixel distribution for the class 'Car'.

(c) Pixel distribution for the class 'Cyclist'.

(e) Pixel distribution for the class 'Bicycle'.

(f) Pixel distribution for the class 'E-scooterist'.

(g) Pixel distribution for the class 'E-scooter'.

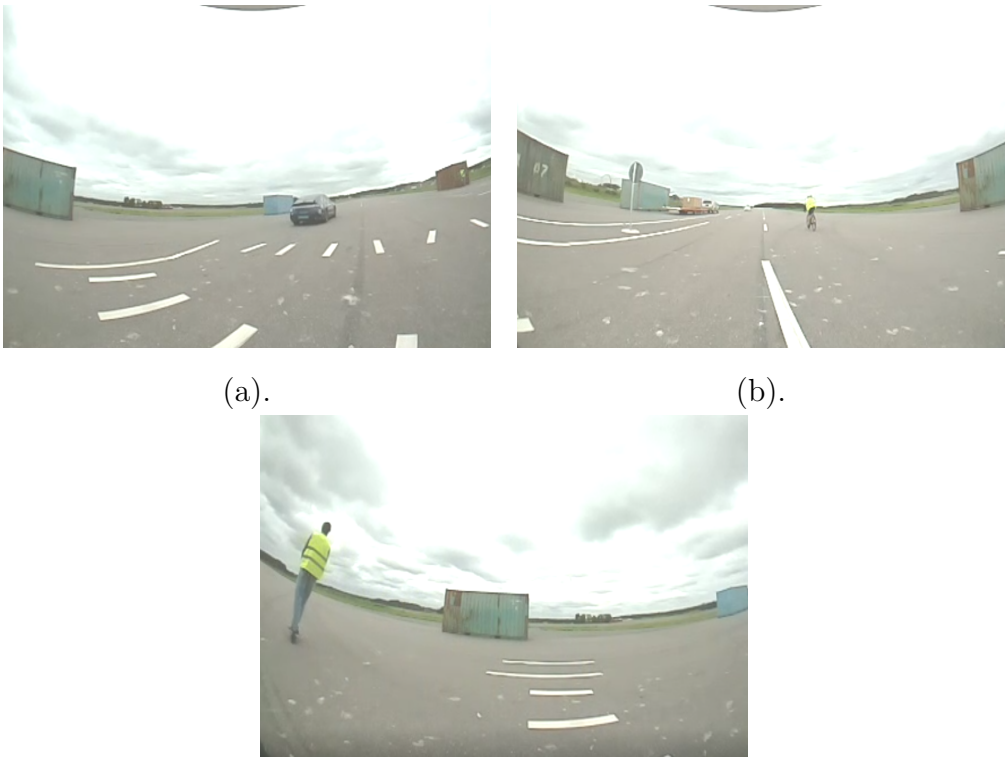
Figure 3.6: Analysis of pixel distribution for the data collected from real driving.

3.3 Position estimation

After using the object detection model on the naturalistic data, the position of the bounding box as well as the size information of each object can be obtained. There is a relationship between the size and position of an object's bounding box and the relative position between that object and the e-scooter. Combined with the ground truth position data collected by LiDAR, the relationship between the two can be fitted by a machine learning algorithm, thus realizing the prediction of the object's position.

3.3.1 Ground truth data collection

The ground truth for the position used as input to the machine learning algorithms was obtained from a LiDAR mounted on the e-scooter. A VLP16 Velodyne LiDAR with vertical FOV of 30 degrees, horizontal FOV of 360 degrees is utilized to extract the position information of pedestrians, cars, e-scooters, bikes, cyclists and e-scooterists. The data collection was completed at Vårgårda airfield using a stationary e-scooter, positioned at different angles, to record other road users moving around it.



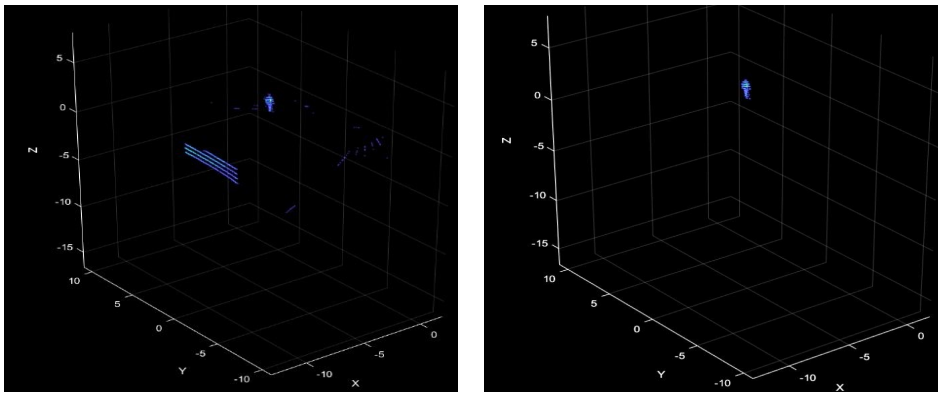
(a). (b).

(c).

Figure 3.7: Ground truth data collection.

3.3.2 Ground truth position extraction

When processing the LiDAR point cloud data, a tracking algorithm is used to filter all the point cloud data except the target object in order to obtain the position of the target object clearly. The following figures show how the objects can be tracked:



(a) Point cloud map before applying DBSCAN to remove noise.

(b) Tracked object point cloud map after applying DBSCAN.

Figure 3.8: The two plots show how to track traffic participants.

The object to be tracked can be specified by inputing the approximate coordinates of the object's centroid of point cloud. To ensure the accuracy of the tracking, a threshold is set for the object's moving speed, and once the object moves faster than this threshold, the tracking will be stopped. Finally, by processing 22 LiDAR point cloud files, the X position, Y position, distance and angle of the tracked objects were saved in different CSV files, which were used for future training and trajectory prediction.

3.3.3 Extracting bounding box information from videos

After the video data is processed by the object detection model, the results of detection are stored in a JSON file. This file includes details for each camera frame, such as the frame number, GPS time, and uptime, along with information about the detected objects like bounding box (BBox), object type, and object ID. Utilizing the object ID as an index, one can extract relevant details such as frame number, GPS time, uptime, and BBox information. The BBox information consists of the x and y coordinates of the top-left point, as well as its height and width. These BBox details serve as input features for the position estimation model. Additionally, the GPS time denotes when the target object appears in the video, aiding in synchronizing camera and LiDAR data.

3.3.4 Synchronizing LiDAR and camera data

Then the ground truth position information can be matched with the bounding box information according to the timestamp. However, the camera has a frame rate of 30 Hz, while the frame rate of LiDAR is 10 Hz. Therefore, it's necessary to do interpolation on the LiDAR frames to generate more data points to match the camera frames. Then the LiDAR data and camera data can be synchronized by the timestamp and generate training data in the format shown in Table 3.2:

Table 3.2: The format of training data for position estimation model.

Col.1	**Col.2**	**Col.3**	**Col.4**	**Col.5**	**Col.6**	**Col.7**	**Col.8**	**Col.9**	**Col.10**	**Col.11**	**Col.12**
X	Y	Distance	Angle	Low mid x	Low y	Height	Width	Top y	Frame no.	Uptime	File name

3.3.5 Training position estimation models

The training of the position estimation model follows the process below. Firstly, the selection of the machine learning model is performed. Commonly used algorithms like Linear Regression, Decision Tree Regressor, Random Forest Regressor, and MLP Regressor are tested on the training data and the model with best performance should be selected. The next step is to do the input feature selection and choose appropriate output. The available input features for the model are low mid x, low y, height and width of the bounding box, where the width of the bounding box should be excluded. This is because the width of the bounding box is not directly related to the distance from that object to the camera, but also to the orientation of that object. Therefore, using the width as an input parameter may introduce errors

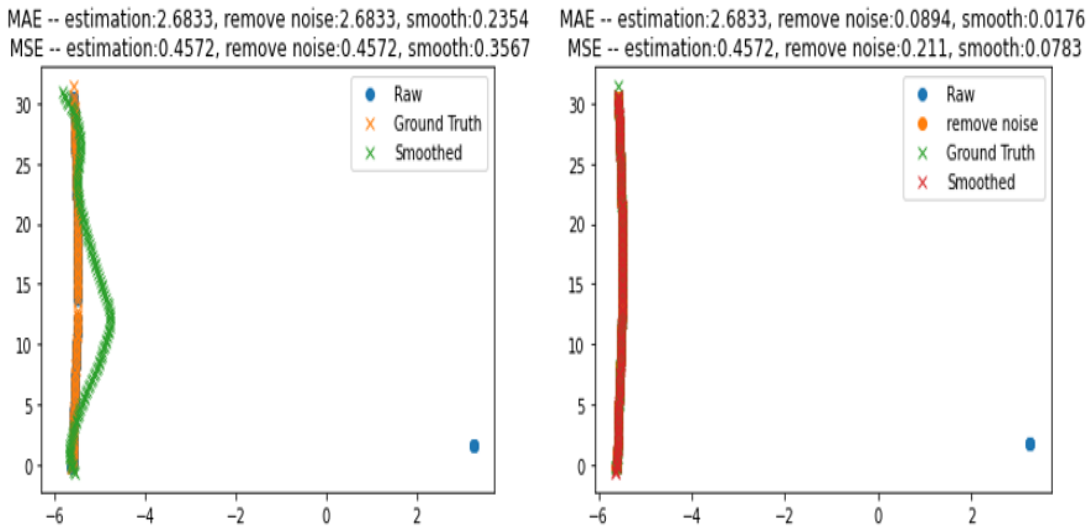
thereby reducing the prediction accuracy. Then, all the models are trained by using low mid x, low y and height as input and polar and Cartesian coordinate as output. The final step is to fine-tuning the models by adjusting their hyperparameters to further improve their performance.

3.4 Post-processing of position estimation

To improve the accuracy of the output of the position estimation model and meanwhile get an estimation of the kinematics information of road users like speed and heading, an outlier removing function based on DBSCAN and filtering algorithms like Kalman filter and RTS could be applied to the raw output of the position estimation model.

3.4.1 Outlier removing algorithm

Due to the limited training data of the position estimation model, the estimation given by the model may deviate a lot from the true trajectory in some cases as shown in Figure 3.9 (a):



(a) Filtered trajectory before applying DBSCAN outlier removing.

(b) Filtered trajectory after applying DBSCAN outlier removing.

Figure 3.9: The estimated trajectory before and after applying DBSCAN.

It can be seen from the figure that there is a raw data point at the lower right corner which is far from other points of the trajectory. Even though RTS is applied to the raw output, the outlier still has a great impact on the estimated trajectory. By contrast, the estimated trajectory improves a lot after applying the outlier removing algorithm, which is almost identical to the ground truth data points as shown in Figure 3.9 (b).
To apply the outlier removal function, it is necessary to tune two hyper-parameters:

eps, denoting the maximum distance between two samples wherein one is considered a neighbor point, and min-samples, representing the minimum count of neighboring points. For eps, a value of 2 is established in line with motion constraints. Determining the min-samples value can be guided by the trajectory's point count (see Table 3.3). To effectively identify outliers, the min-samples value should be set relatively low. Consequently, adjusting the min-samples value in response to changes in n is advisable.

Table 3.3: Selection Rule of min-samples

n (points of trajectory)	$n < 20$	$20 < n < 100$	$n \geq 100$
min-samples	2	$0.1n$	10

3.4.2 RTS smoothing

The first step of RTS smoothing is forward filtering, which is done by employing a Kalman filter to estimate the states of the system based on available measurements up to the current time step. This step provides estimates of the current state as well as the error covariance matrix. Since the frame-per-second of the video is 30, the time interval should be set as $T = 1/30$. It is unlikely that the speeds of road users will change significantly over such a short time interval, making the use of the constant velocity model as process model very appropriate.
As shown in equation (3.1), the state vector of the kinematics information of road users includes the lateral position $x_{k+1}^{p_x}$, the longitudinal position $x_{k+1}^{p_y}$, the lateral velocity $x_{k+1}^{v_x}$, and the longitudinal velocity $x_{k+1}^{v_y}$. The motion noise $\boldsymbol{q}_k$ follows a normal distribution with a zero mean, which can be derived from equation (3.2).

$$\begin{bmatrix} x_{k+1}^{p_x} \\ x_{k+1}^{p_y} \\ x_{k+1}^{v_x} \\ x_{k+1}^{v_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k^{p_x} \\ x_k^{p_y} \\ x_k^{v_x} \\ x_k^{v_y} \end{bmatrix} + \boldsymbol{q}_k, \boldsymbol{q}_k \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}\right) \tag{3.1}$$

$$\begin{aligned} \begin{bmatrix} \dot{x}_{p_x} \\ \dot{x}_{p_y} \\ \dot{x}_{v_x} \\ \dot{x}_{v_y} \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{p_x} \\ x_{p_y} \\ x_{v_x} \\ x_{v_y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{v_x}^2 \\ \sigma_{v_y}^2 \end{bmatrix} \\ \Rightarrow \boldsymbol{q}_k &= T \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{v_x}^2 & 0 \\ 0 & \sigma_{v_y}^2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & T\sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & T\sigma_{v_y}^2 \end{bmatrix} \end{aligned} \tag{3.2}$$

Equation (3.3) displays the measurement model, comprising a 2×4 matrix on the left representing the observation matrix. Only the lateral position $x_k^{p_x}$ and the longitudinal position $x_k^{p_y}$ can be observed. The covariance of the measurement noise $\boldsymbol{r}_k$ conforms to a zero mean normal distribution, constituting the observation covariance. Notably, the variance within the measurement model should exceed that

of the motion model due to inherent noise in the measurement, derived from the output of the position estimation model, thereby causing trajectory irregularities.

$$\begin{bmatrix} y_k^{p_x} \\ y_k^{p_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k^{p_x} \\ x_k^{p_y} \\ x_k^{v_x} \\ x_k^{v_y} \end{bmatrix} + \boldsymbol{r}_k, \boldsymbol{r}_k \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\right) \tag{3.3}$$

After the forward pass Kalman filter traverses all the states, the RTS smoother processes the previously obtained state estimates and error covariances in reverse chronological order, incorporating future information to refine the estimates.

3.5 Critical event identification

After completing the position estimation, the kinematic information of the road users that can be obtained are their Cartesian coordinates and speed. Of these, the speed is obtained from the estimation of the RTS and there is no ground truth speed available to correct the estimation. What's more, errors will accumulate over time, thus the speed suffers from large errors and is unreliable. Next step is the selection of safety metrics. The speed of the road user obtained from the RTS estimation is relative to the speed of the ego vehicle, so the TTC can be obtained by simply dividing the longitudinal distance from the leading vehicle to the ego vehicle by the longitudinal relative speed. However, the use of TTC as a safety metric may cause misdetection and underdetection of critical events due to the inaccuracy of the resulting TTC caused by errors in speed. In contrast, the time gap obtained using the position information of road users and the speed of ego e-scooter is more reliable. But in vehicle following situations the distance to the leading vehicle can be close, which could cause some false detections that are not really critical. In general, the range from road users to ego vehicle and time gap can be potential safety metrics.

3.5.1 Threat assessment by range

Figure 3.10 shows the thresholds for threat assessment, where interaction with other road users is identified to be a critical event when its distance to ego vehicle is less than $0.5 + \alpha * speed$ meter. The value of the threshold in the longitudinal direction is taken in relation to the speed of the ego vehicle, since the distance required for emergency braking or steering of the vehicle increases as the speed increases.The unit of speed here is meters per second. In order to prevent the threshold from varying too much according to speed, the speed should be scaled down. So the parameter α is set to 0.1 in this case. To filter out the situations where other road users are overtaking ego vehicle or there are oncoming vehicles in the adjacent bike lane so that they are identified as critical, the region of interest is set to be within plus or minus 10 degrees directly in front. The algorithm filters out detections when the ego e-scooter speed is less than three meter per second. The filtering is crucial to reduce false positive detections when the ego e-scooter is riding in crowded area at slow speeds.

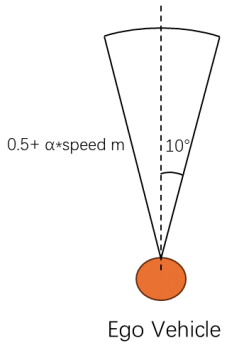


Figure 3.10: The threshold of critical event using range as safety metric.

3.5.2 Threat assessment by time gap

The disadvantage of using range as a safety metric is that it is only detected when the road user is very close to the vehicle, i.e. when the threat is imminent. The introduction of a time gap allows the identification of critical interactions with other road users at some point in the future. Then, the threshold of the time gap should be determined. If the time gap to the road user in the front is less than the time it takes to brake ego vehicle, then the situation is a critical event. The maximum speed of an e-scooter is around $5m/s$ and the maximum braking deceleration is about $-5m/s^2$, which means it takes around 1 second to brake the e-scooter when riding at its maximum speed. Therefore, it is appropriate to choose 1 second as the time gap threshold.

4
Results

4.1 Object detection model

It can be concluded from Figure 4.1 that the average mAP of all six categories reaches 0.893, which indicates a high accuracy on the test data. For cyclists, e-scooters and e-scooterists, the accuracy of detection is particularly high, with mAP values of 0.969, 0.960, and 0.988 respectively. But the mAP values of persons, bicycles and cars are significantly lower, at 0.835, 0.781 and 0.826 respectively. More insights of the performance of the model could be gained through the confusion matrix shown in Figure 4.2. It could be concluded that the background false positive rate of person and car is much higher than other categories, which means other objects in the frame may be falsely detected as persons and cars more frequently. Figure 4.3 shows some false positive examples when applying the model to one of the naturalistic data. It can seen that traffic signs on the side of the road and pedestrian signs on the road are often detected as persons. In addition, rocks in the distance may be mistakenly detected as cars. In terms of false negative rate, the values of person, car and bicycle are obviously higher, which indicates that these categories are more likely to be missed in the detection.

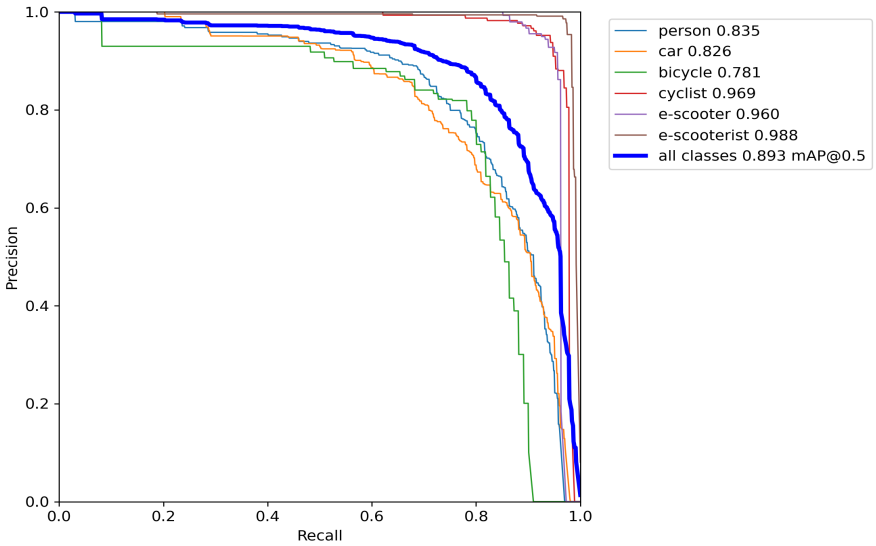


Figure 4.1: PR curve of object detection model

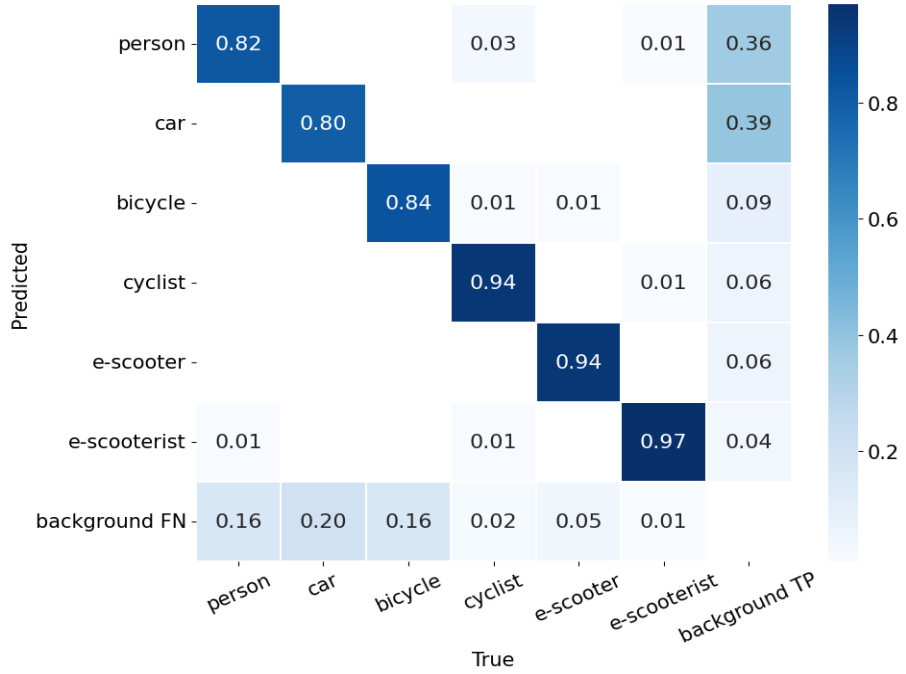


Figure 4.2: Confusion matrix of object detection model

(a). (b).

(c). (d).

Figure 4.3: False positive examples.

4.2 Position estimation model

The model's prediction error is visualized using methods such as heat maps. The following are the training results of position estimation model for different road users.

4.2.1 Position estimation model for cyclists

It can be concluded from Table 4.1 that RandomForestRegressor has the highest R^2 score and meanwhile has the lowest error.

Table 4.1: Comparison of the performance of different regression algorithms for cyclists.

Model	MSE	MAE	R^2 Score
LinearRegression	92.688	4.620	0.595
DecisionTreeRegressor	117.694	2.854	0.835
RandomForestRegressor	65.399	2.667	0.905
MLPRegressor	72.213	3.623	0.827

It is clear from Table 4.2 and Table 4.3 that the input features and output has a great impact on the performance of the model. The model using all three features as input has the highest R^2 score and meanwhile the lowest MSE and MAE. In terms of output, the model predicts x and y coordinates in Cartesian coordinates system is more advantageous in both error level and R^2 score.

Table 4.2: Performance of RandomForestRegressor for cyclists using different parameters.

Parameters	MSE	MAE	R^2 Score
Low mid x,Low mid y	75.411	3.128	0.880
Low mid x,height	76.050	2.924	0.888
Low mid y,height	450.208	9.025	0.409
Low mid x,Low mid y,height	65.399	2.667	0.905

Table 4.3: Performance of RF model for cyclists using polar and Cartesian coordinates.

Algorithms	MSE	MAE	R^2 Score
RF Cartesian	0.953	0.631	0.928
RF Polar	129.846	4.703	0.883

For the random forest model, the hyperparameters that have a large impact on the performance of the model are the number of estimators and the depth of the tree.

Therefore, the effect on the R^2 scores of the prediction results when these two hyperparameters are varied should be explored. It can be concluded from Figure 4.4 that after the number of estimators reaches 20, increasing the number of estimators has a negligible effect on improving the R2 score. So setting *n_estimators* to 20 gives the best accuracy while reducing the amount of computation so that computational efficiency can be increased. In terms of *max_depth*, it is clear from Figure 4.5 that after the maximum depth of the tree reaches 12, the increase in depth has no effect on the R2 score of the model. The *max_depth* should be set to a value around 12 because too much depth may cause the model to overfit the training data thus affecting accuracy.

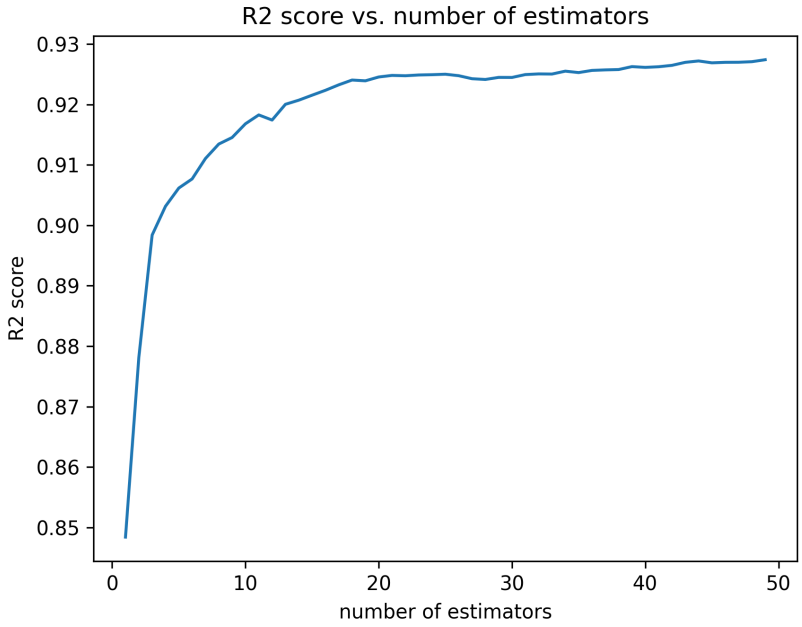


Figure 4.4: RandomForestRegressor hyperparameter optimization. Where *n_estimators* = 20 gives the best performance considering both R^2 score and computational efficiency.

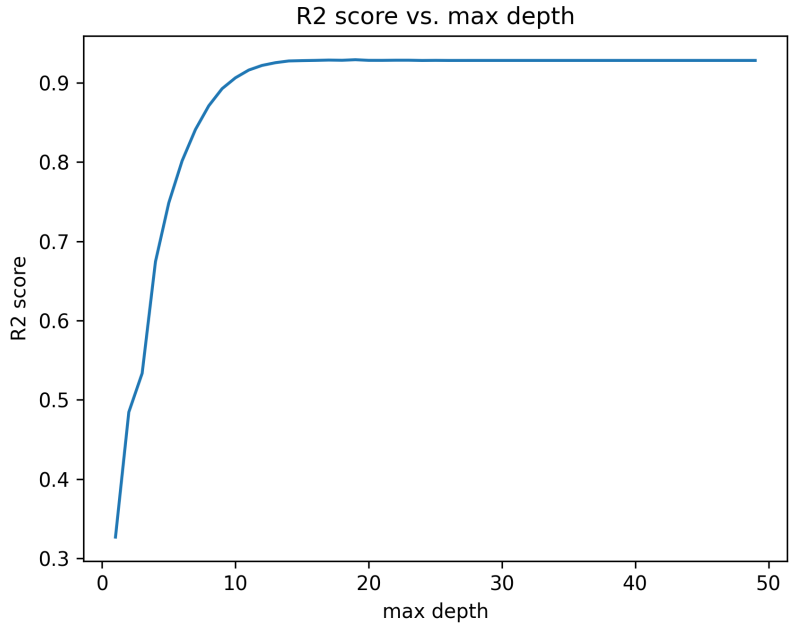


Figure 4.5: RandomForestRegressor hyperparameter optimization. The R^2 score maintains a stable level after *max_depth* reaches 12.

Figure 4.6 shows the actual position and predicted position estimated by the RandomForestRegressor after hyperparameter tuning in Cartesian coordinate system.

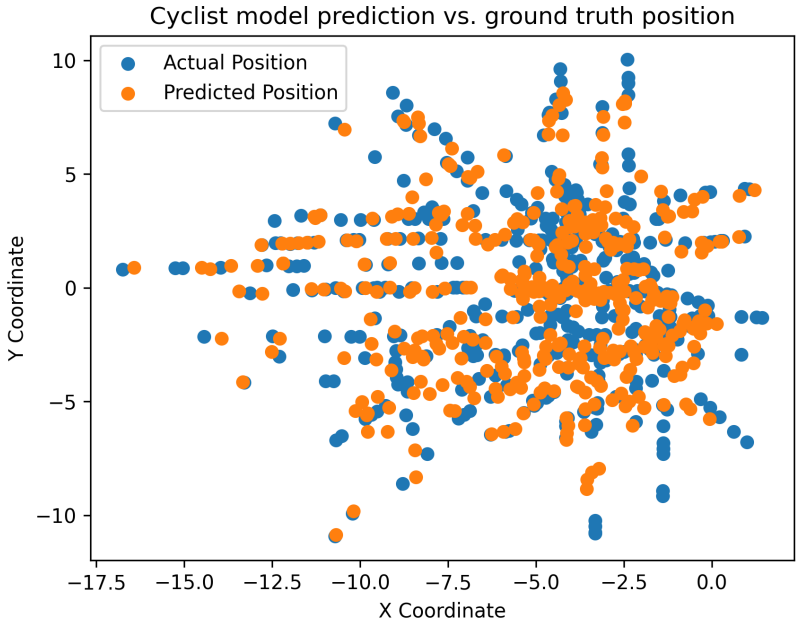


Figure 4.6: Cyclist ground truth data and predicted data using Cartesian coordinates.

In order to have a more intuitive understanding of the magnitude as well as the

distribution of the prediction error, an error heat map was used for visualization as shown in Figure 4.7. It could be concluded that for the bicycle model, the higher errors are mainly concentrated in the positions closer to the ego vehicle and further away around the edge of the frame.

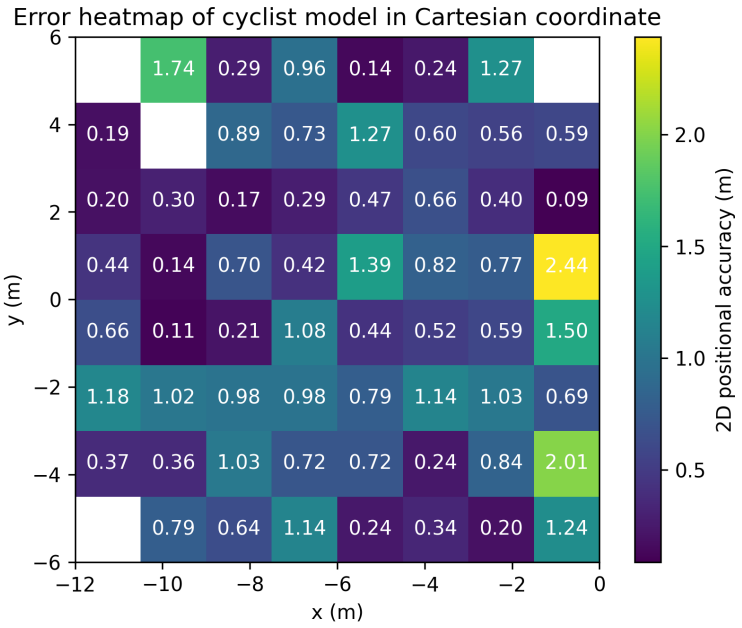


Figure 4.7: Cyclist model error heatmap in Cartesian coordinates.

4.2.2 Position estimation model for e-scooterists

When it comes to position estimation for e-scooterists, RandomForestRegressor still has the best performance among all the models, with a R^2 score of 0.929 and its MSE and MAE are 53.301 and 2.514 respectively.

Table 4.4: Comparison of the performance of different regression algorithms for e-scooterists.

Model	MSE	MAE	R^2 Score
LinearRegression	92.199	4.584	0.634
DecisionTreeRegressor	92.410	2.850	0.873
RandomForestRegressor	53.301	2.514	0.929
MLPRegressor	77.841	3.762	0.844

According to Table 4.5 and Table 4.6, the RandomForestRegressor that uses low mid x, low mid y and height as input and gives predictions in Cartesian coordinates has the highest R^2 score as well as the lowest MAE and MSE.

Table 4.5: Performance of RandomForestRegressor for e-scooterists using different parameters.

Parameters	MSE	MAE	R^2 Score
Low mid x,Low mid y	67.486	2.993	0.909
Low mid x,height	57.521	2.667	0.923
Low mid y,height	442.035	8.520	0.536
Low mid x,Low mid y,height	53.301	2.514	0.929

Table 4.6: Performance of RF model for e-scooterists using polar and Cartesian coordinates.

Algorithms	MSE	MAE	R^2 Score
RF Cartesian	0.709	0.555	0.942
RF Polar	105.893	4.474	0.915

Figure 4.8 and Figure 4.9 shows trend of R^2 score as $n\_estimators$ and max_depth are varied. It could be concluded that as the number of estimators increases, the increase in R^2 score gradually slows down and increases minimally after reaching 20. Also, the R^2 score no longer changes after max_depth reaches 10. So, $n\_estimators = 20$ and $max\_depth = 10$ should be used when training the model to prevent overfitting and improve computational efficiency.

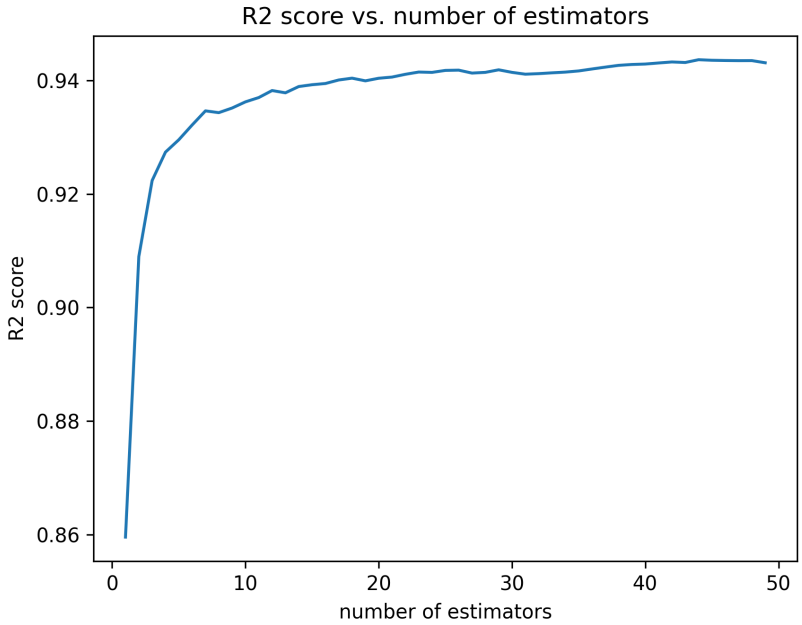


Figure 4.8: RandomForestRegressor hyperparameter optimization. Where $n\_estimators = 20$ gives the best performance considering both R^2 score and computational efficiency.

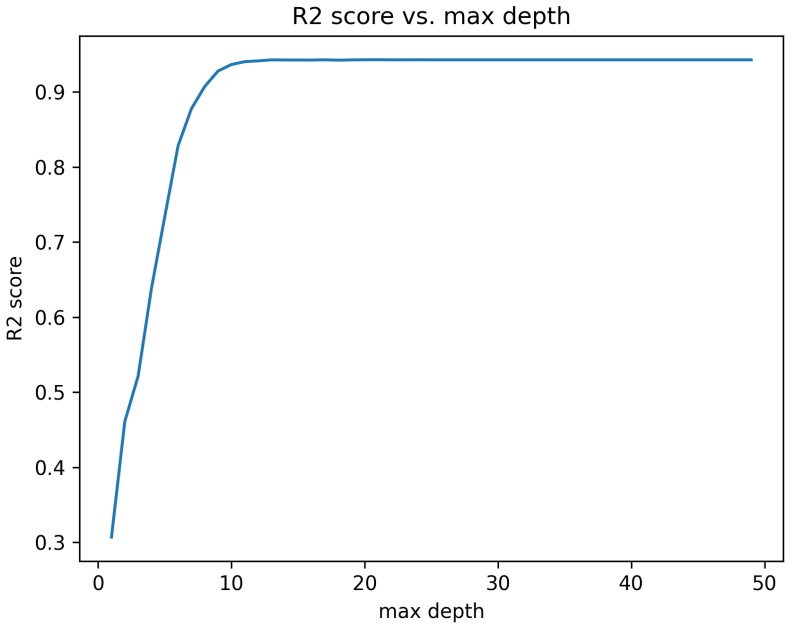


Figure 4.9: RandomForestRegressor hyperparameter optimization. The R^2 score maintains a stable level after *max_depth* reaches 10.

Figure 4.10 shows the actual position and predicted position estimated by the RandomForestRegressor after hyperparameter tuning in Cartesian coordinate system.

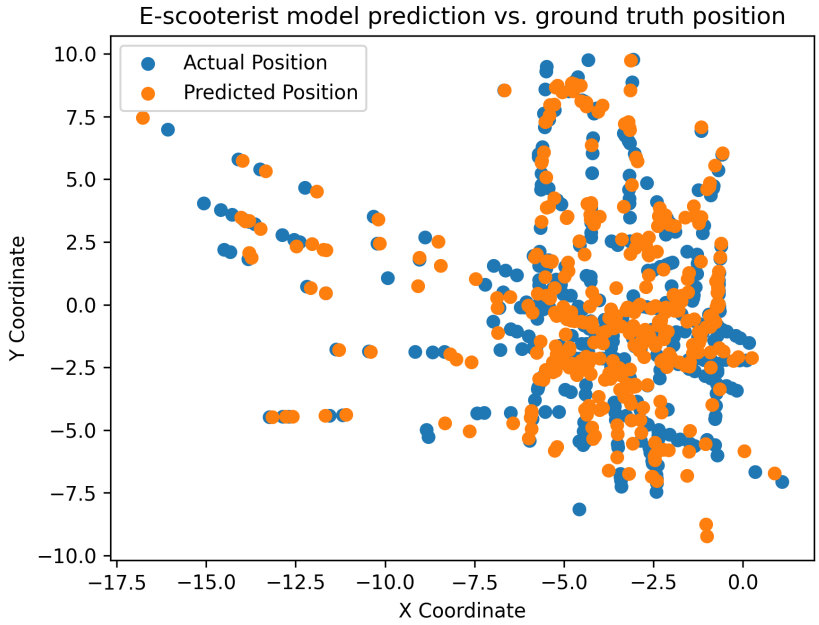


Figure 4.10: E-scooterist ground truth data and predicted data using Cartesian coordinates.

For the e-scooterist model, it is clear from Figure 4.11 the overall error is at a low

level, but it is higher at a distance of about 7 meters from the ego vehicle.

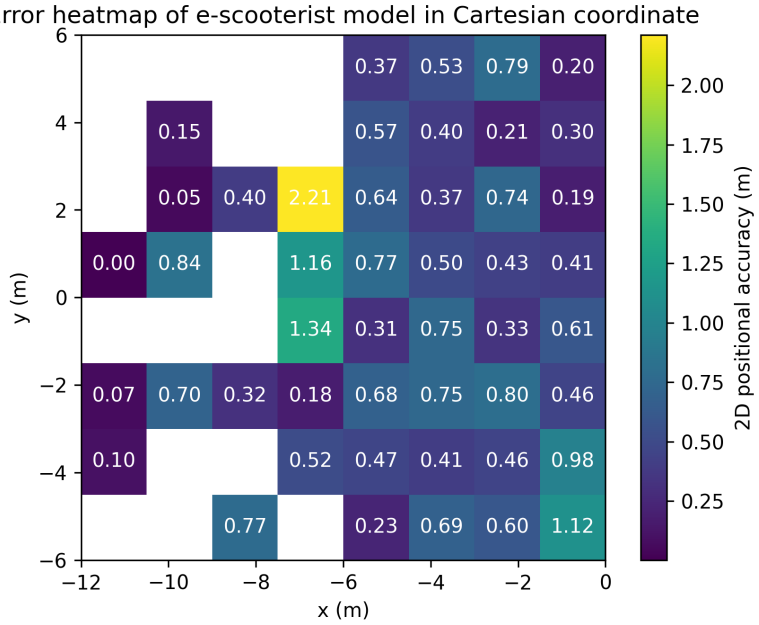


Figure 4.11: E-scooter error heatmap by Cartesian coordinates.

4.2.3 Position estimation model for cars

In this case, it can be seen from Table 4.7 that both RandomForestRegressor and MLPRegressor have very good performance. RandomForestRegressor has a slightly higher R^2 score, while MLPRegressor has lower MSE and MAE values. Therefore, it is necessary to compare these two models when given different combinations of input parameters and make predictions in different coordinate systems.

Table 4.7: Comparison of the performance of different regression algorithms for cars.

Model	MSE	MAE	R^2 Score
LinearRegression	13.492	2.857	0.711
DecisionTreeRegressor	6.605	1.707	0.879
RandomForestRegressor	4.512	1.268	0.965
MLPRegressor	2.984	1.098	0.960

It could be concluded from Table 4.8 and Table 4.9 that both RandomForestRegressor and MLPRegressor perform the best when they use all three parameters as input.

Table 4.8: Performance of RandomForestRegressor for cars using different parameters.

Parameters	MSE	MAE	R^2 Score
Low mid x,Low mid y	35.100	2.799	0.884
Low mid x,height	6.096	1.494	0.947
Low mid y,height	611.704	11.180	0.476
Low mid x,Low mid y,height	4.512	1.268	0.965

Table 4.9: Performance of MLPRegressor for cars using different parameters.

Parameters	MSE	MAE	R^2 Score
Low mid x,Low mid y	20.578	2.669	0.836
Low mid x,height	4.874	1.428	0.941
Low mid y,height	445.319	9.690	0.597
Low mid x,Low mid y,height	2.984	1.098	0.960

Comparing the performance of two models in different coordinate systems as shown in Table 4.10 and Table 4.11, it is clear that the RandomForestRegressor model has the optimal performance using Cartesian coordinates.

Table 4.10: Performance of RF model for cars using polar and Cartesian coordinates.

Algorithms	MSE	MAE	R^2 Score
RF Cartesian	0.762	0.642	0.972
RF Polar	8.261	1.895	0.957

Table 4.11: Performance of MLP model for cars using polar and Cartesian coordinates.

Algorithms	MSE	MAE	R^2 Score
RF Cartesian	0.927	0.756	0.967
RF Polar	5.042	1.441	0.954

Since the RandomForestRegressor has a better performance, its hyperparameters should be further optimized. It could be observed from Figure 4.12 and Figure 4.13 that as $n\_estimators$ increases, the R^2 score shows an overall upward trend, reaching a local maximum when the number reaches 24, followed by a slight decrease and then a small increase. And the R^2 score stabilizes at a fixed level after max_depth exceeds 8. So the optimal hyperparameter combination is $n\_estimators = 24$ and $max\_depth = 8$.

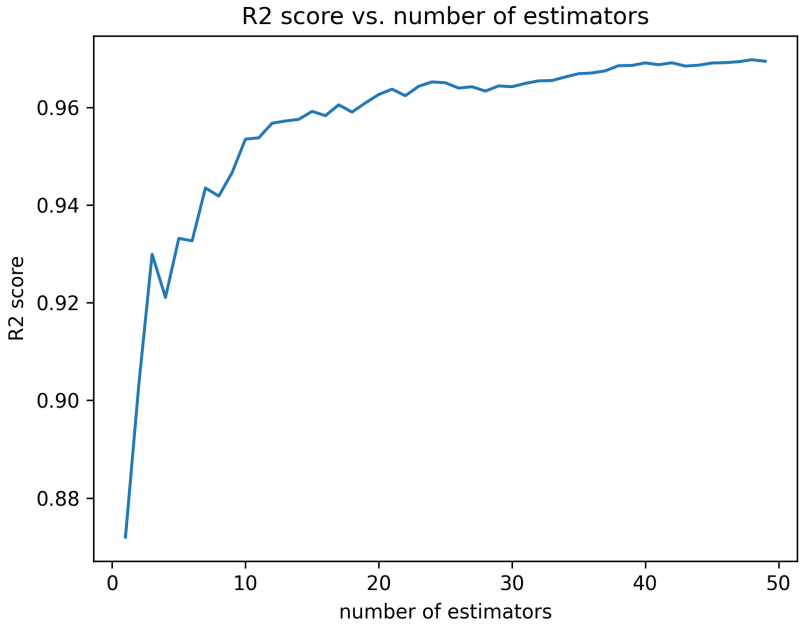


Figure 4.12: RandomForestRegressor hyperparameter optimization. Where $n\_estimators = 24$ gives the best performance considering both R^2 score and computational efficiency.

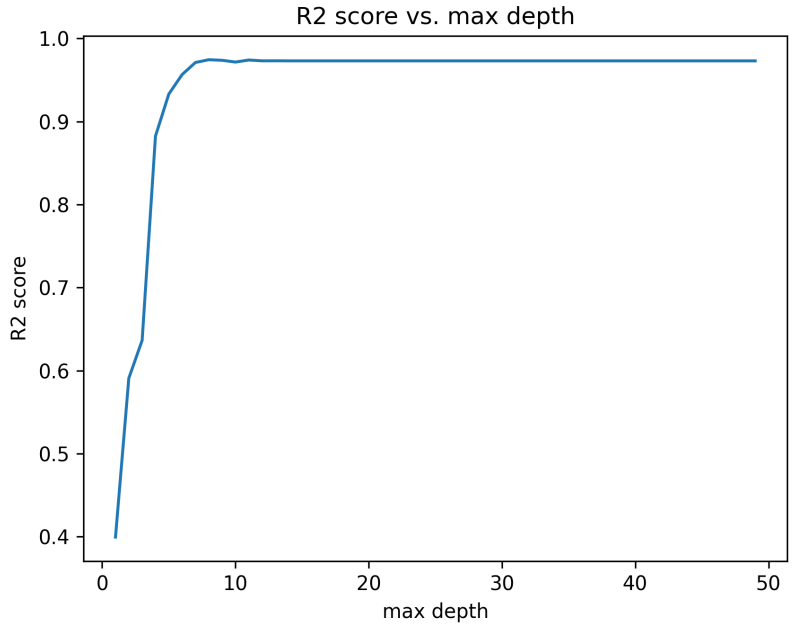


Figure 4.13: RandomForestRegressor hyperparameter optimization. The R^2 score maintains a stable level after max_depth reaches 8.

Figure 4.14 shows the actual position and predicted position estimated by the RandomForestRegressor after hyperparameter tuning in Cartesian coordinate system.

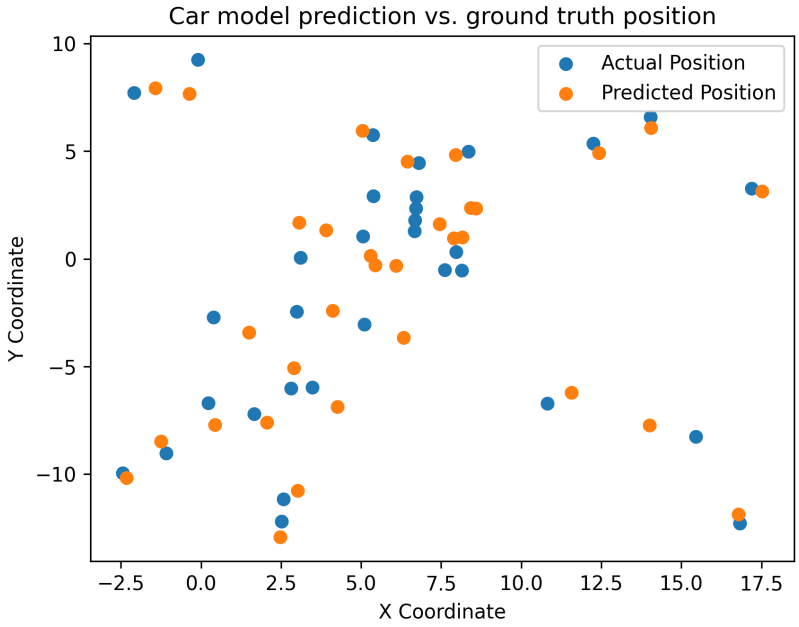


Figure 4.14: Car ground truth data and predicted data using Cartesian coordinates.

It can be concluded from Figure 4.15 that the error level is higher within the area of 4 to 8 meters in longitudinal distance and -2 to 2 meters in lateral distance.

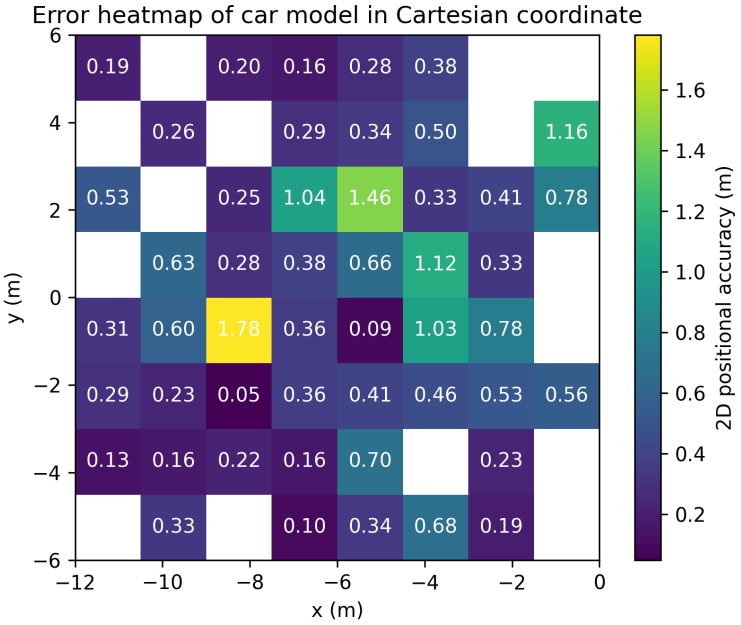


Figure 4.15: Car error heatmap by Cartesian coordinates.

4.3 Threat assessment

Due to the rarity of crashes and near-crash events, there are currently only two naturalistic data segments containing critical events, which can be used to test the threat assessment algorithm. In one naturalistic data video, ego vehicle collided with another e-scooterist, whose object ID is 39. The other one is a near-crash event, where e-scooterist in the front suddenly fell, causing the ego vehicle to brake and steer to avoid the crash. The corresponding ID of the e-scooterist is 87. Table 4.12 shows the IDs of the objects flagged by the threat assessment algorithm in the naturalistic data using different safety metrics and different thresholds. It can be concluded from the table that using range as a safety metric captured the crash very accurately, but failed to capture the near-crash event in the second video. Next, time gap is used as a safety metric, and when the threshold is one second, it can be seen that a lot of objects were flagged in both videos. However, reviewing the videos reveals that most of these objects are flagged because the ego vehicle is following behind them, traveling in the same direction but at a closer distance. In this case, it is not appropriate to call it a critical event. If the threshold is lowered to 0.8 second, there is no change in the flagged objects in the first video, and significantly fewer flagged objects in the second video, but the near-crash event is missed.

Table 4.12: Object ID detected as critical events by different safety metrics in two naturalistic data videos.

Safety Metric	ND1	ND2
Range	39	None
Time Gap (threshold = 1s)	17, 19, 29, 39, 43, 44, 46, 49, 72, 74, 81, 84, 85, 89, 94	24, 25, 27, 56, 66, 76, 77, 78, 80, 87, 89, 131, 132, 133, 136
Time Gap (threshold = 0.8s)	17, 19, 29, 39, 43, 44, 46, 49, 72, 74, 81, 84, 85, 89, 94	24, 25, 27, 56, 78, 77, 80, 89

5
Discussion

5.1 Object detection model

Overall, the performance of the object detection model is satisfactory. Most of the objects in all six categories are accurately detected and tracked at close distances. But when it comes to the case of longer distances, there is more uncertainty in the model's detection. Based on our observations, when pedestrians, cyclists, and e-scooter riders are present at longer distances, the model is likely to confuse these three categories of objects, resulting in a situation where the category of the same object keeps changing. We believe the main reason for this is that the input natural dataset has a low resolution, so that when the objects are far away, the objects are only a few pixel points in size. This leads to too little pixel information thus affecting the model for feature extraction. Another problem is the false detection of pedestrians and cars, which is also related to the resolution of the input video. At low resolution, some objects such as traffic signs on the roadside may have similar characteristics to pedestrians, thus causing false detections, and this is also true for cars. Another possible reason is the lack of diversity in the training data, there are some cases of multiple acquisitions based on the same objects during our data collection, which can have an impact on the generalization ability of the model.

5.2 Position estimation model

The R^2 score, MAE and MSE values of the position estimation model indicate that the model can fit the relationship between the input features and the output very well and the prediction error is at a low level. When the model is tested on the data where the ego e-scooter is stationary, the predictions are usually very accurate. However, when the e-scooter starts moving, the accuracy is slightly affected. But in general, the predictions are accurate enough for the following threat assessment. The whole position estimation pipeline also has some limitations. One limitation is that it is too homogeneous and limited in terms of training data. Each model for all six categories of road users is trained on the data that only include one road user of its category. When training the models for cyclists and e-scooter riders, the height of the bounding box is used as an input parameter, which is related to the rider's height and riding posture. In real traffic environments, cyclists' and e-scooter riders' size and riding posture are very diverse and vary greatly. Therefore, when applying this model to naturalistic data, the differences in riders may introduce some errors. This is also true for the car model, which has training data collected based on the

same car. Another limitation of the model is that it is affected by the dynamics of ego vehicle. As the e-scooter travels down the road, it will tilt or pitch, which will cause the camera to move with it. As a result, the position of objects in the camera frame changes relative to when the e-scooter is stationary. As a result, the prediction will deviate from it actual position because of the changed low mid x and y coordinates. The third limitation is the dependency of the position estimation model on the bounding box generated by the object detection model. The position estimation model is trained by the ground truth position of the object collected by the LiDAR together with the corresponding bounding box given the image of the object. Therefore, this model can only be applied when the same type of camera is used.

5.3 Threat assessment algorithm

According to the results in the previous chapter, it can be concluded that using range as a safety metric can accurately capture crashes, but for near-crash events, it is difficult to use range to make judgments because drivers can anticipate the risk in advance and avoid other road users from coming within the critical range by braking or steering. Employing time gap can capture some potential near-crash events but also introduces some cases where the rider has a lower safety margin. For example, time gap calculated in the case of following a leading vehicle can look more dangerous than they actually are. By contrast, in the case of oncoming vehicle the time gap is longer than the actual time to collision, thus missing some critical events. So the ideal algorithm is to calculate TTC to the vehicle in front, or even ETTC, taking into account all the dynamics of the ego vehicle and other road users. However, this puts a high demand on the prediction of the position estimation model, which is difficult to achieve with the current prediction accuracy. Another issue is the naturalistic data that includes critical events is very limited, which affects the tuning of the thresholds of the threat assessment algorithm. The two critical events currently used to test the algorithm are both interactions with e-scooterists, so the thresholds chosen are more appropriate for this scenario. For other types of road users, this threshold may not be applicable. When sufficient data on critical events are available, tuning the thresholds of the threat assessment algorithm separately for different road users may be a viable strategy.

6
Conclusion

This thesis proposes a detection model to identify common VRUs, including pedestrians, cyclists, and other road users such as cars and stationary bicycles. In addition to these road users, which are commonly seen in many open-source datasets, the proposed model can also detect road users including e-scooterists and e-scooters, which are rarely seen in the current open-source datasets. The entire detection task is completed by training a YOLOv7 model on dataset collected from the experiment e-scooter in daily driving. This dataset covers all six classes of objects mentioned above. The object detection model achieved a high mAP of around 0.9 for all six classes, with especially high mAP for cyclists, e-scooterists and e-scooters, exceeding 0.95.

The position estimation model combines the detection results from YOLOv7 as input features with the position data collected from LiDAR as ground truth. It can predict the position information of the target objects appearing in the camera footage. It is achieved by training and tuning different machine learning regression models, with random forest performing best among the tested models.

For threat assessment, a rule-based model is proposed. The model mainly focuses on detecting rear-end events by using the range which specifies a region of interest to detect crashes and using the time gap between the ego and front vehicle to detect near-crash events. Due to the limited volume of the dataset, more complicated and robust methods, such as detecting critical events by considering the trajectory of road users, are challenging to implement. Therefore, in future work, as the dataset scale increases, the threat assessment can be improved.

7
Future Work

This section shows where the work accomplished in this thesis may be further refined in the future. First of all, there is still a huge room for improvement in the object detection model. The camera we are currently using is a low-resolution fish-eye camera with distortions, which makes the model perform worse than expected in distorted areas and when objects are far away. If a high-resolution camera without distortion can be used in the future, the accuracy of detecting objects at the edges of the frame as well as objects in the distance can be significantly improved. What's more, automatic annotation of naturalistic data videos can be achieved using existing object detection model. A large amount of training data can be obtained by simply checking the annotations generated by the model and correcting them manually if necessary. In the future if more and more diverse video data can be collected, it will enable the model to have better performance in more scenarios.
As mentioned in the previous chapter, the training data is too homogeneous and does not cover the entire field of view of the camera for the position estimation model. Therefore, one direction for future work could be to extend the existing training data and create a set of validation data to verify the performance of the model. Another direction could be to use a deep learning model to perform position estimation using the image as well as bounding box information as inputs instead of a traditional machine learning solution that uses only bounding box information. In addition, eliminating the effect of ego vehicle's dynamics on position estimation is also well worth exploring. But this requires some sensors to obtain the kinematic information of the ego vehicle, such as yaw and pitch angles, and correct them by certain algorithms. Moreover, calibrating the camera to estimate the camera posture can be helpful to eliminate the effect of the ego vehicle's dynamics.
Once a more accurate position estimation algorithm is established, the speed information of road users will become much more reliable. As a result, the methods for conducting threat assessment will become more diverse. If real-time assessment is not pursued, it can be based on the complete trajectory of a road user. It is also possible to detect critical scenarios based on TTC that cannot be detected using only the distance to the vehicle in front.

Bibliography

[1] WHO. *Road traffic injuries*. URL: https://www.who.int/health-topics/road-safety#tab=tab_3.

[2] European Commission. *ITS Vulnerable Road Users*. URL: https://transport.ec.europa.eu/transport-themes/intelligent-transport-systems/road/action-plan-and-directive/its-vulnerable-road-users_en.

[3] Karan Bharti. *Estimating road-user position from a camera: a machine learning approach to enable safety applications*. 2023. URL: https://odr.chalmers.se/items/c5c843b1-773e-4c9d-a8a7-b291ca6614fd.

[4] Luhan Fang et al. *Using machine learning to estimate road-user kinematics from video data*. 2024. URL: https://odr.chalmers.se/items/8be84750-9530-4062-948a-93e93ef7767f.

[5] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in context*. Jan. 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1\{_}48. URL: https://doi.org/10.1007/978-3-319-10602-1_48.

[6] A.V. Prabu et al. "A wearable data collection system for studying Micro-Level E-Scooter behavior in naturalistic road environment". In: *arXiv (Cornell University)* (Dec. 2022). DOI: 10.48550/arxiv.2212.11979. URL: https://arxiv.org/abs/2212.11979.

[7] Marichelo Garcia-Venegas et al. "On the safety of vulnerable road users by cyclist detection and tracking". In: *Machine vision and applications* 32.5 (Aug. 2021). DOI: 10.1007/s00138-021-01231-4. URL: https://doi.org/10.1007/s00138-021-01231-4.

[8] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. DOI: 10.1109/cvpr.2014.81. URL: https://doi.org/10.1109/cvpr.2014.81.

[9] Ross Girshick. "Fast R-CNN". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. DOI: 10.1109/iccv.2015.169. URL: https://doi.org/10.1109/iccv.2015.169.

[10] Shouxin Ren et al. "Faster R-CNN: towards Real-Time Object Detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (June 2017), pp. 1137–1149. DOI: 10.1109/tpami.2016.2577031. URL: https://doi.org/10.1109/tpami.2016.2577031.

[11] Kumar Apurv, Renran Tian, and Rini Sherony. "Detection of E-scooter Riders in Naturalistic Scenes". In: *arXiv preprint arXiv:2111.14060* (2021).

[12] Hien Thi Thu Nguyen, Minh Nguyen, and Qian Sun. *Electric Scooter and its Rider Detection Framework based on Deep Learning for supporting Scooter-*

Related Injury Emergency Services. Jan. 2021, pp. 233–246. DOI: 10.1007/978-3-030-72073-5\{_}18. URL: https://doi.org/10.1007/978-3-030-72073-5_18.

[13] Shane Gilroy et al. "E-Scooter rider detection and classification in dense urban environments". In: *Results in engineering* 16 (Dec. 2022), p. 100677. DOI: 10.1016/j.rineng.2022.100677. URL: https://doi.org/10.1016/j.rineng.2022.100677.

[14] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao et al. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors". In: arXiv, 2022. URL: https://doi.org/10.48550/arXiv.2207.02696.

[15] Alexey Bochkovskiy, Chien Yao Wang, and Hong Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: (2020).

[16] Eric R. Ziegel et al. "Applied Linear Statistical Models". In: *Technometrics* 34.1 (Feb. 1992), p. 121. DOI: 10.2307/1269588. URL: https://doi.org/10.2307/1269588.

[17] Leo Breiman et al. *Classification and regression trees*. Oct. 2017. DOI: 10.1201/9781315139470. URL: https://doi.org/10.1201/9781315139470.

[18] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (Jan. 2001), pp. 5–32. DOI: 10.1023/a:1010933404324. URL: https://doi.org/10.1023/a:1010933404324.

[19] Naomi Altman. "An introduction to kernel and Nearest-Neighbor nonparametric regression". In: *The American Statistician* 46.3 (Aug. 1992), pp. 175–185. DOI: 10.1080/00031305.1992.10475879. URL: https://doi.org/10.1080/00031305.1992.10475879.

[20] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning representations by Back-Propagating errors*. Sept. 2002, pp. 213–222. DOI: 10.7551/mitpress/1888.003.0013. URL: https://doi.org/10.7551/mitpress/1888.003.0013.

[21] "Optimal state estimation: Kalman, H [infinity], and nonlinear approaches". In: *Choice Reviews Online* 44.06 (Feb. 2007), pp. 44–3334. DOI: 10.5860/choice.44-3334. URL: https://doi.org/10.5860/choice.44-3334.

[22] F. TUNG H. E. RAUCH and C. T. STRIEBEL. "Maximum likelihood estimates of linear dynamic systems". In: *AIAA JOURNAL* (1965).

[23] Erich Schubert et al. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[24] Society of Automotive Engineers. "Operational definitions of driving performance measures and statistics". In: *Surface Vehicle Recommended Practice J2944* (2015).

[25] Roboflow. *RoboFlow Universe: Open Source Computer Vision Community*. URL: https://universe.roboflow.com/.

[26] Tzutalin. *LabelImg. Git code (2015)*. URL: https://github.com/tzutalin/labelImg.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY