CYCLIST STATE PREDICTION SYSTEM

D.M.A Mahawatta

199345P

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

June 2022

CYCLIST STATE PREDICTION SYSTEM

D.M.A Mahawatta

199345P

Dissertation submitted in partial fulfillment of the requirements for the Degree
MSc in Computer Science specializing in Software Architecture.

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa

Sri Lanka

June 2022

DECLARATION

I declare that this is my own work, and this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic, or another medium. I retain the right to use this content in whole or part in future works.

....

D.M.A Mahawatta Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this project report is acceptable for evaluation of the Final Research.

..... ……..

Prof. G.I.U.S Perera Date

ACKNOWLEDGEMENT

My sincere gratitude is extended to Dr. Indika Perera, my supervisor, Head of the Department of Computer Science and Engineering, Faculty of Engineering, the University of Moratuwa for the valuable guidance and support given throughout the end of this journey.

I would like to thank my family members and friends for the motivation and continuous support provided throughout this period of my life.

My appreciation goes to all my batchmates in the MSc batch and colleagues at my workplace for the assistance extended in managing work.

ABSTRACT

In a world moving more towards green initiatives, cycling has become a major way of transportation for many people across the world. Due to the rapid growth of automobile usage, cyclists are considered as one of the most vulnerable groups of road users. Even though there are various Collision avoiding systems available right now, most of them focus on pedestrians and highway driving scenarios. There are a smaller number of systems that focus on the safety of cyclists. Detecting Cyclists and predicting their intentions real-time may help in increasing cyclist safety in an urban environment. Some existing research require ideal conditions to predict the cyclist state while few are implemented exploiting various constants in the environment. Some research work requires to have known jersey patterns to detect cyclists among other automobile whereas some other work requires the data such as curb position and location to be constant/predefined while the ethnicity of the demography is also considered while it predicts only risky cycling scenarios. This research presents a holistic solution to detect and recognize cyclists in a complex environment without specific user given information focusing on ellipses detection applied to wheel patterns of the cyclists.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

With the rapid growth of automobile usage in the modern world, many issues have been evolved over the time. Due to the higher number of vehicles that are on the road, Cycling has emerged as a good alternative to the usage of automobile by decreasing the traffic in a more ecofriendly manner. Not only it does reduce the daily traffic caused by other vehicles, but also it provides limitless benefits to people and to our mother earth. Using bicycles for single person short distance rides is much better than using a fuel powered vehicle. Cycling will reduce the damage that is caused by carbon dioxide emission, and it will also uplift the health condition of people providing constant exercises.

In a world moving more towards green initiatives, cycling has become a major way of transportation for many people across the world. Specially in the European countries, there is a high tendency of people using cycling as one of their day to day traveling mode. Even though many people are encouraged to use bicycles instead of fuel powered vehicles, there is a high risk attached making it obnoxious due to several reasons.

Due to the rapid growth of automobile usage, cyclists are considered as one of the most vulnerable groups of road users [1]. Pedestrians and cyclists both are subjected to half of the road traffic accidents according to the World Health Organization (WHO) records [2]. The statistics against them is higher because pedestrians and cyclists do not wear any special means of protections such as helmets and clothing items. Cycling is beneficial to avoid traffic congestion, but its weak protection has drawn a red light towards cycling community. The damages of cyclist accidents are directly involved with personal injuries which could vary from mild to severe. According to WHO by 2030, traffic accidents related deaths will take the fifth place of leading death causes [3,4]. Pedestrian and cyclists' accident rates were recorded as 26% and 6% respectively out of all traffic accidents in United Kingdom 2017 [5]. These statistics show the necessity of a solution that prevents such accidents by providing collision avoidance and mitigation. In order to provide with a holistic solution, it is crucial to understand the cycling safety with respect to cycling maneuver.

Even though there are various Collision avoiding systems available right now, most of them focus on pedestrians and highway driving scenarios [6]. As an example, nowadays there are Advanced

Driver Assistance Algorithm Systems (ADAS) focusing highway driving scenarios mitigating special collisions that include pedestrians and other vehicles. However, the systems that focus on the safety of cyclists are less. Volvo has come up with a cyclist detection system, but the goal of that system is only to detect cyclists with the aid of camera and radar to support their auto braking system [7].

With that limitation as well as the rapid growth of cycle usage, Visual cyclist analysis has now gained a huge attention specially from European countries as they have shown a huge tendency in using cycles for their day today travels. Cyclist analysis is also highly beneficial for the autonomous vehicle manufacturing companies to ensure the safety of their products. This kind of analysis mainly focuses on understanding the cyclist behavior, but the cyclist state can also be predicted as an extension to such a system.

Detecting cyclist pattern behavior is considered as a complex process as it is always a combination of a bicycle and a pedestrian. Yet successful real time detection may help in increasing cyclist safety in urban environments. If a system can analyze cyclist behavior and predict the state of a cyclist behavior, it would massively help in preventing accidents that are caused by other automobiles including autonomous self-driving vehicles.

To bring this innovative idea into action, many research works have been carried out based on different theories. Use of state-of-the-art sensors along with deep learning approaches and computer vision take a prominent place in this research area.

The research aims to provide a novel approach to estimate cyclist intentions by analyzing ellipse along with cyclist pose. Ellipse detection in images is a technically complex and a challenging process which can be used in various applications such as eye tracking, wheel detection, road sign detection, etc. An accurate and robust ellipse detection is required to identify the front and back wheels of the bicycle and to estimate the state of the cyclist along with analyzing the cyclist behavior.

1.1 Problem

The related research problem is how to identify cyclist behaviors and classify them by analyzing wheel ellipse patterns along with cyclist postures.

It is important to carry out this study because cyclists are considered as one of the most vulnerable groups of road users yet there is not much mainstream active research going on targeting cycling safety by mitigating collisions. There are some cycling safety related researches going on such as [6], [8]. The Limitations or drawbacks of those can be pointed out as follows. Some research requires ideal conditions to predict the cyclist state while few are implemented exploiting various constants in the environment. As an example [6] requires having known jersey patterns to detect cyclists among other automobiles. [8] requires the data such as curb position and location to be constant/predefined while the ethnicity of the demography is also considered while it predicts only risky cycling scenarios.

In addition to that, automotive safety is also a significant factor which needs to be addressed due to the rapidly increasing number of roadside accidents. It is equally important as much as the cyclist and pedestrians' safety. Preventing automobile accidents will automatically result in increased cyclist safety for a certain extent because a considerable amount of reported collisions occur between vehicles and cycles. Some automobiles are equipped with collision avoidance and mitigation systems, yet their main objective is far from achieving cyclist safety. Some driver assistant systems have the capability of identifying the pedestrians with the use of infrared cameras and indicate driver accordingly. These systems are even capable of braking the system and stop the vehicle in order to prevent an accident happening [18].

Automotive safety is now considered as an active research area [17] which holds a significant level of importance towards the society. As a result of that, many researchers have developed algorithms that looks promising in the fields of detecting and tracking pedestrians using various technologies such as [19], [20], [21]. These studies mainly focus on detecting pedestrians, but they do not identify cyclists separately. With that information, it is proven that much less has been found in the context of cyclist safety even though bicyclists are considered as a vulnerable group of road users.

1.2 Objectives

Gather knowledge by reviewing and understanding the literature that are published on Cyclist state prediction frameworks.

Identifying the limitations/gaps of the existing literature and find solutions to overcome them.
Review and understand possible approaches to improve accuracy and reliability for cyclist state prediction frameworks.

Choose/create/annotate a cyclist dataset suitable to train a deep learning network for state prediction.

Design and implement a cyclist state prediction framework and evaluate with existing platforms.

1.3 Scope

At the successful completion of the research, a fully functioning cyclist state prediction system would be implemented and evaluated against the existing approaches.
The design of the system and the architecture will address the limitations that were identified during the literature review process. The implemented system will be capable of detecting cyclists from images which are captured through a dash camera mounted to an automobile. The system will further analyze the detected cyclist in order to localize the cycle wheels by ellipse detection. As the last step, the system will predict the cyclist dynamics after a series of calculations done based on the location and the angles of the ellipses.
The scope of this research project is to design and implement a fully functioning cyclist state prediction system which provides a solution to the identified problem and to create a dataset with proper annotations which is suitable to train the deep learning network. In addition to that the system evaluation will also belong in the project scope and it will be carried out separately for both cycling detection module and the state prediction system.

1.4 Outline

The first chapter starts with an introduction about the research. It discusses the problem that has been addressed by the research and the motivation behind the project. It also provides the objectives of the research which need to be achieved at the successful completion of the project duration and the project scope.

The second chapter discusses about the existing research work that addresses the same problem and their limitations. The chapter two emphasizes the gaps between existing studies versus the suggested approach. The literature review has been carried out under three different sections targeting different modules of the suggested approach.

The third chapter describes about the proposed methodology of the research. It starts with a small introduction about the proposed system and its design. The chapter first explains about the dataset that is being used and then it discusses about two main modules of the system. The design and architectural level details are emphasized during chapter three.

The fourth chapter is dedicated for the architecture of the system and for the implementation details.

The next chapter discusses about the evaluation criteria and the method of evaluation of the system. This chapter includes evaluation details for two main modules which are cycling detection module and the state prediction system.

The final chapter holds information about the conclusion of the research. It discusses about the limitations of the research and the future work/extensions that can be done on the research.

The necessary screenshots, code segments References and other additional information can be found under annexes.

CHAPTER 2
LITERATURE REVIEW

This chapter discusses about the similarities and differences of available research work that address similar problems related to the domain of cycling detection. These mentioned research works are focused on cyclist detection, cyclist recognition, cyclist maneuver classification and ellipse detection. Some of the mentioned research work will be used as the groundwork for the proposed system. Cyclist detection will be considered first as it is the first step that this research aims to achieve. Once the cyclist is detected and recognized, the maneuver must be classified. In order to do so, the cycle wheels must be detected and recognized. The existing research work that falls under the above-mentioned categories have been reviewed here.

2. 1 Research carried under Cyclist Detection

As the usage of autonomous vehicles is getting famous and trending, the need of having a holistic system to avoid collision is crucial. Lot of large-scale automobile manufacturers such as Volvo and Tesla have invested in research works on cyclist detection as cyclist accidents is one of the major problems their autonomous vehicles face.

In 2019, Sarfraz Ahmed et al [9] presented a review of recent research works carried out on cyclist and pedestrian detection to increase the safety of vulnerable road users and autonomous vehicle users. The methods and techniques involved with deep learning approaches such as Fast Region-Convolutional Neural Network (R-CNN), Faster R-CNN and Single Shot Detector (SSD) are mainly discussed in this review [9].

According to [9], region proposal approaches (two stage detectors) and non-region approaches (single stage detectors) can be used to perform pedestrian and cyclist detection. R-CNN, Regional-Fast Convolutional Network (R-FCN) and Faster R-CNN are some of the region proposal-based techniques that can be used for cyclist detection [9]. Single Shot Detector (SSD) and You Only Look Once (YOLO) are some of the non-region proposal-based techniques that can be applied to detect cyclists and pedestrians. Deep Neural Networks, DPM variants and Decision Forests can be used for classification along with the above detection techniques. The region proposal-based techniques can identify the existence of objects in the image and the identified locations can be fed into classifiers in order to determine the class of the object [9].

Non-region proposal-based techniques are widely used to reduce the computation power consumed by two stage techniques. Those are capable of both identifying the region where the pedestrian/cyclist appears and classifying the object using a single network. The systems that use non-region proposal-based techniques are considered as faster due to their low computation power consumption. To perform cyclist detection using these deep learning related techniques, different types of sensors are being used. The proposed system follows a deep learning approach with a Mask R-CNN model which is based on Faster R-CNN technique.

In 2018, Alexander Masalov et al [6] presented a novel method for cyclist recognition with a higher degree of accuracy using pre known jersey patterns. In this research they have introduced a hybrid approach to detection and classification known as CyDet [6]. Detection, Classification, and tacking are being done using machine learning approaches. The importance of this research study is that the researches have experimented on several detection methods which has supported them in finding the best detection approach with highest accuracy. The limitation of this method is that it is based on custom designed jersey styles which make it impossible to apply at more generic situations. Primarily this approach works only when the cyclist is wearing a jersey with known patterns.

The study [6] presents a hybrid approach incorporating hand-crafted knowledge layers along with Machine Learning layers in order to get the highest accuracy with a relatively smaller data set. In generally vehicles, other physical objects such as signposts, trees, unexpected glares, shadows on the videos, the distance between the bicycle and the camera often led to low visibility [6]. When detecting cyclists who are not entirely visible in the footage, a known pattern on their jerseys will be used. This approach helps in increasing the accuracy of the research. A cyclist is considered as one object rather than decomposing it as a human and bicycle which helps in eliminating the false negatives improving the recognition quality.

Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG) are used as visual descriptors for the classification. During the earlier stages of the algorithm, LBP and HOG both are applied on grayscale images which are the transformed into a set of descriptors [6]. This set of descriptors is later used as features vectors for the classification in Machine learning layer. This research [6] has used comparatively a smaller number of descriptors in order to decrease the model

size as well as the possibility of overfitting. AdaBoost has been selected as the Machine Learning classifier. Therefore, two classifiers are separately trained using LBP and HOG descriptor sets. These separate classifiers have been trained for different camera views and for known jersey patterns. If the cyclist is wearing one of the known jersey patterns, the classifier can recognize the cyclist object even if the bicycle object is out of view. Winkam's pattern detection algorithm has been used for the jersey pattern detection [6].

In order to enhance the recall and precision in the suggested algorithm, the final tracking layer checks for the previous frame features. If a cyclist is visible in a particular frame but not visible in previous and following frames, that event will be canceled. This tracking mechanism had significantly increased the accuracy rate according to the study [6].

Deep learning-based cyclist detection methods are proven to be well performing with respect to the traditional methods such as support vector machines, decision trees, etc. [9].

[35] presents a new approach to detect both pedestrians and cyclists together. The proposed method will identify candidate objects first and later through a faster R-CNN model, the objects will be classified and localized. This method consists of three main modules where the first module is responsible of detecting the upper body of the cyclist [35]. Since this research addresses both pedestrians and cyclists, appearance of the upper body has been considered as common to both pedestrians and cyclists and this theory has been the basis for the entire research. The upper body of the road users has been detected by focusing on the head and the torso. The detection happens through the upper body detector where locally decorrelated channel features variant is used [35]. The output of the upper body detector would be multiple potential regions where pedestrians and cyclists are visible. The potential regions will fed into the faster R-CNN model for classification and localization.

These research works are the groundwork that are laid to support the research area on cyclist state prediction. The state prediction can be done only if accurate cyclist detection is available. Cyclist detection itself cannot predict the behavior of cyclists. Hence research on Cyclist detection can be considered as the first step of cyclist state prediction.

2.2 Research carried under cycling maneuvers classification

Under this research work [10] cyclist maneuvers are identified in order to secure the cyclist safety in a complex physical environment involved with mixed traffic conditions. Yuanli Gu et al [10] have presented a new set of definitions of cycling maneuvers for Chinese cyclists along with a classification for those identified cycling maneuver categories. The recognized cycling maneuvers are as follows ; Passing, Avoiding, Carriageway-occupied, Sidewalk-occupied, Regular riding maneuvers [10]. The classification of above categories is performed by a Convolutional Neural Network (CNN) based method.

This research [10] contains two main modules to categorize cycling maneuvers into said five categories and to introduce a Deep Learning approach to classify different cycling maneuvers. The definitions for each maneuver category are clearly stated by Yuanli Gu et al [10] at the beginning of their research publication. Meeting maneuvers are not included in this study as China has unidirectional bicycle lanes. That is also considered as one of the gaps of this study because this research is based on Chinese cyclists and the CNN is modeled based on a feature set which is specific to China. As an example, bicycle path separation, width of bicycle path, road type, bicycle volume, etc. are used to train the CNN which makes this solution specific to one demographic area. An input layer, convolutional layer, pooling layers, fully connected layers, and an output layer are included in the CNN model.

The evaluation of this study is in favor of the suggested CNN model with respect to other traditional predicting models. A fair analysis has been performed on the testing data set before coming into conclusions.

The gap between [10] and the current study is that [10] is only applicable to Chinese cyclists and it cannot be applied to a complex environment. Furthermore, the use case of this research [10] is not focused on collision avoidance with respect to the usage of autonomous vehicles.

[36] presents a novel technique to identify bicycle passing maneuvers with the intention of ensuring the freedom of movement of cyclists in multi lane cycle paths. According to [36], cyclists who would use cycle paths with multiple lanes tend to loosen the cycling comfort due to the limited opportunities in changing the lanes freely because of other cyclists passing them. This approach first classifies the cycle passing maneuver and then models it to calculate the number of passes

[36]. Three types of passing maneuvers had been focused under this study[36]. Those are: A faster cycle passing a slower bicycle, cyclist meeting a pedestrian or another cyclist and delayed passing where the space in front of the cyclist is taken by other road users.

The gap between this study and the proposed method is that this method is only applicable to identify cyclist passing maneuvers. It cannot identify whether the cyclist is crossing a road or cycling along the road, etc. Moreover, the cyclists need to be in a multi lane unidirectional cycle path which can be considered as a huge constraint. This research will not be applicable in general conditions due to the said constraints and limitations. In addition to that, the problem that the proposed system has addressed will not be solved by the findings of [36] due to its limitations.

[37] presents a groundwork towards cycling maneuver classification where cyclist maneuver has been detected as a group or a single cyclist. In order to proceed with the research work, the researchers have ruled out a definition for cyclist groups. If the cyclist trajectory falls into same direction and if and only if the distance between the cyclists does not pass a certain value, that set of cyclists will be considered as a group of cyclists [37]. As the first step, the cyclist trajectories will be extracted from a sequence of video recording. The sequence of coordinates will be analyzed to classify them into clusters accordingly [37]. In order to accomplish this, various clustering algorithms such as agglomerative clustering, affinity propagation and DBSCAN have been tried in this study. Once the similar trajectories are identified, the second condition will be tested which is the distance between multiple cyclists. Euclidean distance has been calculated between two trajectories that share the same coordinates. To decide whether the cyclist exceeds a certain distance value or not, the optimal parameters must be selected. The model had been calibrated and validated using the available video clips. Initially the model has been calibrated manually to set the ground truths.

This study [37] only focusses on identifying whether the cyclists travel as a group or a single. It can be taken as a preliminary research work that supports and provides a partial solution to the problem that the current study has addressed. Furthermore, [37] does not focus on identifying further behaviors such as crossing the road, walking, etc.

2.3 Research carried under Ellipse Detection

Predicting the state of a cyclist based on a camera footage requires lots of research work. One feasible approach is to detect the wheels of the bicycle and detect the ellipse, by which the cyclist state can be predicted whether the cyclist is intended to turn the bicycle to left or right. The first step under a similar approach would be detecting the cyclist and the bicycle whereas detecting the ellipses comes as the second step. The previous sub chapters discuss about few research works that have been carried under cyclist detection as well as classification of their maneuvers. Ellipse detection has been considered as the next step of this study and it has often considered as the beginning of many computer vision applications.

In 2014, Michele Fornaciari et.al [11] presented a new approach for ellipse detection in real images using the arcs that belong to a same ellipse [11]. As the first step of the algorithm, the visible edges will be detected which will lead to arc detection later. The extracted arcs are separated into four classes according to the convexity [11]. The steps up to this point are combined and named as arc extraction. Canny edge detector with automatic thresholding has been used for edge detection and each identified edge point is classified using a classification algorithm [11]. The convexity of each arc will be calculated and arcs will be divided into two regions as concave and convex.

The next module is named as ellipse detection and the steps involved in this module are arc selection and parameter estimation [11]. In this approach an ellipse is considered as a triplet which means that it is a combination of three arcs. If three arcs satisfy a certain set of criteria, its likely to belong in one ellipse. In reality the number of arcs is higher. Therefore, performing such a complex calculation for all the available arcs would cost a lot of computing power. Therefore, in order to reduce the false detections, a triplet has been identified as two pairs sharing one arc. The algorithm will first select pairs whose arcs satisfy constraints related to convexity and mutual positions [11]. Once a triplet has been identified, the ellipse center will be computed. Based on that the entire ellipse will be created. The entire procedure will be finalized after the completion of post processing module which includes validation and clustering.

Another research work carried out in 2010 presents a promising approach for the ellipse detection based on the convexity of the curves and arcs. This research has been considered as the base for the current study as well. In [34], the edge curve extraction will process first with the use of canny

edge detectors. Once the edge map is retrieved, the connected edge contours will be extracted based on research work published in [38]. An abstract curve will be generated from the extracted edge contours and it will be the input for the process of extracting smooth edge curves [34]. The properties of identified smooth curves will be fed into the process of grouping them and generating the ellipses. The grouping and ellipse generating module will not physically combine the edge contours, but it will identify the edge contours that may belong to the same ellipse. The ellipse generation algorithm may produce multiple ellipses around the same region for the same ellipse. These multiple ellipses will be combined based on a clustering technique introduced in [34]. The most important characteristic of this research is that individual threshold values are selected for each image separately. It will increase the accuracy of the ellipse fitting algorithm significantly.

[34] has been evaluated against five other existing ellipse detection techniques and it has been proven that [34] beats the other five techniques in terms of precision, recall and F1 score. However, the time taken by the research study [34] is comparatively higher than the other five techniques. But the authors have further mentioned that the proposed approach can be optimized and parallelized based on specific application types and categories. One of the major benefits of this approach is that the entire technique is based on very less number of parameters and the methodology is less sensitive to the changes in control parameters [34].

These research work can be considered as the middle layer of cyclist state prediction system where the first step was involved in cyclist detection and recognition. The successful combination of compatible two approaches on cyclist detection and ellipse detection would be the ideal starting point to the state prediction system.

2.4 Summary

This chapter discusses about the existing literature based on three categories. The first section consists of information about the research work carried under cyclist detection. It identifies the gaps and the limitations between the existing study and the proposed system. Cycling detection research works have been conducted worldwide, yet most of them have different constraints attached. The second part describes about the research work carried under cycling maneuver classification. Similar to the first part, the limitations and the gaps of existing studies are emphasized here. The third part explains about the studies that have been conducted to detect ellipses. The objective of studying this topic is to determine the wheels of cyclists properly. This is a field where bit complex mathematical concepts are involved.

CHAPTER 3

METHODOLOGY

As stated in the above introduction, the intent of the research is to develop a system which can model/predict the behavior of a cyclist in the eye of a driver. The system will output whether the cyclist will be on a collision path in the future, relative to the vision of a driver, based on the given current cyclist position.

Two modules to be developed, first in order to localize the cyclist as well as the cycle wheel position in the given image, and the second to map a dynamic model of the cyclist path using the localized wheel positions/angles.

The modules will be tested on the below mentioned dataset with the scope of predicting whether the cyclist trajectory will be on a collision path with the host vehicle.

3. 1 Dataset

Tsinghua-Daimler Cyclist Detection Benchmark Dataset [13] will be used during the research project in order to create a new dataset that can train the model properly. Tsinghua-Daimler Cyclist Benchmark is a dataset which has nearly 15000 images of cyclists [13]. Also, the objects such as pedestrians, cyclists, motor cyclists, tricyclists, wheelchair users and moped riders are labeled with bounding boxes. This dataset is widely used in cyclist detection related research all over the world. The dataset consists of four subcategories namely Train, Valid, Test and NonVRU [13].

The training dataset has 9741 images with annotations for cyclists. The cyclist objects in train dataset are higher than 60 pixels and fully visible. Valid dataset has 1019 images which can be used for validation purposes, Object annotation is available there in the valid dataset for the above-mentioned rider categories and objects will be annotated only if they exceed 20 pixels. The test dataset contains 2914 images with annotations similar in the valid dataset. Test dataset images are used for testing purposes and there also the object must have 20 or more pixels to be annotated. NonVRU category has 1000 images which have no rider objects of above categories visible.
This dataset is freely available for academic purposes upon a license terms agreement.

	Training			Test set	Total
	Set 1	Set 2	Non-VRU		
Total Frames	9741	5095	1000	14570	30406
Labeled Frames	9741	1019	1000	2914	14674
Total BBs	16202	3016	0	13143	32361
Cyclist BBs	16202	1301	0	4658	22161
Pedestrian BBs	0	1539	0	7380	8919
Other rider BBs	0	176	0	1105	1281

Figure 3.1: Statistics of Tsinghua-Daimler cyclist benchmark

Source: [13]

Figure 3.2: Samples from the dataset. (a) Cyclist Samples; (b) Test images with annotations

Source: [13]

This dataset caters only for cycling detection. But in this research, it is important to localize the wheels of detected cyclists. Therefore, to facilitate the requirements of the research, a new dataset is required with annotations for the wheels of cyclists.

A new wheel annotated dataset was created with the use of the Tsinghua-Daimler Cyclist Detection Benchmark Dataset. The selected images were annotated using the VGG Image Annotator (VIA) tool which is a simple standalone manual annotation tool [22]. The tool can easily run on a web browser without any installation or setup. People can manually annotate interested regions in images as well as in videos using this tool. The annotations can be exported into plain text data formats such as JSON and CSV [22]. The new dataset created for the research contains two main directories named train and val. Train directory consists of jpg image files used as training data

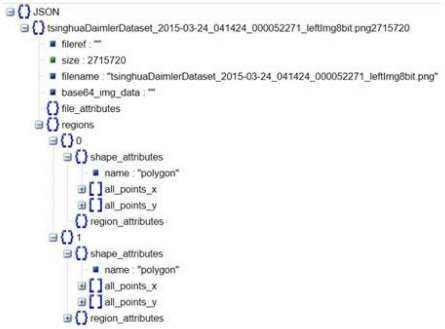
and the JSON file which contains annotation details. Similarly, Val directory contains images used for validation as well as the JSON file with the annotation details.

(a)

(b)

(c)

(d)



(e)

#filename	file_size	file_attribi	region_co	region_id	region_shape_attributes	region_attributes
tsinghuaD	2715720	{}	2	0	{"name":"polygon","all_points_x":[2,15,28,53,68,90,116,147,175,201,225,238,247,247,241,234,225,203,182,155,107,83,22,7,2,2,0,2,2,0,2],"all_points_y":[792,777,766,753,748,742,737,735,744,766,790,814,845,880,899,923,950,971,989,1002,1013,1015,1006,985,969,945,888,862,840,814,792]}	{}
tsinghuaD	2715720	{}	2	1	{"name":"polygon","all_points_x":[352,359,365,372,387,409,427,440,462,468,477,488,497,503,503,497,486,477,462,444,420,394,381,372,363,357,352],"all_points_y":[831,777,748,735,713,696,691,687,687,687,702,715,742,775,796,823,858,875,893,910,915,915,901,884,871,849,831]}	{}

Figure 3.3: Wheel annotation with VIA. (a) Original image; (b) Annotation of front wheel; (c) Annotation of rear wheel; (d) JASON file of annotations; (e) CSV file of annotations

3. 2 Cyclist wheel localization module

3.2.1. Mask R-CNN

Convolutional Neural Network (CNN) as known as ConvNet is a type of deep neural networks that can be used in analyzing visuals. CNNs are widely used in the field of Image recognition and classification due to its higher-level accuracy compared to other low level feature extraction techniques. CNNs consist of multiple layers as in the below figure.

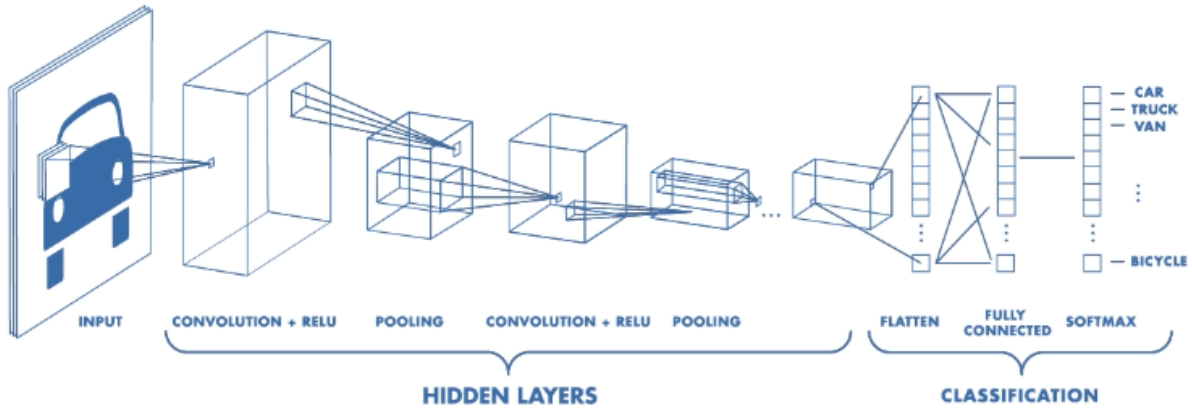


Figure 3.4: CNN Architecture

Region-Based Convolutional Neural Network (RCNN) is also a class of deep learning which produces even more accurate results than CNN due to its region-based analysis. R-CNN selects a limited number of regions from the given image and evaluates CNNs separately on all regions of interests [24]. R-CNN architecture specifically focusses on image detection related tasks. R-CNN was later improved to Faster R-CNN version where the architecture is consists with two main stages namely Regional Proposal Network (RPN) and Fact R-CNN [25]. Faster R-CNN is faster than R-CNN due to the feature map that it produces. The neural network of Faster R-CNN does not have to operate on all interested regions unlike in R-CNN. The sole reason behind inventing Faster R-CNN is to reduce the computation time of R-CNN. Faster R-CNN is the foundation of Mask R-CNN. A pretrained Mask R-CNN framework will be used in this study.

Mask R-CNN is an extended version of Faster R-CNN [23] and it consists with two phases. The first phase will scan the image and produce the interested regions whereas the second stage will classify the objects and generate the bounding boxes and masks. The difference between Faster R-CNN and Mask R-CNN is that the latter provides a binary mask for each ROI during the second stage [23].

In this research study, A combination of ResNet101 standard CNN and Feature Pyramid Network (FPN) has been used as the backbone. The beginner layers of CNN identify the low level features such as edges and corners whereas the later layers will identify high level complex features such as different objects. The feature map created by the ResNet101 will be the input to the next stages. The feature extraction process is further improved by the combination of FPN to ResNet101.

Regional Proposal Network (RPN) then scans the generated feature map for regions which are called as anchors. RPN will produce the class type and the bounding box for each anchor. According to the RPN predictions, the highest anchors with objects will be selected to record the location and size.

Next stage consists of the ROI classifier and Bounding box regressor which will work on the output of RPN. Each ROI will receive a class type and the bounding box refinement at the end. The difference here is, the class being very specific unlike in RPN, and it will identify the object type as in human context such as car, ball, wheel, etc. Even though the object is properly classified at the end of this stage, further improvements are required due to few issues that may occur based on the different input sizes of the ROI boxes. ROI pooling will resize the feature map and convert it to a fixed size that is required by the classifier.

As the final step, mask branch will be added to the Faster R-CNN. It will consider the positive regions provided by the classifier and generate masks.

3.2.2. Training the Mask R-CNN

As for the first step, the new dataset was created by annotating images taken from Tsinghua-Daimler Cyclist Detection Benchmark Dataset. The wheels of the bicycles were annotated with the use of VGG Image Annotator tool. Annotating images and preparing the dataset consume a considerable amount of time. Transfer learning method was used to train the model as the CNN was pre trained. It has certain advantages over training the model from the scratch. The model was performing well even under a limited number of annotated images. Therefore, it increases the accuracy of the entire system.

The JSON file taken from the VIA tool was used to extract the annotations and add classes.

Step 1: Reading the JSON file

```
def load_wheel(self, dataset_dir, subset):
    |
    # Add classes.
    self.add_class("wheel", 1, "wheel")

    # Train or validation dataset?
    assert subset in ["train", "val"]
    dataset_dir = os.path.join(dataset_dir, subset)
```

Step 2: Load Annotations

```
annotations = json.load(open(os.path.join(dataset_dir, "via_region_data.json")))
annotations = list(annotations.values())  # don't need the dict keys
```

Step 3: Build the dataset

```
    self.add_image(
        "wheel",
        image_id=a['filename'],  # use file name as a unique image id
        path=image_path,
        width=width, height=height,
        polygons=polygons)
```

Step 4: Generate the bitmap mask

```
def load_mask(self, image_id):
    image_info = self.image_info[image_id]
    if image_info["source"] != "balloon":
        return super(self.__class__, self).load_mask(image_id)
    info = self.image_info[image_id]
    mask = np.zeros([info["height"], info["width"], len(info["polygons"])],
                    dtype=np.uint8)
    for i, p in enumerate(info["polygons"]):
        rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
        mask[rr, cc, i] = 1
```

Step 5 : Training

```
%cd ~/Mask_RCNN

!python cyclist.py train --dataset=dataset/newAnnotations/ --weights=coco

%cp -av "/logs/*" "/content/drive/My Drive/Colab Notebooks/logs/"
```

3.3 Ellipse fitting module

The next main module is the ellipse fitter module. This module is responsible of identifying the wheel from the given image with the bounding box details provided by the wheel localization module. This module will display the image with marked wheels as the final output.

The cyclist wheel localization module will output the image with bounding boxes around the wheels. The ellipse fitting process will start from the ROI coordinates that can be retrieved through the said image data file. The canny edge detector will be applied to the above data to retrieve the edge map. If the CNN has identified more than one wheel in the image, the canny edge detector will be applied to all possible wheel ROIs.

As for the next step, pixels with positive and negative orientations will be identified. This is done based on the sign of the tangent of the orientation.

$$\text{sign}\left(\tan\varphi(\rho)\right) = \text{sign}\left(G_x(\rho)\right).\text{sign}\left(G_y(\rho)\right) \tag{3.1}$$

Horizontal and Vertical derivatives are named as $G_x(\rho)$and $G_y(\rho)$ respectively.

The pixels will be stored in two different sets according to their orientation sign. In the identified edges, there could be connected arcs. Those connected arcs will be defined based on the edge linking algorithm presented by [26]. The connected arcs obtained from positive and negative sets will again store on a separate group of sets based on the sign. In order to determine the direction of the arc, the convexity is calculated according to the algorithm presented in [27].

$$\mathrm{Conv}(a_i) = \begin{cases} 1, & \mathrm{Area}(L^i) > \mathrm{Area}(U^i) \\ -1, & \mathrm{Area}(U^i) > \mathrm{Area}(L^i) \end{cases} \tag{3.2}$$

The quadrant to which the arch must belong can be identified based on the information gathered through above calculation.

The next step is to group the arcs and draw the ellipse. Arcs will be grouped based on an anticlockwise order. The pairing threshold can be changed based on the external factors such as the wheel thickness and the size of the bicycle. A good threshold value will result in pairing the arcs that belong in the same ellipse.

The pairs which share a common arc will grouped together to form a triplet. These triplets will be used to fit the ellipse and that will be the last step of this module.

3.4 Cycle dynamics prediction module by extracted wheel parameters.

3.4.1 Kalman filter

Kalman filter, also known as linear quadratic estimation is used widely in trajectory estimation and optimization which uses a series of measurements observed over time. This algorithm uses the idea of projecting the state forward with a function to transfer state [15].

$Algorithm: Kalman\_Filter$
$Input: s_{t-1}, P_{t-1}, u_t, z_t$
$Output: s_t^-, P_t^-$

1. $s_t = As_{t-1} + Bu_t + w_t$
2. $P_t = AP_{t-1}A^T + Q$
3. $K = PH^T(HPH^T + Q)^{-1}$
4. $\hat{z} = (z_t - H_t s_t)$
5. $s_t^- = s_t + K\hat{z}$
6. $P_t^- = (I - K_t H_t)P_t$
7. $return\ s_t^-, P_t^-$

Figure 3.5: Pseudo code for the Kalman filter

Source : [15]

3.4.2 Particle Filter

Particle filtering is used mostly when the state-space model becomes non-linear and the starting state as well as the noise distributions are taking a random form, where it uses a set of samples i.e., particles to represent the future position/state given those partial observations [16].

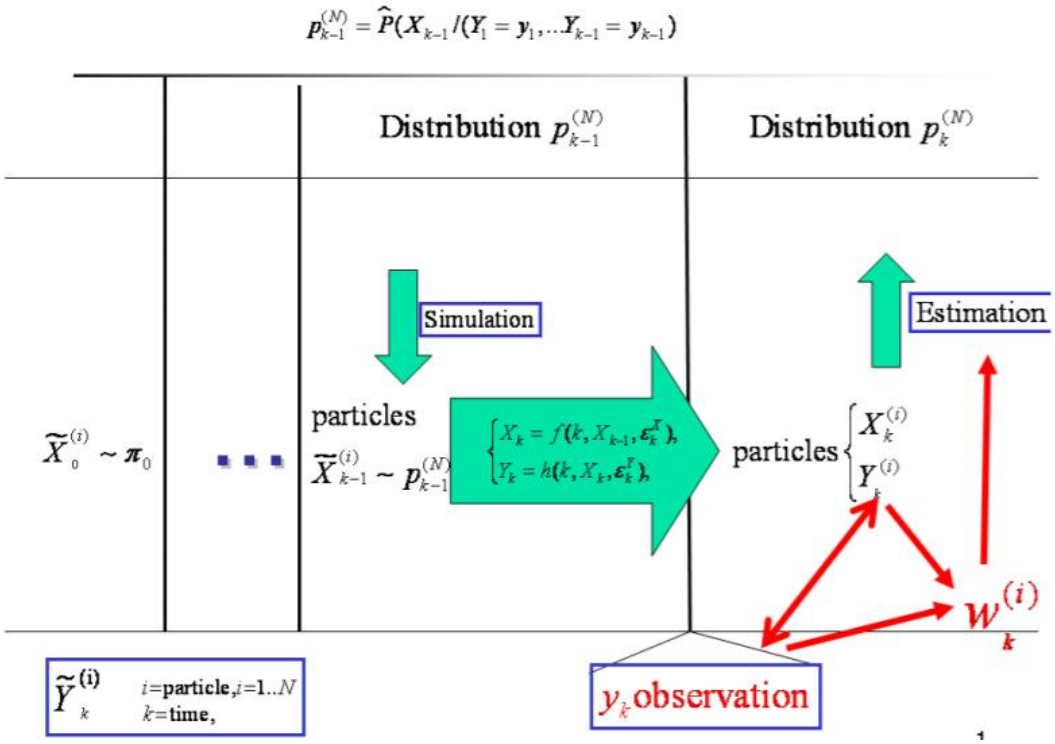


Figure 3.6: Summarization of particle filter algorithm

Source: [16]

3.4.3 Proposed Approach for the Cycle dynamics prediction module

Using the detected model of the wheel state, the cycle trajectory will be predicted with the accuracy of the ellipse fitting done in the above module. Using a Kalman filter/particle filter is proposed to predict the trajectory of the cycle along with a development of a motion model. One of the above filtering approaches are proposed to be used when estimating the state of the cyclist. This module will use the ellipse parameters obtained by the 3.1 as the input and will use that to model the predicted state of the cyclist. A comparison of the Kalman filter against the particle filter will be included into the scope of this research for the prediction of the trajectory.

Furthermore, the scope of the state prediction will be derived to determine only to predict whether the cycle is crossing the path of the host vehicle, leading to a collision.

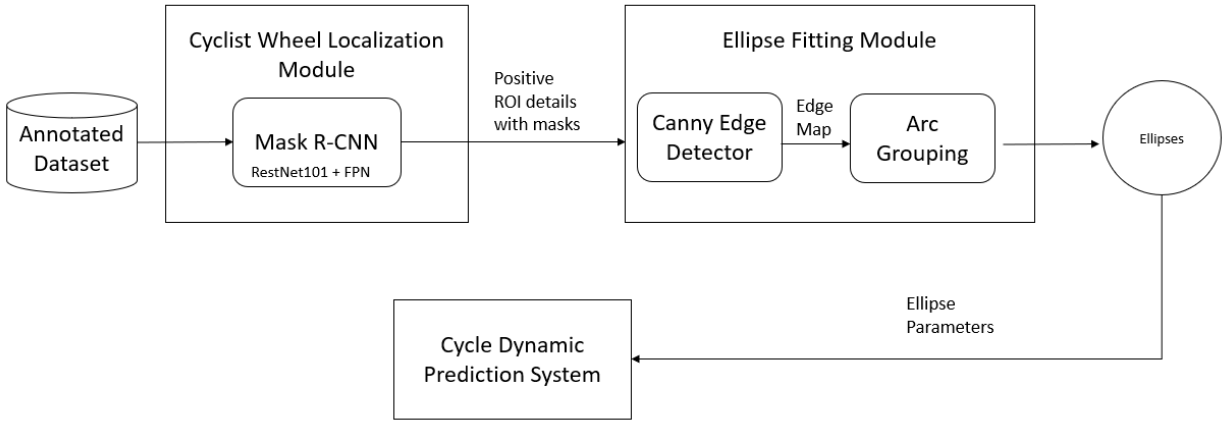


Figure 3.7: Component Diagram of the overall system

3.5 Summary

The above chapter describes the main components of the research and the methodology behind them. Furthermore, the chapter elaborates on how the new dataset was created from an existing dataset. A detailed description of the design of the Mask R-CNN model and its training process can be found in this chapter.

CHAPTER 4

SYSTEM ARCHITECTURE AND IMPLEMENTATION

The proposed system design of the Wheel localization module is based on a combination of ResNet101 architecture and Feature Pyramid Network (FPN). During the implementation, FPN has been created right after building the ResNet. FPN consumes a bit of time and memory due to its multiple processing, yet it increases the accuracy of the ROI align layer.

4.1 ResNet101 Architecture

ResNet architecture was introduced to eliminate the accuracy drop issue in deep CNN models due to the higher number of layers [28]. ResNet is a shortened form for the term Residual Network. ResNet architecture has many variants namely ResNet34, ResNet50, ResNet101 and ResNet152. Out of all variants, ResNet50 and ResNet101 are widely used in the object detection discipline. ResNet101 is often used with Faster R-CNN models which is the main reason behind choosing ResNet101 as the architecture for this study because it is based on Mask R-CNN which is an extension of Faster R-CNN.
The pre-trained ResNet101 model has the ability to identify and classify images into nearly a thousand classes. This pretrained model has been trained from the Coco dataset [30] which has over a million images. Therefore, it is said that the pre-trained ResNet101 model has the capability of classifying objects accurately similar to the human eye.

ResNet101 is a CNN with 101 deep layers. It has 3 more layer blocks which help in constructing 101 layers. In residual Networks, a shortcut connection is inserted to convert the network into a counterpart residual version. The ResNet architecture consists of four main stages. The model accepts input images with width and height in multiples of 3 and 32 channel width. The max-pooling of ResNet101 will be done based on 3x3 and 7x7 kernel sizes with a stride of 2. When the additionally added block which means the residual block is performed with a stride of 2, that implies that the input size will be reduced by half whereas the channel width will be doubled. Solid arrows in Figure 4.1 show the shortcut connection.

During the implementation, training was restricted for some layers to freeze their original weights. Transfer learning has been done on the model with the wheel annotated dataset which was created from the scratch using the Tsinghua-Daimler Cyclist Detection Benchmark Dataset. Freezing

unwanted layers will help in reducing the training time massively. In addition to that since the model was trained with coco dataset which has over million images, the results showed high accuracy level even the model was retrained with a few images sample.

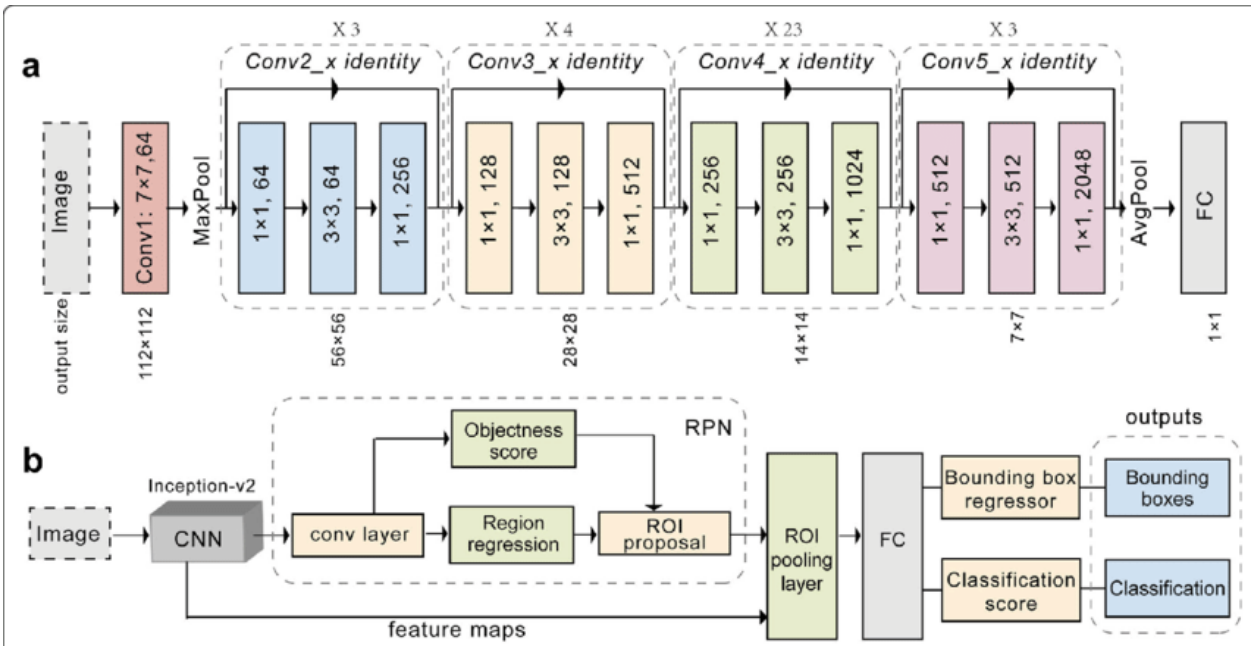


Figure 4.1: ResNet101 Architecture

Source: [29]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.2: ResNet general Architecture

Source: [31]

4.2 Feature Pyramid Network (FPN) Architecture

FPN was combined with the Mask R-CNN model in order to improve the feature extraction with a higher accuracy level. The FPN will create a pyramid of features focusing on the speed of the extraction as well as the accuracy.

FPN considers a single-scale image as the input and produces an output with correspondingly sized feature maps at several levels. This process is independent of the architecture of the ResNet. The bottom-up pathway of FPN is used in this study where a feed-forward computation will execute [32]. FPN will add additional feature pyramids that will receive high-level feature data from the first pyramid and pass them to the lower layers. This architecture will enable each level to access both higher and lower level feature details.

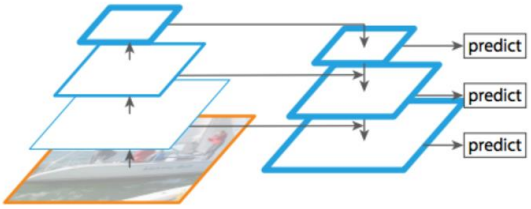


Figure 4.3: FPN Architecture

Source: [32]

4.3 Region Proposal Network (RPN) Architecture

FPN is only a feature extractor, it cannot act as an object detector. Therefore, in order to detect objects in each image, a Region Proposal Network (RPN) has been implemented. RPN can detect the regions of the image where certain objects are visible. These regions are called anchors and they will be distributed across the entire image. The features extracted through FPN are fed into the RPN as inputs [33]. RPN scans the feature map to make a prediction whether an object is available or not.

4.4 Implementation

The Mask R-CNN model was implemented using python 3 and TensorFlow with Keras Python deep learning API. It has two main blocks called the identity block and the convolutional block. The identity block is used if the input has the same dimensions as the output. The convolutional block is used when the input and output activations have different dimensions.

```
def identity_block(input_tensor, kernel_size, filters, stage, block,
                   use_bias=True, train_bn=True):

    nb_filter1, nb_filter2, nb_filter3 = filters
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    x = KL.Conv2D(nb_filter1, (1, 1), name=conv_name_base + '2a',
                  use_bias=use_bias)(input_tensor)
    x = BatchNorm(name=bn_name_base + '2a')(x, training=train_bn)
    x = KL.Activation('relu')(x)

    x = KL.Conv2D(nb_filter2, (kernel_size, kernel_size), padding='same',
                  name=conv_name_base + '2b', use_bias=use_bias)(x)
    x = BatchNorm(name=bn_name_base + '2b')(x, training=train_bn)
    x = KL.Activation('relu')(x)

    x = KL.Conv2D(nb_filter3, (1, 1), name=conv_name_base + '2c',
                  use_bias=use_bias)(x)
    x = BatchNorm(name=bn_name_base + '2c')(x, training=train_bn)

    x = KL.Add()([x, input_tensor])
    x = KL.Activation('relu', name='res' + str(stage) + block + '_out')(x)
    return x
```

Figure 4.4: Implementation of Identity block

```
def conv_block(input_tensor, kernel_size, filters, stage, block,
               strides=(2, 2), use_bias=True, train_bn=True):
    nb_filter1, nb_filter2, nb_filter3 = filters
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    x = KL.Conv2D(nb_filter1, (1, 1), strides=strides,
                  name=conv_name_base + '2a', use_bias=use_bias)(input_tensor)
    x = BatchNorm(name=bn_name_base + '2a')(x, training=train_bn)
    x = KL.Activation('relu')(x)

    x = KL.Conv2D(nb_filter2, (kernel_size, kernel_size), padding='same',
                  name=conv_name_base + '2b', use_bias=use_bias)(x)
    x = BatchNorm(name=bn_name_base + '2b')(x, training=train_bn)
    x = KL.Activation('relu')(x)

    x = KL.Conv2D(nb_filter3, (1, 1), name=conv_name_base +
                  '2c', use_bias=use_bias)(x)
    x = BatchNorm(name=bn_name_base + '2c')(x, training=train_bn)

    shortcut = KL.Conv2D(nb_filter3, (1, 1), strides=strides,
                         name=conv_name_base + '1', use_bias=use_bias)(input_tensor)
    shortcut = BatchNorm(name=bn_name_base + '1')(shortcut, training=train_bn)

    x = KL.Add()([x, shortcut])
    x = KL.Activation('relu', name='res' + str(stage) + block + '_out')(x)
    return x
```

Figure 4.5: Implementation of Convolutional block

```python
def resnet_graph(input_image, architecture, stage5=False, train_bn=True):
    assert architecture in ["resnet50", "resnet101"]
    # Stage 1
    x = KL.ZeroPadding2D((3, 3))(input_image)
    x = KL.Conv2D(64, (7, 7), strides=(2, 2), name='conv1', use_bias=True)(x)
    x = BatchNorm(name='bn_conv1')(x, training=train_bn)
    x = KL.Activation('relu')(x)
    C1 = x = KL.MaxPooling2D((3, 3), strides=(2, 2), padding="same")(x)
    # Stage 2
    x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1), train_bn=train_bn)
    x = identity_block(x, 3, [64, 64, 256], stage=2, block='b', train_bn=train_bn)
    C2 = x = identity_block(x, 3, [64, 64, 256], stage=2, block='c', train_bn=train_bn)
    # Stage 3
    x = conv_block(x, 3, [128, 128, 512], stage=3, block='a', train_bn=train_bn)
    x = identity_block(x, 3, [128, 128, 512], stage=3, block='b', train_bn=train_bn)
    x = identity_block(x, 3, [128, 128, 512], stage=3, block='c', train_bn=train_bn)
    C3 = x = identity_block(x, 3, [128, 128, 512], stage=3, block='d', train_bn=train_bn)
```

```python
    # Stage 4
    x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a', train_bn=train_bn)
    block_count = {"resnet50": 5, "resnet101": 22}[architecture]
    for i in range(block_count):
        x = identity_block(x, 3, [256, 256, 1024], stage=4, block=chr(98 + i), train_bn=train_bn)
    C4 = x
    # Stage 5
    if stage5:
        x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a', train_bn=train_bn)
        x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b', train_bn=train_bn)
        C5 = x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c', train_bn=train_bn)
    else:
        C5 = None
    return [C1, C2, C3, C4, C5]
```

Figure 4.6: Implementation of ResNet

```
class PyramidROIAlign(KE.Layer):
    def __init__(self, pool_shape, **kwargs):
        super(PyramidROIAlign, self).__init__(**kwargs)
        self.pool_shape = tuple(pool_shape)

    def get_config(self):
        config = super(PyramidROIAlign, self).get_config()
        config['pool_shape'] = self.pool_shape
        return config

    def call(self, inputs):
        boxes = inputs[0]
        image_meta = inputs[1]
        feature_maps = inputs[2:]
        y1, x1, y2, x2 = tf.split(boxes, 4, axis=2)
        h = y2 - y1
        w = x2 - x1
        image_shape = parse_image_meta_graph(image_meta)['image_shape'][0]
        image_area = tf.cast(image_shape[0] * image_shape[1], tf.float32)
        roi_level = log2_graph(tf.sqrt(h * w) / (224.0 / tf.sqrt(image_area)))
        roi_level = tf.minimum(5, tf.maximum(
            2, 4 + tf.cast(tf.round(roi_level), tf.int32)))
        roi_level = tf.squeeze(roi_level, 2)
        pooled = []
        box_to_level = []
```

```
    for i, level in enumerate(range(2, 6)):
        ix = tf.compat.v1.where(tf.equal(roi_level, level))
        level_boxes = tf.gather_nd(boxes, ix)
        box_indices = tf.cast(ix[:, 0], tf.int32)
        box_to_level.append(ix)
        level_boxes = tf.stop_gradient(level_boxes)
        box_indices = tf.stop_gradient(box_indices)
        pooled.append(tf.image.crop_and_resize(
            feature_maps[i], level_boxes, box_indices, self.pool_shape,
            method="bilinear"))
    pooled = tf.concat(pooled, axis=0)
    box_to_level = tf.concat(box_to_level, axis=0)
    box_range = tf.expand_dims(tf.range(tf.shape(input=box_to_level)[0]), 1)
    box_to_level = tf.concat([tf.cast(box_to_level, tf.int32), box_range],
                             axis=1)
    sorting_tensor = box_to_level[:, 0] * 100000 + box_to_level[:, 1]
    ix = tf.nn.top_k(sorting_tensor, k=tf.shape(
        input=box_to_level)[0]).indices[::-1]
    ix = tf.gather(box_to_level[:, 2], ix)
    pooled = tf.gather(pooled, ix)
    shape = tf.concat([tf.shape(input=boxes)[:2], tf.shape(input=pooled)[1:]], axis=0)
    pooled = tf.reshape(pooled, shape)
    return pooled

def compute_output_shape(self, input_shape):
    return input_shape[0][:2] + self.pool_shape + (input_shape[2][-1], )
```

Figure 4.7: Implementation of FPN

```python
def rpn_graph(feature_map, anchors_per_location, anchor_stride):
    shared = KL.Conv2D(512, (3, 3), padding='same', activation='relu',
                       strides=anchor_stride,
                       name='rpn_conv_shared')(feature_map)
    x = KL.Conv2D(2 * anchors_per_location, (1, 1), padding='valid',
                  activation='linear', name='rpn_class_raw')(shared)
    rpn_class_logits = KL.Lambda(
        lambda t: tf.reshape(t, [tf.shape(input=t)[0], -1, 2]))(x)
    rpn_probs = KL.Activation(
        "softmax", name="rpn_class_xxx")(rpn_class_logits)
    x = KL.Conv2D(anchors_per_location * 4, (1, 1), padding="valid",
                  activation='linear', name='rpn_bbox_pred')(shared)
    rpn_bbox = KL.Lambda(lambda t: tf.reshape(t, [tf.shape(input=t)[0], -1, 4]))(x)

    return [rpn_class_logits, rpn_probs, rpn_bbox]
```

Figure 4.8: Implementation of RPN

The pretrained model was imported into the python script for transfer learning purposes. Visual Studio Code was used as the development environment and Google collab was used to train the model with wheel annotated dataset.

```python
# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from maskrcnn.config import Config
from maskrcnn import model as modellib, utils
```

Figure 4.9: Importing the model

```
[ ]  %cd ~/Mask_RCNN

     !python cyclist.py train --dataset=dataset/newAnnotations/ --weights=coco

     %cp -av "/logs/*" "/content/drive/My Drive/Colab Notebooks/logs/"
```

Figure 4.10: Training the model with new dataset using Google collab

```
image ID: cyclist.tsinghuaDaimlerDataset_2015-03-24_041424_000042541_leftImg8bit.png (0) /root/I
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.10000  float64
image_metas              shape: (1, 14)               min:    0.00000  max: 1024.00000  int64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
gt_class_id              shape: (2,)                  min:    1.00000  max:    1.00000  int32
gt_bbox                  shape: (2, 4)                min:  141.00000  max:  636.00000  int32
gt_mask                  shape: (56, 56, 2)           min:    0.00000  max:    1.00000  bool
roi                      shape: (3, 4)                min:  135.00000  max:  768.00000  int32
514
y1                       shape: ()                    min:  514.00000  max:  514.00000  int32
x1                       shape: ()                    min:  737.00000  max:  737.00000  int32
y2                       shape: ()                    min:  603.00000  max:  603.00000  int32
x2                       shape: ()                    min:  768.00000  max:  768.00000  int32
517
y1                       shape: ()                    min:  517.00000  max:  517.00000  int32
x1                       shape: ()                    min:  263.00000  max:  263.00000  int32
y2                       shape: ()                    min:  613.00000  max:  613.00000  int32
x2                       shape: ()                    min:  327.00000  max:  327.00000  int32
```

Figure 4.11: Output from the model; ROI coordinates of the bounding boxes

The wheel ellipse fitting module was developed with MATLAB since a lot of mathematical calculations were included.

4.4 Summary

The fourth chapter explained the implementation details of the overall system along with the architectures of each component. It further describes the technologies and the development environments that were used during the implementation stage.

CHAPTER 5

EVALUATION

The evaluation chapter shows the results produced by the system and the evaluation of the system based on separate components. The wheel localization module and ellipse fitting module will be evaluated against existing similar research work authored by Prasad. D.K et al. [34]. This research work is also based on the same dataset. This study has been published in 2012 and it focuses more on mathematical calculations related to convexities and edge curvatures of the wheels. The proposed system has been tested on a dataset with around 1500 images which contains both real and synthetic images. The paper concludes with the evaluation details of the system mentioning that it performs well and better than existing approaches.

Various trials were carried out during the implementation process to identify certain threshold values. Currently, the system is not focusing on preprocessing the input images to a great extent. Therefore, the input images need to be in good quality to get accurate results. But further research can be carried out on the same topic providing room for users to upload poor quality images. As for the moment, the input images should be originals without any further processing such as resampling or application of filters.

The Mask R-CNN model which acts as the wheel localization model was able to localize the wheels of given images by drawing a bounding box around wheels. Figure 5.1 shows the outputs of the CNN model with the bounding boxes drawn around all visible wheels. In this method, the system does not have to identify the cyclist and then further analyses to localize the wheels. But instead, it can directly identify the regions where wheels are visible.

Figure 5.2 shows the outputs of the ellipse fitting module. It shows the original image, the ROI based on the CNN output, candidate patches and the best fit for the ellipse. This module takes coordinates of the ROI as input and separate the region to perform the mathematical calculations on it. For the recording purposes a middle step has been shown in Figure 5.2 (c) where multiple candidate patches are shown.

Figure 5.1: Output from the Wheel localization model

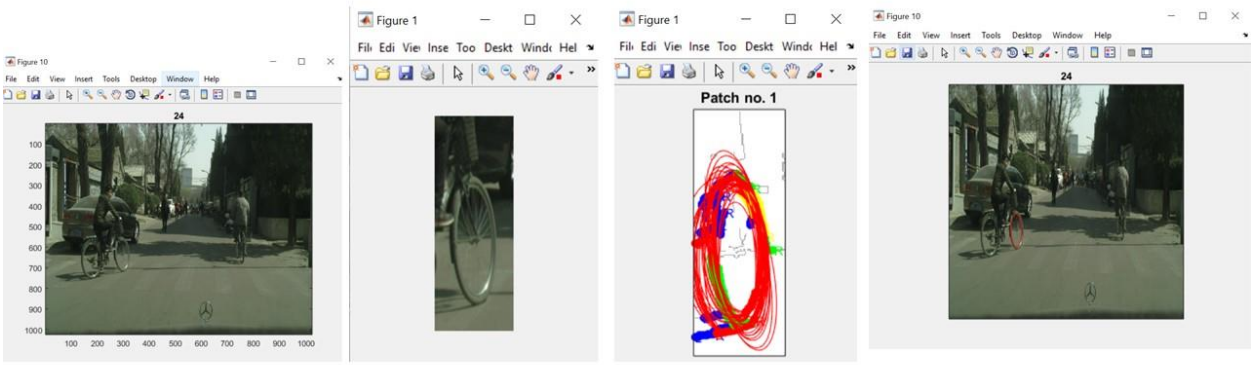


Figure 5.2: Output of ellipse fitting module. (a) Original image; (b) ROI based on the inputs of Mask R-CNN; (c) candidate patches; (d) fitted ellipse;

5.1 Evaluation Matrices

The below matrices were used to evaluate the system.

$$Precision = \frac{True\ Positives}{Predicted\ Condition\ Positive} \tag{5.1}$$

High precision value (Position Predictive Value) – the system often yields accurate results

$$Recall = \frac{True\ Positives}{Condition\ Positive} \tag{5.2}$$

High recall value (Sensitivity) – System is reliable

$$F1\ Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \tag{5.3}$$

The evaluation was carried out to a data sample that contains 100 images where cyclists and wheels were visible in 84 images and cyclists and wheels were not visible in 16 images. The system localized wheels and fitted them correctly in 62 images. True positive hypothesis was considered as system identifying wheels and fitting them correctly in the images where cyclists and wheels are visible. Therefore, the true positive rate for the evaluation holds the value 56. The system correctly identified that wheels were not visible in the 16 images where cyclists and wheels were actually not present. Other objects were identified as wheels in 14 images leading to a false positive rate of 14. Wheels were not identified in 08 images making the false negative value 08.

Table 5.1: Evaluation Statistics

		True Condition	
	Total Population	Condition Positive = 70	Condition Negative = 30
Predicted Condition	Predicted Condition Positive - 76	True Positive (TP) =62 (Power)	False Positive (FP) = 14 (Type I Error)
	Predicted Condition Negative - 24	False Negative (FN) = 8 (Type II Error)	True Negative (TN) = 16
		Recall = True Positive Rate (TPR) = TP/Condition Positive = 62/70 = 0.8857	Fall Out = False Positive rate (FPR) =FP/Condition Negative =13/30 = 0.4333
		Miss Rate = False Negative Rate (FNR) = FN/Condition Positive = 8/70 = 0.1143	True Negative Rate (TNR), Specificity = TN/Condition Negative = 16/30 = 0.5333

Table 5.2: Evaluation Statistics

<table>
<tr><td>Prevalence = Condition Positive/ Total Population =70/100 = 0.70</td><td colspan="2">Accuracy (ACC) = (TP+TN)/ Total Population =78/100 = 0.78</td></tr>
<tr><td>Precision = Positive Predictive Value (PPV) = TP/Predicted Condition Positive = 62/76 = 0.8158</td><td colspan="2">False Discovery Rate (FDR) =FP/ Predicted Condition Negative =14/24 = 0.5833</td></tr>
<tr><td>False Omission Rate (FOR) = FN/ Predicted Condition Negative =8/24 = 0.3333</td><td colspan="2">Negative Predicted Value (NPV) = TN/ Predicted Condition Negative =16/24 = 0.6667</td></tr>
<tr><td>Positive Likelihood Ratio (LR+) = TPR/FPR = 0.8857/0.4333= 2.0440</td><td rowspan="2">Diagnostic Odds Ratio (DOR) = LR+/LR- =2.0440/0.2143= 9.5380</td><td rowspan="2">F1 Score = 2 / {(1/Recall) + (1/ Precision)} =2/ {(1/0.8857) + (1/ 0.8158)} = 0.8493 = 0.85</td></tr>
<tr><td>Negative Likelihood Ratio (LR-) = FNR/TNR = 0.1143/0.5333= 0.2143</td></tr>
</table>

Based on the above calculations and statistics, The system has a recall (sensitivity) value of 0.8857 and a precision value of 0.8158. The system has scored a rounded F1 score of 0.85.

5.2 Discussion

As mentioned above, the wheel detection and ellipse fitting modules were evaluated against a data set that contains 100 images. According to the results of the evaluation, the system is performing well with a higher accuracy level.

The wheel detection and ellipse fitting modules were initially developed based on a Deformable Parts Model (DPM) and the accuracy levels were checked. The DPM model consisted of a root filter and several part filters to identify the potential patches where the bicycle and the wheels were visible. The potential patches identified by part filters were further processed in order to fit the ellipses. The level of accuracy recorded with this method was extremely low. Hence it was decided to follow a CNN based approach and evaluate the accuracy separately.

With the newly proposed CNN approach, ResNet101 architecture was implemented. But a comparison could've performed if the system was developed under two different architectures such as ResNet50 and ResNet101.

Currently the system has been trained only for complete wheels, but this sort of reduced the overall accuracy by increasing the false positive rate. The system should be further improved by training with images where wheels are partially visible. That will help in increasing the accuracy levels further. Also, the system should be trained with additional features in order to identify the front and back wheels separately. Currently all the wheels available on the given picture will be identified and the relevant ROI coordinates should be fed to the ellipse fitting module. This process can be further improved by training the model with additional features which will help to identify the front and back wheels separately.

Furthermore, object detection and classification can be performed by several other approaches that belong to non-region-based categories. The selection between region proposal-based approaches and non-region-based approaches needs to be done after carefully evaluating the limitations and advantages of both approaches. You Only Look Once (YOLO) is a very promising algorithm that falls under non-region-based approaches. It consumes low computational power compared to region-based approaches and also it is faster [9]. But when compared with Faster R-CNN methods, YOLO showed less accuracy when identifying objects which are small and closer to each other due to its limited anchor boxes. In this research work, the considered objects are specifically limited to cycle wheels and usually the wheels appear small and closer to each other. Therefore, the accuracy drop in YOLO was a significant hit to the overall success of the project. Hence a region-based approach was used.

CHAPTER 6

CONCLUSION

Cycling safety has become an emerging research area due to the higher number of people moving towards using bicycles for their day today short distance travels. Cyclists and pedestrians are considered as vulnerable road users due to the higher number of accidents that they tend to face. With that being said, it has become very important to come up with collision mitigation systems focusing on both pedestrians and cyclists. When studying about the literature related to the research problem, it was noticed that many research have been carried out focusing on pedestrians and their safety, yet very less number of studies/ researches are there focusing on cyclist safety. Therefore, it was crucial to address the problem and provide a solution.

Most of the existing studies which addressed the issue on cycling safety had many constraints such as demographical issues, clothing patterns, etc. Therefore this research provides a solution to overcome the gaps between current studies.

The research shows of an accuracy pattern of 72% while scoring a F1 score of 0.85. These statistics prove that the suggested methodology is performing well. If further improvements are done on the this methodology, even better accuracy levels could be achieved in future.

6.1 Research Contribution

In this study, a thorough investigation has been conducted to provide a complete solution for the problem addressed. There are various ways of detecting cyclist and wheels based on different techniques. But this research presents a novel approach by combining a convolutional Neural Network to a mathematical model. The overall research solution provides a good F1 score proving that the system is reliable and accurate. The proposed methodology is completely new from the existing solutions, and it certainly addresses most of the limitations and gaps available in existing literature. It has added a considerable contribution to the research area that addresses cyclist safety. In addition to that, this research aims to achieve high performance given the fact that real time, quick processing is required to prevent or mitigate collisions in real life.

Furthermore, an entire dataset was created to cater the requirements of the research study. This dataset would be very beneficial to the fellow researchers who would like to continue studies

related to cyclists and wheels. The dataset was created based on Tsinghua-Daimler Cyclist Detection benchmark dataset. The wheels of the cyclists were annotated by a third-party annotator tool. This entire process cost a considerable amount of time from the project duration.

6.2 Limitations of the approach

One of the limitations of the system is that the training with the proper annotated dataset has not been done for all layers. If the middle layers were also trained with the annotated dataset, a higher accuracy level, F1 score could've been achieved.

Furthermore, the current system will only notify the automobile drivers that the cyclist is in a collision path or not. It would've been better if the cyclist path projection could be given to the automobile drivers real-time. It may prevent lot of collisions happening between vehicles and cyclists.

6.3 Further work

During the implementation, it was required to create an annotated dataset to train the MASK R-CNN model. But due to the time constraints and the project scope, it was difficult to create a dataset which has tens of thousands of images. If the model was trained over many images, the results would've been high in accuracy. In addition to that the Mask R-CNN model was a pre trained model based on COCO dataset. But if it was initially trained on Tsinghua-Daimler Cyclist Detection benchmark dataset, which focuses only on cyclists, the results would've been high in accuracy.

It would've been better if the Mask R-CNN model was developed on different ResNet variants, which would allow more opportunities for validation. In addition to that it would allow future researchers to select the best CNN model and continue their research work focusing on a different angle of the research problem.

As for another future work, the input images should undergo a proper extensive pre-processing in order to improve the accuracy of the system. Introducing a suitable and promising preprocessing technique to the system would increase the accuracy of the entire system in a great deal.

In addition to that the new dataset must be expanded with more images with wheel annotations. Wheel annotation details must be in JSON format. One of the most useful future works would be to publish the data set and make it public for academic purposes. This would be really helpful for lots of researchers out there who would be interested in a similar research area.

Next item on the future work list is to enhance the collision avoidance algorithm by adding the vehicle velocity vector. It would help a lot in identifying and predicting the cycle position ahead of a time with respect to the position of the vehicle. In addition to that, it is expected to implement the research work proposed by Tohid Ardeshiri et al [18] and integrate it to the system to develop an end to end system with an accurate cyclist state prediction system.

Finally, the use of FPN and RPN sometimes may affect the system performances. If an alternative approach was tried out, the performances could’ve been increased. Use of FPN usually consumes a considerable amount of time and memory. Therefore, FPN is said to be better only if the speed of the process is not an issue. But since this study includes real time processing, it might be safer to try out a different approach with the intention of replacing FPN module.

References

[1] S. Ahmed, M. N. Huda, S. Rajbhandari, C. Saha, M. Elshaw, and S. Kanarachos, “Pedestrian and Cyclist Detection and Intent Estimation for Autonomous Vehicles: A Survey,” Applied Sciences, vol. 9, no. 11, p. 2335, Jun. 2019.

[2] World Health Organisation. Global Status Report on Road Safety 2015—Summary; WHO: Geneva, Switzerland, 2015.

[3] Toroyan, T.; Peden, M.M.; Iaych, K. Supporting a decade of action. World Health Organisation 2013, 1, 318, doi:10.1258/jrsm.2010.090426.

[4] Bieshaar, M.; Reitberger, G.; Zernetsch, S.; Sick, B.; Fuchs, E.; Doll, K. Detecting Intentions of Vulnerable Road Users Based on Collective Intelligence. arXiv 2018, arXiv:1809.03916

[5] Robineau, D. Reported Road Casualties Great Britain, Annual Report 2017; Department for Transport: London, UK, 2017.

[6] A. Masalov, J. Ota, H. Corbet, E. Lee and A. Pelley, "CyDet: Improving Camera-based Cyclist Recognition Accuracy with Known Cycling Jersey Patterns," 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 2143-2149, doi: 10.1109/IVS.2018.8500668.

[7] D. Woods, “Volvo car group reveals world-first cyclist detection with full auto brake in geneva,” March 2013, https://www.media.volvocars. com/ca/fr-ca/media/pressreleases/48219.

[8] Y. Gu, Z. Shao, L. Qin, W. Lu and M. Li, "A Deep Learning Framework for Cycling Maneuvers Classification," in IEEE Access, vol. 7, pp. 28799-28809, 2019, doi: 10.1109/ACCESS.2019.2898852.

[9] S. Ahmed, M. N. Huda, S. Rajbhandari, C. Saha, M. Elshaw, and S. Kanarachos, “Pedestrian and Cyclist Detection and Intent Estimation for Autonomous Vehicles: A Survey,” Applied Sciences, vol. 9, no. 11, p. 2335, Jun. 2019.

[10] Y. Gu, Z. Shao, L. Qin, W. Lu and M. Li, "A Deep Learning Framework for Cycling Maneuvers Classification," in IEEE Access, vol. 7, pp. 28799-28809, 2019.

[11] Fornaciari, Michele & Prati, Andrea & Cucchiara, Rita. (2014). A fast and effective ellipse detector for embedded vision applications. Pattern Recognition. 47. 3693–3708. 10.1016/j.patcog.2014.05.012.

[12] A. Y. Chia, S. Rahardja, D. Rajan and M. K. Leung, "A Split and Merge Based Ellipse Detector With Self-Correcting Capability," in IEEE Transactions on Image Processing, vol. 20, no. 7, pp. 1991-2006, July 2011, doi: 10.1109/TIP.2010.2099127.

[13] X. Li, F. Flohr, Y. Yang, H. Xiong, M. Braun, S. Pan, K. Li and D. M. Gavrila. A New Benchmark for Vision-Based Cyclist Detection. In Proc. of the IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, pp.1028-1033, 2016.

[14] Janai, Joel & Guney, Fatma & Behl, Aseem & Geiger, Andreas. (2017). Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. Foundations and Trends® in Computer Graphics and Vision. 12. 10.1561/0600000079.

[15] Montella, Corey. (2011). The Kalman Filter and Related Algorithms: A Literature Review.

[16] Oppenheim, Georges & Philippe, Anne & Rigal, Jean. (2008). The particle filters and their applications. Chemometrics and Intelligent Laboratory Systems. 87-93. 10.1016/j.chemolab.2007.09.010.

[17] D. Bernstein, editor. Special Issue on Active Safety, volume 30. Control Systems Magazine, IEEE, aug. 2010

[18] T. Ardeshiri, F. Larsson, F. Gustafsson, T. B. Schön and M. Felsberg, "Bicycle tracking using ellipse extraction," 14th International Conference on Information Fusion, 2011, pp. 1-8.

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Proceedings of the 18th Conference on Computer Vision and Pattern Recognition (CVPR 2005), San Diego, CA, USA, volume 1 of IEEE, pages 886 –893 vol. 1, 2005

[20] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In Proceedings of the 22nd Conference on Computer Vision and Pattern Recognition (CVPR 2009), Miami, Florida, USA., IEEE, pages 304 –311, 2009

[21] H. K. Yuen, J. Illingworth, and J. Kittler. Detecting partially occluded ellipses using the Hough transform. Image Vision Computing, pages 31–37, 1989.

[22] Dutta, Abhishek and Andrew Zisserman. “The VGG Image Annotator (VIA).” ArXiv abs/1904.10699 (2019): n. pag.

[23] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[24] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 1, pp. 142-158, 1 Jan. 2016, doi: 10.1109/TPAMI.2015.2437384.

[25] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[26] Kovesi, P.: Edge linking and line segment fitting

[27] Pu, J., Zheng, B., Leader, J.K., Gur, D.: An ellipse-fitting based method for efficient registration of breast masses on two mammographic views. Med. Phys. 35(2), 487– 494 (2008)

[28] Benali Amjoud A., Amrouch M. (2020) Convolutional Neural Networks Backbones for Object Detection. In: El Moataz A., Mammass D., Mansouri A., Nouboud F. (eds) Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science, vol 12119. Springer, Cham. https://doi.org/10.1007/978-3-030-51935-3_30

[29] Tong, Yan & Lu, Wei & Deng, Qin-qin & Chen, Changzheng & Shen, Yin. (2020). Automated identification of retinopathy of prematurity by image-based deep learning. Eye and Vision. 7. 40. 10.1186/s40662-020-00206-2..

[30] Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

[31] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[32] T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.

[33] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[34] Prasad, D.K., Leung, M.K., Cho, S.-Y.: Edge curvature and convexity based ellipse detection method. Pattern Recogn. 45(9), 3204–3221 (2012)

[35] X. Li et al., "A Unified Framework for Concurrent Pedestrian and Cyclist Detection," in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 2, pp. 269-281, Feb. 2017, doi: 10.1109/TITS.2016.2567418.

[36] Z. Li, W. Wang, P. Liu, J. Bigham and D. Ragland, "Modeling Bicycle Passing Maneuvers on Multilane Separated Bicycle Paths", Journal of Transportation Engineering, vol. 139, no. 1, pp. 57-64, 2013. Available: 10.1061/(asce)te.1943-5436.0000480.

[37] G. Grigoropoulos, N. Khabibulin, A. Keler, P. Malcolm and K. Bogenberger, "Detection and Classification of Bicyclist Group Behavior for Automated Vehicle Applications," 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), 2021, pp. 1883-1889, doi: 10.1109/ITSC48978.2021.9564548.

[38] P.D. Kovesi, MATLAB and Octave Functions for Computer Vision and Image Processing (2000 ed.) (/http://www.csse.uwa.edu.au/pk/Research/MatlabFns/ index.htmlS).