Master Computer Science

A camera based Safety Assist System for bicycles

Name:	Wytze P. Breukel
Student ID:	2991918
Date:	28/10/2023
Specialisation:	Advanced Computing and Systems
1st supervisor:	Dr. E.M. Bakker
2nd supervisor:	Prof.dr. M.S.K. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

This paper describes and tests a novel camera-based active safety system for bicycles called Safety Assist System (SAS). The goal of SAS is to warn cyclists about overtaking and/or fast-approaching vehicles. There is currently little research into safety systems for bicycles and none on the market that use cameras. The goal is to create an effective system that is robust enough to be mounted on a bicycle. The SAS hardware consists of an embedded GPU platform to which a RGB camera and haptic feedback actuators are connected. The embedded system is capable of running the SAS vision model on a captured stream of images. The SAS vision model consists of two parts: an object detection model based on YOLOv5, fine-tuned with the KITTI dataset to recognise road users (cyclists, cars and pedestrians), and an Entity Tracking algorithm which uses these detections to determine if the user needs to be warned of the current traffic situation. If a warning is necessary the haptic actuators located in the handlebars of the bicycle will vibrate. The vision model is shown to offer acceptable performance compared to the state of the art on the KITTI dataset while keeping a low interference speed. Experiments on the Entity Tracking algorithm exhibit performance below the state of the art. To validate the SAS a prototype has been constructed and volunteers have been asked to follow a predetermined route through real-life traffic situations. As far as we know this is the first real-life experiment performed with a camera-based safety system on a bicycle. The user experiments showed that SAS performs well at recognising dangerous situations and is intuitive to use, but gives too many false positives to be useful in its current form.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. E.M. Bakker for his invaluable advice, large amount of useful feedback, and interesting and entertaining conversations about teaching, robotics, technology and other topics during our meetings. I am also extremely grateful to my friends, parents and especially my girlfriend for supporting and motivating me during this process. Special thanks to every participant who participated in the experiments, without them I would not have been able to test the SAS in a real-life environment. The LIACS master thesis support group led by Drs. A. Blank was also a big help by providing feedback and sharing thesis tips with fellow master students. Furthermore, I would like to thank William Corsel for providing tips to get YOLO and TensorRT correctly running on the Jetson Nano. Lastly, I want to express my gratitude to my employer the University of Applied Sciences in Leiden for being flexible and providing me with the time and space to work on this thesis.

1 Introduction

Using a bike for transport is extremely common in the Netherlands [1]. It has many health benefits and the potential to reduce traffic congestion. However, it also comes with safety risks because a cyclist is almost completely unprotected and more vulnerable to accidents compared to, for example, a car [2]. Almost all bicycle accidents involve some sort of collision with another vehicle. These accidents can be divided into several types with varying severity [3]. Research has shown that collisions with motorcycles are most common for cyclists, followed by ones involving trucks [4]. If one of these accidents results in an ER admission the mortality rate is 5.7% in the Netherlands [5].
In recent years there has been an increase in the usage of safety systems, which have as goal to improve road safety and prevent accidents. Developing these systems is becoming a large field of research. Right now there are several different systems available for vehicles such as cars or motorbikes. However, there are very few systems on the market which can be used on bicycles. These systems are run on embedded computers in vehicles and can either warn the users of incoming danger or directly intervene. Some of these safety systems use techniques like machine learning to detect danger after being trained on datasets of traffic situations [6].
The goal of this paper is to design and test a safety assist system for bicycles which will be called the Safety Assist System (SAS). We hope that this system can have a positive effect on the road safety of the users, by warning them of vehicles approaching from behind. It will achieve this by using a camera to capture images of the traffic situation behind the user. These images will be analysed for possible dangers and, if necessary, the SAS will warn the users via haptic feedback.
Although there are simpler solutions for reducing the danger of vehicles approaching from behind, i.e., a rearview mirror, SAS's biggest advantage lies in the fact that it actively warns the user. This means that user does not have to pay attention to have the SAS effectively warn them.
The SAS uses a vision model that is created specifically for analysing traffic situations which are dangerous for cyclists. It consists of two parts: a machine learning detection model based on YOLOv5 and fine-tuned on vehicle detection, and an algorithm to determine if the current traffic situation is dangerous for the user.
Since the SAS will be installed on a bicycle there are limitations on the amount of battery power available. Also, the hardware used needs to be compact, lightweight and robust enough to endure shocks encountered while cycling. As a consequence the computing power is limited. And expensive and/or large sensors are not suitable. Therefore, we will be aiming to minimise the complexity and the cost of the SAS. This means that there are no expensive sensors used like Lidar. This brings some challenges in the area of accuracy and response time of the warnings.
To evaluate the performance and the rider acceptance of the SAS a prototype has been constructed and tested with volunteers. These volunteers have used the SAS in real-life traffic situations and reported their experiences. They have also been asked about the ease of use and their feelings when using the SAS.
This paper is structured as follows: Section 2 discusses the related work and the position this paper takes relative to the field. Then Section 3 covers the fundamental metrics used to evaluate the experiments in this paper. Section 4 discusses the baseline of the vision model: YOLOv5. The vision model is then explained in detail in Section 5. The complete design of the SAS is detailed in Section 6. This design is tested in Sections 7 and 8 which contain

experiments about the feeling of safety the SAS provides, and the performance of the SAS in real-life scenarios respectively. Conclusions about the SAS can be found in Section 9, and finally, ideas for future work are covered in Section 10.

2 Related work

Safety systems for bicycles There has been previous research into developing a safety system for a bicycle. One approach uses custom laser and sonar sensors to detect cars approaching from behind and the right-hand side of the bicycle while minimising the cost, size and weight [7]. Other research looked into combining audio and video data to warn users of vehicles approaching from behind [8]. Another approach uses radar to detect cars behind the cyclist [9]. A different system using LIDAR focuses on having low-cost components [10]. However, these approaches have only been tested in simulations or lab-controlled environments where they performed well, although performance constraints were an issue. An approach that was tested in the field used a smartphone to warn motorists if they are endangering the cyclist [11]. This approach focused even more on low-cost components by using a smartphone and an off-the-shelf speaker. The prototype achieved a 95% accuracy in detecting vehicles approaching from the left side. Compared to this prototype SAS is focusing on detecting vehicles which are approaching from behind, furthermore, SAS will also be able to differentiate between cars cyclists and pedestrians.

Safety systems for powered two-wheelers
The field of safety systems for Powered Two Wheelers (PTW) displays a lot of similarities with bicycles and is therefore relevant to our research. In this field, there are two categories of systems: one type of system aims to warn the rider (e.g. collision warning, curve warning), and the other type directly controls the PTW to prevent danger (e.g. ABS). [6]. The SAS will not be actively intervening to prevent danger to users, thus falling under the first category. All the vision based systems desrcibed in [6] are detecting danger only to the front of the user while the SAS will be detecting danger approaching from behind the user.

Autonomous driving
The field of autonomous driving is quite large and active, addressing various challenges such as localization, perception and human-machine interfaces [12], which makes it interesting for the development of safety assist systems. Earlier approaches to the problem of automated driving were often modular in which the task was split up into different parts. For example, separate research went into using a GPS and other sensors to perform vehicle localization [13]. Other research looks into using LIDAR point clouds to perform 3D object detection to recognise vehicles [14]. Recently there has been a switch to a more end-to-end approach to address the entire task using deep learning methods [15]. Although these approaches are very interesting they are often too complex to use for the SAS. Furthermore, automated responses to dangerous situations are also not in scope for this paper.

SAS sensors
The most common sensor types used in existing safety systems are Radar, Lidar or stereo cameras (or a combination of these sensors). For PTW stereo cameras seem to be the best option to use due to the tilting nature of these types of vehicles [16]. A system which uses

stereo cameras is proposed in [17] where they are utilised for ABS on motorcycles. Another example is [18], which is an automatic lane change assist system that utilises stereo cameras combined with deep reinforcement learning to recognise vehicles. When the system detects a dangerous situation during a lane change maneuver it issues a visual and haptic warning to the user. This is a similar approach to the one the SAS will be taking. However [18] is reliant on lane markings on the road and is only tested in simulations.
In [19] the authors propose a method for detecting objects in the blind spot of cars by using cameras in combination with a Digital Signal Processor (DSP). This DSP transforms the 2D images into 1D signal information and compares it to the information of previous images to try and track approaching vehicles. It can do this with a 91% accuracy in varying weather conditions. For PTWs, there have been studies focusing on using a Lidar setup complemented by an Inertial Measurement Unit (IMU) for a Simultaneous Localization and Mapping (SLAM) based approach. However, real-time object detection using Lidar remains a challenge for these systems [20].
Another possible approach for designing a safety assist system for cyclists is the usage of a GPS built into the cyclists' smartphones. However, the authors found that at the time of research the accuracy of the smartphones used was too low for safety applications [21]. The SAS is constrained by cost and the need to be robust enough to be placed on a bicycle. Custom DSPs or expensive sensors such as LIDAR are therefore not suitable. Instead, the SAS will be using a mono RGB camera.

Rider acceptance of the SAS
When designing a safety system it is important to ensure that users want to use it. There has been research into predicting and evaluating the acceptance of rider assistance systems on motorcycles. It has shown that social norms and the interface design are significant factors in predicting if users will use a rider assistance system [22]. Other factors that improve the acceptance of rider assistance systems are how large the user perceives the risk of riding and how little the task interferes with the riding itself [23]. Another study focusing on comparing different types of interfaces concluded that users prefer a haptic interface which utilises vibrations to notify users over a visual or audio interface [24]. Haptic feedback minimises distractions and can be built into the handle of a PTW [25]. In our SAS system haptic feedback is built into the handlebars of the bicycle. There has also been research focusing on creating a smart bike to support its rider both cognitively and physically, by helping the rider pass green lights [26].

SAS related vision models
Visual Object detection has been a very active field of research these past decades with impressive advances in the last decade due to the application of DNNs. For an embedded safety system on a bicycle one requirement for a visual object detection DNN is extremely important: It needs to be able to perform real-time interference on limited hardware. Dangerous road users need to be detected quickly enough that the user has time to respond. This means that most of the state-of-the-art object detection DNNs are not suitable. An example of one lightweight (and thus scaleable) object detection DNN is EfficientNet [27], which uses a new dimension scaling method to create a new family of neural nets. Another family of object detection models that focuses on interference speed and small size is the You Only Look Once (YOLO) family. Various versions of YOLO have been developed by different researchers [28]. The more recent variants include YOLOv4 [29] and YOLOv5 [30]. Each version improves on

the previous one in some aspect or adds extra features. One of the advantages of the YOLO family is that they are easily re-trainable on different classes.
It was decided to use YOLOv5 for the SAS due to the low computational resources required, the rapid interference speed and the support for re-training. The current model competitive with the state-of-the-art in the YOLO family is YOLOv8 [31] [32] which is created by the same developers [33] as YOLOv5. YOLOv8 has a mAP score which is 3% higher than YOLOv5 on the MS COCO dataset [31]. And has been used for various tasks such as fire detection [34] or detecting objects from the perspective of an unmanned aerial vehicle (UAV) [35]. Furthermore, YOLOv8 has inbuilt object tracking support. However, at the start of this project YOLOv8 was not yet released.

Object tracking
Other sets of vision models are trained on the tracking of objects in videos, which is called MOT (multiple object tracking). This field is widely researched and there are several different approaches to solve the associated problems. Most of the research into MOT focuses on tracking pedestrians [36]. Tracking cars is also a largely studied subfield in MOT, for example, to solve problems like tracking traffic congestion [37].
One of the problems encountered when performing MOT is the occlusion of the to-be-tracked object. There has been research that used YOLOv3 combined with an auxiliary tracker to mitigate this problem during autonomous driving tasks [38]. Another paper [39] uses a Markov decision process running on a Jetson TX2 embedded in a car to achieve real-time tracking of other vehicles. To evaluate the performance of such models the MOT challenge is used. This challenge consists of videos of crowded scenes combined with the ground truth to compare models accurately with each other [40].
The computational resources required to implement these object tracking models and the goal of the SAS to be as efficient and computationally light as possible result in these models being unsuitable for usage in this paper.

SAS related datasets
To train, compare and test vision models on traffic-related images, various research benchmark datasets have been created. One such dataset focuses on car pose estimation [41]. There are also several datasets which are captured in real-life traffic situations. For example [42] is a dataset captured in Málaga with stereo cameras and laser scanners. Another dataset [43] contains stereo camera images taken at night in Oxford to help models make more accurate predictions in dark environments. An additional dataset available for training and evaluation vision models is the KITTI dataset with 200k 3D object annotations [44]. For the training of the vision model in this paper the KITTI dataset has been selected for two reasons: It contains a large amount of relevant annotated images, And its the widespread use in the literature for training and evaluating state-of-the-art vision models for traffic-related tasks.

3 Fundamentals

This chapter discusses the metrics used in this paper to evaluate the data from the experiments.

3.1 Precision and accuracy metrics for the vision model

Several different metrics are used in the research literature to assess how well a vision model is performing. In this section each metric used in this paper is discussed. All of these metrics are calculated on the basis of three values: True Positives (TP), False Positives (FP) and False Negatives (FN).

$$Precision = \frac{TP}{TP + FP}$$

The goal of the precision metric is to determine how accurate all the detections of the model are, it ranges from 0 to 1. Therefore, for each false detection which the model makes the precision score will be lower [45].

$$Recall = \frac{TP}{TP + FN}$$

Recall signifies how many True Positives the model misses. The more True Positives are not detected, i.e., the higher the number of False Negatives, the lower the recall score. Once again ranging between 0 and 1 [45].

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

The F1 score can be used to mitigate the drawbacks of both precision and recall scores. This score is determined by using both the precision and the recall values. This results in a more realistic representation of the performance of a model, especially if the model has a very high precision score and low recall score or visa versa [45].

meanAP score A disadvantage of the F1 score is that it can only represent the performance of a model at a single confidence threshold. The confidence threshold signifies how confident (expressed as a percentage) the model needs to be in a possible detection to mark it as positive. This means that precision and recall vary based on the confidence threshold. If the confidence threshold is high, the precision will increase but the recall will suffer because detections which might have been correct are discarded. With a low confidence threshold, the recall will increase and the precision will decrease. This happens because detections that the model is less confident about will be marked as correct and these detections inevitably have a higher chance of being incorrect. The advantage of the meanAP (mAP) [46] score is that it can represent the performance of the model at different confidence thresholds. To calculate the mAP first the Average Precision for each class has to be calculated:

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n$$

Where:

R_n = Recall at nth confidence threshold
P_n = Precision at nth confidence threshold

Then the AP score of each class is added together and divided by the number of classes which results in the mAP score.

A way to further specify the performance of a vision model with the mAP is to incorporate different intersection over union (IoU) thresholds. The IoU signifies the percentage of the detected bounding box for the object that needs to overlap with the ground truth bounding box to be marked as correct. This is notated as a number after the mAP score. For example, a mAP-50 means that the IoU needs to be equal to or larger than 50% to be marked as correct.

3.2 Likert scale

To assess the performance of the SAS prototype a user-evaluating experiment has been set up. The user evaluations are done by using a Likert scale. This metric is chosen because it is a standardized way to gather subjective feedback on the quality of products [47]. Specifically a 7-point Likert scale will be used because various studies have shown that users prefer a 7-point scale because of, among other reasons, its greater granularity compared to a more limited 5-point scale [48] [49]. When evaluating the SAS the users will be presented with a statement about the functioning of the SAS and asked to choose one of the following options:

- Strongly disagree
- Disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat agree
- Agree
- Strongly agree

4 Baseline for the vision model: YOLOv5

This chapter discusses the object detection model used in the SAS vision model: YOLOv5. When starting this project, YOLOv5 was a competitive to the state-of-the-art vision model of the YOLO family (You Only Look Once) [50]. After conducting a literature review, YOLOv5 was chosen because of its very competitive trade-off options on inference speed, size and accuracy. Furthermore, it allows for effective transfer learning on new image classes.

4.1 YOLOv5 Supported Classes

YOLOv5 is trained on the COCO [51] dataset. This dataset is commonly used to train vision models and contains 91 different object categories, 82 of these have more than 5,000 labelled instances with a total of 2,500,000 instances spread over 328,000 images. Due to the traffic-oriented nature of the SAS, most of the object categories in the COCO dataset that YOLOv5 can detect are irrelevant. However, person, bicycle, car, motorcycle and bus are useful classes for the SAS and are part of the COCO dataset and therefore recognisable by the COCO pre-trained YOLOv5 model.

4.2 YOLOv5 pre-trained versions

YOLOv5 offers a few different pre-trained versions differentiated by the amount of parameters. Each version has a different trade-off between interference time and accuracy [50]. The performance of these pre-trained versions on the COCO dataset as provided by the developers of YOLOv5 can be found in Table 1. The SAS's hardware implementation has limited computational resources but the inference must be performed in real-time. Therefore, the pre-trained version used in the SAS needs to strike the right balance between accuracy and interference speed. Experiments to determine the best pre-trained version can be found in Section 5.3.

Pre-trained version	mAP 50	Speed (ms)	Parameters (M)
YOLOv5n	45.7	**6.3**	**1.9**
YOLOv5n6	54.5	8.1	3.2
YOLOv5s	56.9	6.4	7.2
YOLOv5m	**64.1**	8.2	21.2

Table 1: Performance of the relevant pre-trained versions provided by the developers [30] inference speed is measured on a Nvidia V100 tensor core GPU

4.3 YOLOv5 Transfer Learning

YOLOv5 also offers support for transfer learning with which the architecture of an existing pre-trained version can be leveraged to help train for a similar but new task. This technique has been used to train YOLOv5 on different classes not contained in the COCO dataset [52]. In the SAS vision model transfer learning is utilised to add a cyclist class to YOLOv5 and to improve its accuracy in traffic scenarios.

5 SAS Vision model

This section covers how the baseline YOLOv5 model, as described in Section 4, will be used to create a vision model specifically for the SAS. In Figure 1 a high-level overview of the SAS vision model is shown. This diagram describes the steps the vision model takes to assess if the user needs to be warned. The SAS vision model consists of two parts: a trained YOLOv5n model and the Entity Tracking algorithm. Both are described in more detail in the following section.
The first step to create the trained YOLOv5 model is to pick the most suitable pre-trained version offered by YOLOv5. For selecting the pre-trained version there are two important metrics: the speed at which the model can handle a single frame on the hardware used by the SAS, and the accuracy of the detections. These two metrics are mostly antagonistic to one another since versions with a higher accuracy often take longer to process a single frame (see Table 2).
To determine the best pre-trained version for the safety assist system two preliminary experiments are performed: The first one focuses on the speed of the model (see Section 5.3.1) and the second one on the accuracy of the predictions (see Section 5.3.2). Based on the results of the preliminary experiments the most suitable pre-trained version is determined.
YOLOv5 is pre-trained on the COCO dataset [51]. However, this dataset does not include a cyclist class, which is necessary for the functioning of the SAS. To solve this, and to improve

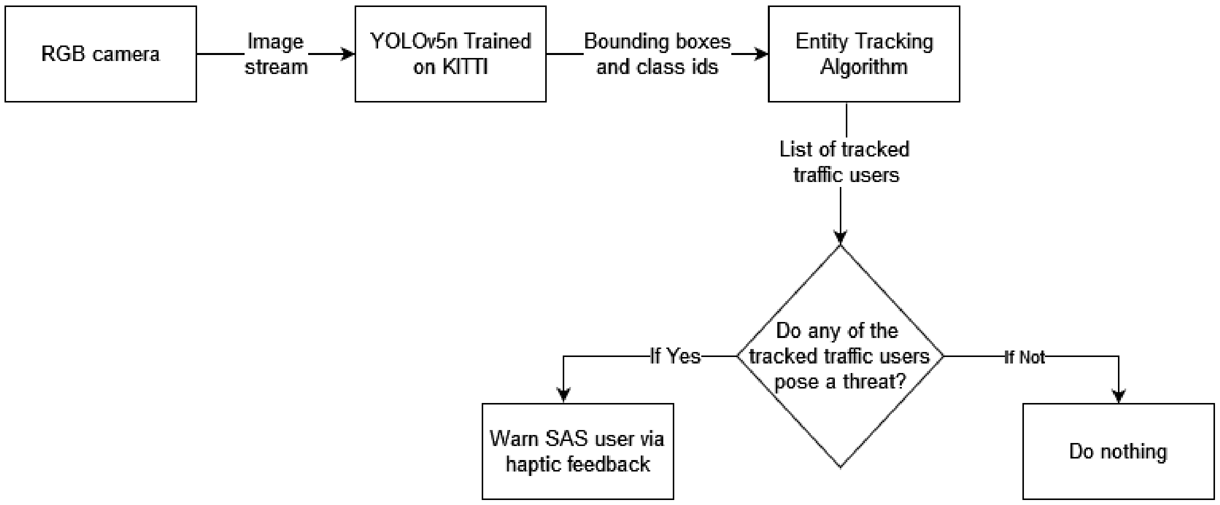
RGB camera
Image stream
YOLOv5n Trained on KITTI
Bounding boxes and class ids
Entity Tracking Algorithm
List of tracked traffic users
Do any of the tracked traffic users pose a threat?
If Yes
If Not
Warn SAS user via haptic feedback
Do nothing

Figure 1: A high-level overview of the SAS vision model

the accuracy of the detections, YOLOv5 is fine-tuned on the following classes: cars, pedestrians and cyclists. This is done by using the KITTI dataset [44].
Finally, to improve the speed of detection the trained model is optimised with TensorRT, which is described in more detail in Section 5.1.
When the SAS vision model has detected a traffic user it tries to determine if this traffic user poses a danger to the user of the SAS. In Section 5.6 we propose an Entity Tracker algorithm that uses the output from the trained YOLOv5 model to create a record of relevant traffic users behind the SAS user. It will then use this record to determine if the SAS user needs to be warned via haptic feedback.

5.1 TensorRT

To maximise the performance of the vision model TensorRT is used. TensorRT is an SDK developed by Nvidia which optimises the interference speed of a given trained neural network [53] at the cost of a relatively small loss of accuracy.
Various steps are used to improve performance such as post-training quantization and Floating Point 16 (FP16) optimizations. It supports the major machine learning frameworks, such as: Pytorch, TensorFlow, ONNX and MatLab. A use case of TensorRT is optimising vision models in embedded environments like the Jetson Nano used in the SAS. TensorRT is based on the Nvidia CUDA parallel programming model therefore, it only functions on Nvidia GPUs.

5.2 Dataset

The well-known KITTI vision benchmark suite [44] has been used for both training and evaluating the performance of the SAS vision model i.e. YOLOv5.
This dataset is chosen because it consists of images taken while driving in a medium-sized German city (Karlsruhe), which is comparable with the expected environment the Safety Assist System will be operating. The dataset is annotated by humans, and contains three classes: Pedestrians, Cyclists and Cars. Those are also the three classes the Safety Assist System needs to recognise. The dataset consists of 7481 training images and 7518 test images.

However, since the KITTI dataset will be used for both training and evaluating the SAS vision model, the entire dataset will be split into three parts: A training set which contains 80% of the images, a validation set containing 10% of the images and a test set containing the remaining 10% of the images. The splitting will be done by a custom Python script which will ensure that the ratio of classes in the subsets is approximately the same as in the total dataset.

5.3 Preliminary experiments

5.3.1 Preliminary experiment: interference speed of the model

The goal of this experiment is to measure the time it takes to perform interference on a single image with each of the relevant available YOLOv5 pre-trained versions. This experiment has been run on the hardware used in the SAS: the Jetson Nano (for specifications of the Jetson Nano see Appendix D).
To perform the experiment the to be tested pre-trained version is loaded into Pytorch, which is configured to use the GPU of the Jetson Nano. The model will then perform interference on a test image 100 times while logging the time it takes to complete the interference in milliseconds. All the results are collected and then summarised by using the median, which is converted to the amount of Frames Per Second (FPS). The median is used, because the first few results are significantly slower when using the GPU, so an average would skew the results. This is probably because that the model has to be loaded into the GPU. However, since the SAS runs in real time the warm-up phase has very little effect on the final implementation. The results of this experiment can be found in Table 2.

5.3.2 Preliminary experiment: Accuracy of the model

The goal of this experiment is to determine the accuracy of the YOLOv5 pre-trained versions in traffic situations.
The KITTI dataset (see Section 5.2) is used to compare the accuracy of each pre-trained version. This dataset contains three classes pedestrians, cyclists and cars. Since YOLOv5 is trained on the COCO dataset [51] (see Section 4), it does not contain a cyclists class (which consists of both the bicycle and the rider). However, the KITTI dataset does have a cyclist class. To mitigate this problem we used the bicycle class (which consists of just the bicycle) in the COCO dataset as a stand-in the for cyclist class. This means that the accuracy on the bicycle class will be very low since the ground truth of the KITTI dataset expects the entire cyclist to be marked (bicycle + rider) while YOLOv5 will only mark the bicycle during interference. However, since the goal of this preliminary experiment is to compare the YOLOv5 pre-trained versions with each other, only the relative scores are relevant. To evaluate the pre-trained versions the validation python code from YOLOv5 repository [50] is used.
The results of this experiment can be found in Table 2.

5.3.3 Preliminary experiment results

The results of the preliminary experiments are listed in Table 2. The table shows that the mAP50 scores of the different pre-trained versions are quite similar with a maximum difference in accuracy of 0.035 between the slowest and the fastest pre-trained version. However, the

Model	Person mAP50	Bicycle mAP50	Car mAP50	All classes mAP50	FPS on Jetson
YOLOv5n [30]	0.349	**0.0178**	0.603	0.323	**14.52**
YOLOv5n6 [30]	0.347	0.0116	0.583	0.314	14.08
YOLOv5s [30]	0.366	0.00658	0.627	0.333	7.56
YOLOv5m [30]	**0.371**	0.00707	**0.638**	**0.339**	3.62
YOLOv5n TensorRT [30]	0.333	**0.0167**	0.564	0.304	19.63
YOLOv5n6 TensorRT [30]	0.324	0.0133	0.541	0.293	**19.75**
YOLOv5s TensorRT [30]	0.368	0.0103	**0.623**	**0.334**	12.9
YOLOv5m TensorRT [30]	**0.389**	0.00916	0.604	**0.334**	7.07

Table 2: Comparison of performance of pre-trained versions ordered on the number of parameters

slower pre-trained versions have a small advantage in accuracy compared to the faster ones as expected.
However, the interference speed of the models varies significantly. It is also clear that converting to TensorRT gives a significant speed boost while sacrificing very little accuracy. Therefore it seems beneficial for the SAS to use a TensorRT version of YOLOv5.
Since the accuracy does not vary much between the models the deciding factor will be the FPS. The two fastest models are YOLOv5n TensorRT and YOLOv5n6 TensorRT, which respectively have 19.63 and 19.75 FPS. Comparing these two, the YOLOv5n has better accuracy so the decision was made to continue with the YOLOv5n TensorRT although the difference is very small.
To prevent any unforeseen accuracy drops, which may occur during training, YOLOv5s will also be trained. This way there is an alternative model to switch to if during testing accuracy has a more significant impact on the functioning of the SAS than expected. However, the main focus will be on the YOLOv5n TensorRT.

5.4 SAS Training

To increase the accuracy of the YOLOv5 at recognising traffic users the model is retrained using the KITTI dataset described in Section 5.2. To leverage the existing infrastructure of YOLOv5 we will be retraining from a pre-trained version, as recommended by the documentation [54]. To speed up the training process it is performed on a more powerful desktop machine than the Jetson Nano, which has a higher amount of available RAM and better CPU and GPU performance. Complete specifications of the desktop machine can be found in Appendix E. The training code provided by the YOLOv5 repository [50] was used with the following parameters: an image size of 640, a batch size of 16 and 100 epochs. The number of epochs was set at 100 because several trial runs had no significant improvement after the 100th epoch.

5.4.1 Training results

As seen in Table 3 the trained models are performing significantly better than YOLOv5 out of the box (as seen in Table 2). This is to be expected because both models are evaluated on a test set consisting of images from the KITTI dataset, but only our trained model is trained on a train set consisting of images from the KITTI dataset. YOLOv5 which is trained on the COCO dataset does not have this advantage. YOLOv5 has other disadvantages for example, as described previously it is not trained to detect a cyclist class but uses a bicycle class as a stand-in. Still, the significant improvements mean that the newly trained model will perform better at recognising traffic users compared to the YOLOv5 out of the box. In Table 4 a comparison per class shows that the models have the best accuracy when detecting cars and the worst accuracy when detecting pedestrians. For more in-depth training results see Appendix B. The YOLOv5s models have a slightly better accuracy score. However, since interference speed is a crucial factor for the SAS we decided to continue with the YOLOv5n TensorRT due to it scoring almost 10 FPS more than the YOLOv5s TensorRT model.

Model	Precision	Recall	mAP50	mAp50-95	FPS on Jetson
YOLOv5n trained on KITTI[30]	0.912	0.793	0.883	0.582	12.45
YOLOv5n TensorRT trained on KITTI[30]	0.883	0.797	0.875	0.567	**17.43**
YOLOv5s trained on KITTI [30]	**0.925**	**0.867**	**0.925**	**0.66**	5.24
YOLOv5s TensorRT trained on KITTI [30]	0.94	0.857	0.922	0.647	7.46

Table 3: Performance of trained YOLOv5 models on all classes

Model	Pedestrian mAP50	Cyclist mAP50	Car mAP50
YOLOv5n trained on KITTI[30]	0.831	0.855	0.979
YOLOv5n TensorRT trained on KITTI[30]	0.823	0.84	0.964
YOLOv5s trained [30]	0.879	**0.916**	**0.979**
YOLOv5s TensorRT trained [30]	**0.884**	0.906	0.978

Table 4: Performance of trained YOLOv5 models per class

After training the model, a confusion matrix has been generated (see Appendix I). It seems that the model is most likely to make a False Positive by classifying something in the background as a car. The model also seems to struggle with determining the difference between a pedestrian and the background.

5.4.2 Training results compared to the current state of the art

At the start of the project, YOLOv5 was the most recently released YOLO model. However, as mentioned in Section 2, during the research process various new versions of YOLO have been released with as latest YOLOv8 [33]. To compare the performance of our trained model to the state of the art we have trained a YOLOv8n model on the same dataset and with the same parameters as our YOLO5n model. The results can be found in Table 5.
There seems to be a negligible difference between the two models with YOLOv5n even having a 0.4% higher score when comparing the mAP50 all classes score. Therefore, we can conclude

Model	mAP50 Pedestrian	mAP50 Cyclist	mAP50 Car	mAP50 All classes
YOLOv8n trained on KITTI [33]	0.814	**0.861**	0.962	0.879
YOLOv5n trained on KITTI[30]	**0.831**	0.855	**0.979**	**0.883**

Table 5: YOLOv8n trained compared to YOLOv5n Trained

that the improvements added to YOLOv8 compared to YOLOv5 are not beneficial for the SAS.

5.4.3 Training results compared to competitive approaches

To compare the trained model to the current state of the art, two papers proposing a YOLO model trained on the KITTI dataset have been selected. Each of these models has been compared against our trained model.

model	mAP50 pedestrian	mAP50 Car	mAP50 Cyclists	mAP50 All classes
YOLOv5n TensorRT trained on KITTI	0.823	**0.964**	**0.84**	**0.876**
The Application of Improved YOLO V3 in Multi-Scale Target Detection [55]	**0.934**	0.767	**0.84**	0.847

Table 6: Comparing the trained model to the paper The Application of Improved YOLO V3 in Multi-Scale Target Detection [55]

The first paper [55] re-trains a YOLOv3 model on the KITTI dataset and improves the model by adding extra convolution layers. When comparing their results (See Table 6) our model performs better or equal in every class except pedestrians. Because the experiments conducted in [55] were performed on different hardware the interference speed could not be compared.

model	mAP50 All classes	FPS on the Jetson Nano
YOLOv5n trained on KITTI	0.883	12.45
YOLOv5n TensorRT trained on KITTI	**0.875**	17.43
Edge YOLO [56]	0.821	9.7
Edge YOLO TensorRT [56]	0.726	**25.5**

Table 7: Comparing the trained model to the paper Edge YOLO: Real-Time Intelligent Object Detection System Based on Edge-Cloud Cooperation in Autonomous Vehicles [56]

The second paper [56] proposes a modified version of YOLOv4 to create a new model called Edge YOLO. Edge YOLO is trained on the KITTI dataset to be used in Edge computing and utilises TensorRT to improve interference speed. Due to the fact that they used a Jetson Nano as Edge node we can compare both the accuracy and the interference speed of the different models. The results of this comparison can be seen in Table 7. Although the accuracy of the detections is similar in both non-TensorRT versions of the different models, Edge YOLO has a significant drop in accuracy of almost 10 % when converted to a TensorRT version. While our trained YOLO versions had only a 1% drop. However, Edge YOLO does have a significantly higher FPS score after utilising TensorRT: 25.5 vs. 17.43 on our trained YOLOv5 version.

5.5 Final YOLOv5 version

The SAS will be using a version of YOLOv5n which is optimised with TensorRT and trained on the KITTI dataset to detect road users. YOLOv5n was chosen because it offers a good trade-off between accuracy and interference speed. This interference speed is further improved by using TensorRT, which has a relatively small impact on the accuracy. To improve the detection accuracy in traffic situations it is trained on the KITTI dataset. When comparing the trained model to the state-of-the-art and comparative approaches from the literature it performs with comparable accuracy and interference speed.

5.6 Entity Tracker algorithm

The purpose of the Entity Tracker algorithm is to detect if a vehicle approaches the SAS user from behind and to determine if this creates a dangerous situation for the user.
To achieve this, the Entity Tracker algorithm analyses the output from the custom-trained YOLOv5 model described in Section 5.4. To effectively estimate if a road user approaches from behind the SAS needs to be able to track the road user (called an entity in the algorithm) between multiple frames. This is needed because to determine if an entity comes closer the SAS uses the difference in size of the entity between frames as a reference. The SAS assumes that if the size of an entity increases it is approaching.
The problem lies in the fact that the vision model works on a frame-per-frame basis so there is no explicit connection between the entities detected in each frame. To solve this proposed Entity Tracker algorithm tries to match the entities from the previous frames to the entities of the current one. This matching process consists of three steps. First, the model will discard any entity detections which are most likely false or not relevant to the SAS. It will then try to match the detected entities in the previous frames to the newly detected entities. And, finally, it will determine if the entity has come closer and might pose a threat to the SAS user. This whole process is modelled in the diagram in Figure 2. Pseudo code describing the entire Entity Tracker algorithm can be found in Algorithm 1. For a more detailed look at specifically the Entity Matching algorithm see Algorithm 2.

5.6.1 Entities

Each bounding box and class ID combination found by the trained YOLOv5n vision model is transformed into an entity in the form of a Python object. These entities contain the information the Entity Tracker algorithm needs as properties. Table 8 contains all the properties of an entity with a short description of its usage by the Entity Tracking algorithm. The pseudo-code for creating entities is found in the function CreateEntitiy in Algorithm 1.

5.6.2 Entity Validity Checker

The purpose of the Entity Validity Checker is to filter out any false detections or non-dangerous entities to prevent confusion for the SAS vision model.
During testing, it was noticed that the SAS reported a significant amount of False Positives which were caused by entities that were wrongly detected or were in positions from which they

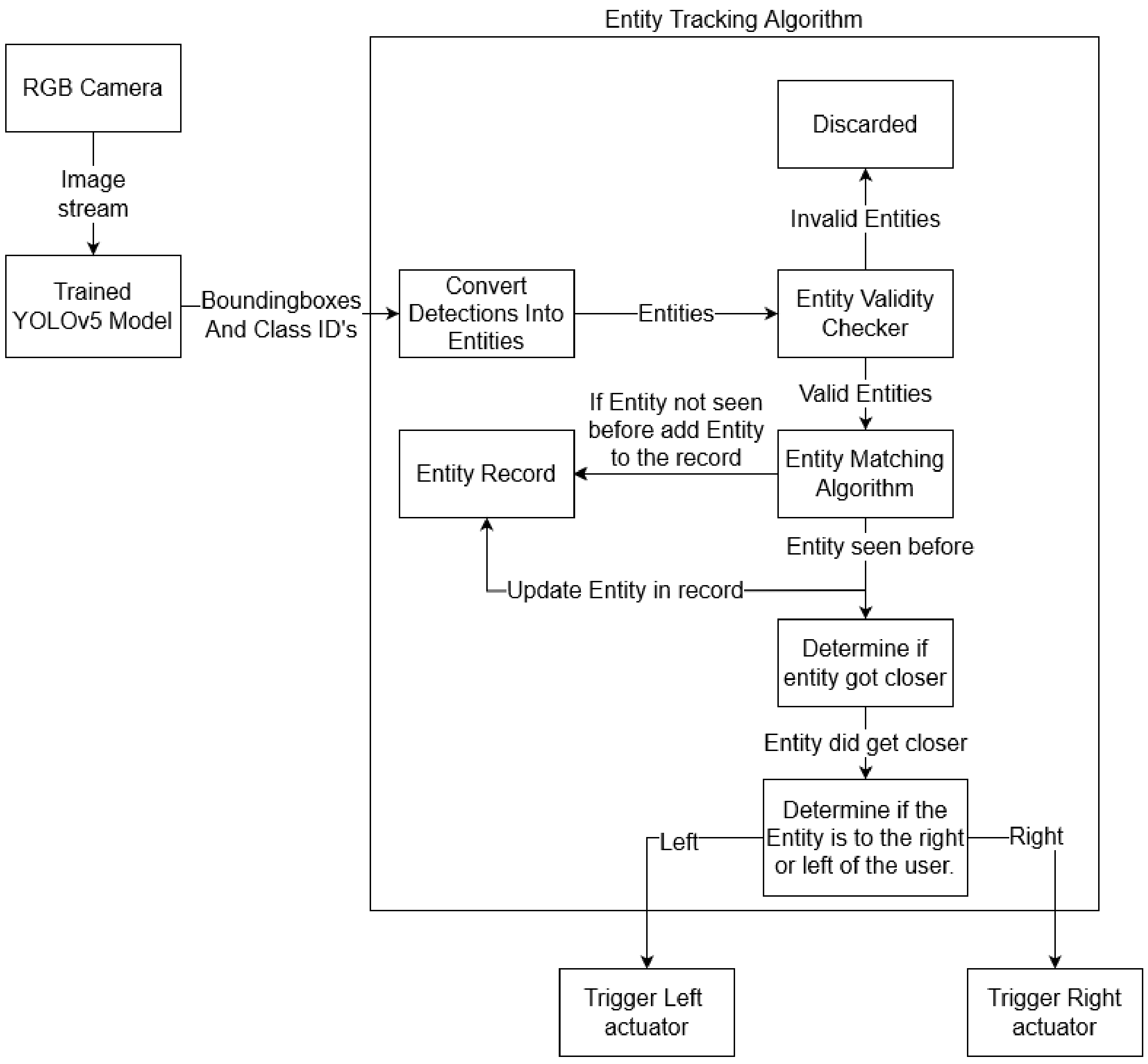
Entity Tracking Algorithm
RGB Camera
Image stream
Trained YOLOv5 Model
Boundingboxes And Class ID's
Convert Detections Into Entities
Entities
Entity Validity Checker
Invalid Entities
Discarded
Valid Entities
Entity Matching Algorithm
If Entity not seen before add Entity to the record
Entity Record
Entity seen before
Update Entity in record
Determine if entity got closer
Entity did get closer
Determine if the Entity is to the right or left of the user.
Left
Right
Trigger Left actuator
Trigger Right actuator

Figure 2: The Entity Tracker Algorithm

Algorithm 1 Entity Tracking algorithm

```
function EntityTrackingAlgorithm
    $Image \leftarrow CaptureImageFromRGBCamera$
    $BoundingboxesAndClassIDs \leftarrow TrainedYOLOv5n(Image)$
    for <Every BoundingboxAndClassID in BoundingboxesAndClassIDs> do
        $Entities \leftarrow CreateEntity(BoundingboxAndClassID)$
    end for
    for <Every Entity in Entities> do
        if $CheckIfEntityIsValid(Entity)$ then
            $ValidEntities \leftarrow Entity$
        end if
    end for
    $ApproachingEntities \leftarrow EntityMatchingAlgorithm(ValidEntities)$
    for <Every ApproachingEntitiy in ApproachingEntities> do
        WarnUser($ApproachingEntitiy$)
    end for
end function

function CreateEntitiy($BoundingBoxAndClassID$)
    $Size, Position, DetectedClass, Height \leftarrow BoundingBoxAndClassID$
    return $Enitity \leftarrow Size, Position, DetectedClass, Height$
end function

function CheckIfEntitityIsValid($Entity$)
    if $Entity.Size < MinimumSizeForDetectedClass$ then
        return $False$
    end if
    if $Entity.Position$ outside right or left border then
        return $False$
    end if
    if $Entity.Width/Entity.Height > MaximumWidthHeightRatio$ then
        return $False$
    end if
    return $True$
end function
```

posed no danger to the user. To filter out these entities several rules were created based on footage made during testing. If an entity does not conform to all the rules it is discarded by the Entity Tracking algorithm.
The first rule checks that each entity has a minimum width in pixels, this minimum differs for each class. This rule serves two purposes: Firstly most of the False Positives the vision model makes are very small, and secondly, if the entity detected is very far away it most likely does not pose a threat to the SAS user.
The second rule the entity needs to conform to focuses on the position of the entity. During testing parked cars gave a large number of False Positives, due to the vision model confusing two cars parked next to each other as one moving car. See figure 3 for an example. To mitigate

Property name	Description	Used for:
position	The last recorded position	Used to match entities across frames and check if the entity is valid
id	An unique number	Used to identify entities in the memory of the algorithm
size	The width of an entity in pixels	Used to detect if an entity is approaching and to check if an entity is valid
amount_of_detections	The number of times a specific entity has been tracked	Used for logging
detected_class	The class in which the entity has been classified.	Used to match entities across frames and check if an entity is valid
frames_to_live	The number of frames the entity can not be detected until it is removed from the record	Used for matching entities
height	The height of an entity in pixels	Used to check if an entity is valid

Table 8: Proprieties of the entity

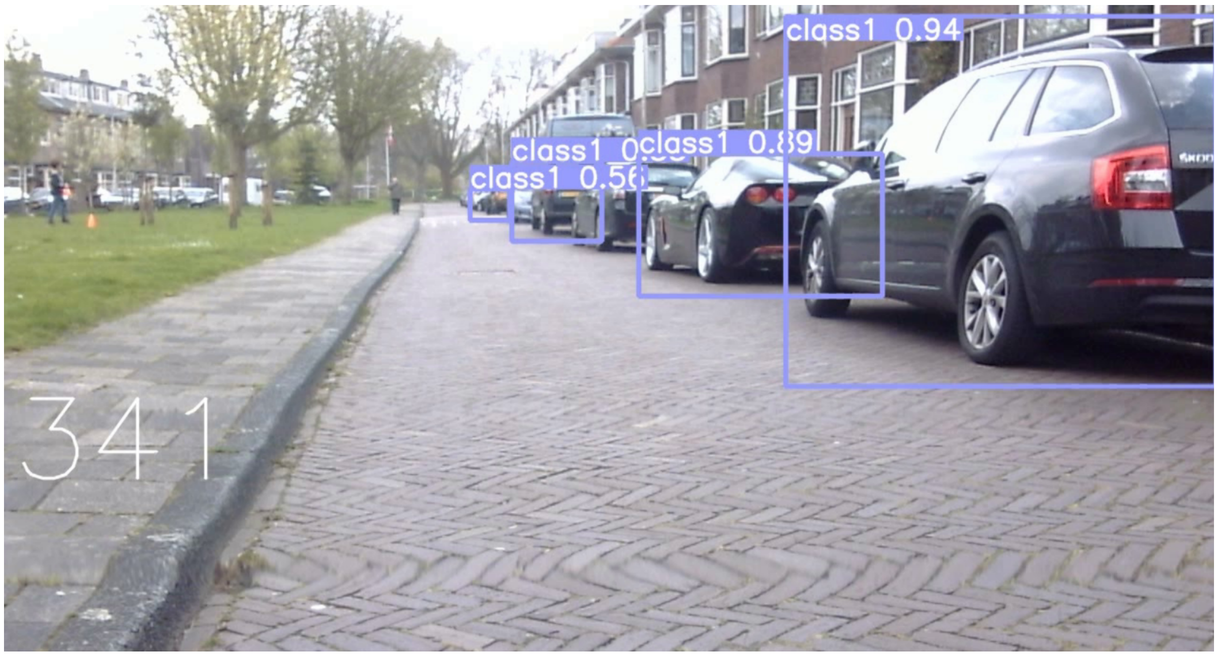

Figure 3: Parked cars that can confuse the SAS vision model

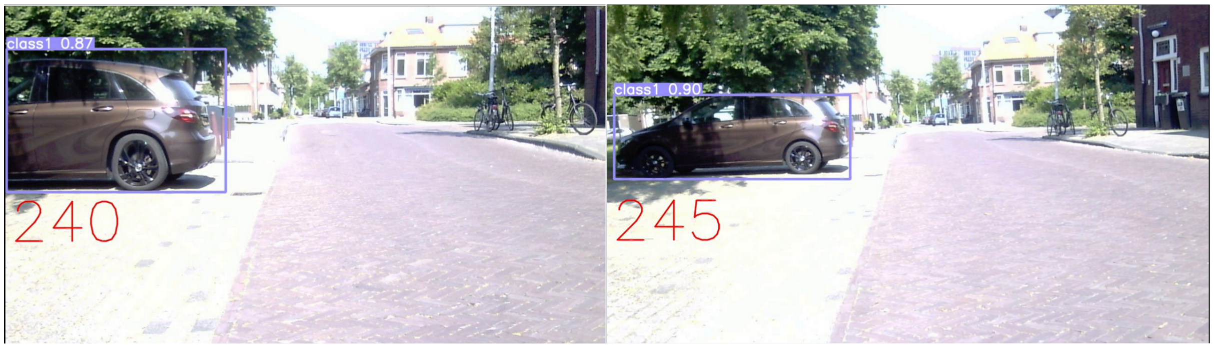

Figure 4: Problematic situation with sideways parked cars

this problem a rule is created that ensures that the detected entity is in a location from which it could pose a threat to the user of the SAS.
Lastly, there was a problem with evaluating the cars which were parked sideways from the perspective of the SAS. Because the SAS uses the width of an entity to determine if it approaches the SAS or not, a sideways car which is half in view in the first frame will appear "larger" in the subsequent frames. For an example see Figure 4. To solve this issue, each detected entity's width-by-height ratio needs to be below a threshold. This ensures that entities with are very wide relative to their height will be discarded. These rules are implemented in the CheckIfEntityIsValid function found in pseudo-code in Algorithm 1

5.6.3 Entity Matching algorithm

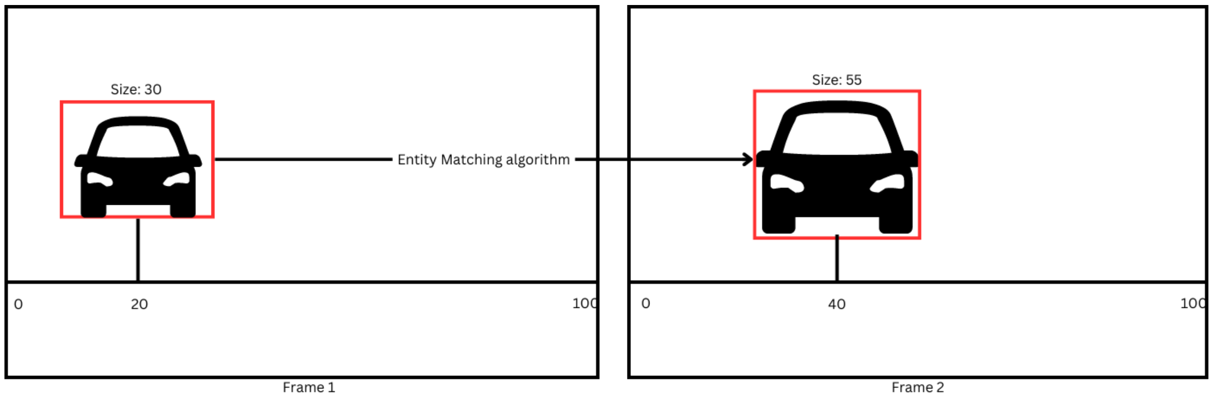

Figure 5: Schematic view of the working of the Entity Matching Algorithm

The purpose of the Entity Matching Algorithm is to build a record of all relevant entities (entities as defined in Section 5.6.1) detected in the last few frames. The record should represent the current traffic situation behind the SAS user. This record is constructed based on the filtered output of the YOLOv5 vision model. The pseudo-code for this algorithm can be found in Algorithm 2
For all the entities found in each new image, the algorithm tries to match each new entity to an already existing entity in its record. If it cannot find a match it assumes that the entity is new and adds it to the record. There are three steps in total to determine if a newly detected entity is a match to an entity in the record.

Algorithm 2 Entity Matching Algorithm

```
function EntitityMatchingAlgorithm($DetectedEntitities$)
    for <Every detected entity> do
        $Pairs \leftarrow$ FindEntityPairs($DetectedEntitities$)
    end for
    $BestMatches \leftarrow$ FindBestMatches($Pairs$)
    for <Every match in best matches> do
        if $EntitySizeIncrease \geq MinimumSizeIncrease$ then
            return $Entity$
        end if
    end for
    CleanUpEntitiesInRecord()
end function

function FindEntityPairs($DetectedEntity$)
    for <Every entity in the record> do
        if $DetectedEntity$ position matches with $EntityInRecord$ then
            $pairs \leftarrow (DetectedEntity, EntityInRecord)$
        end if
    end for
    return $pairs$
end function

function FindBestMatches($pairs$)
    for <Every unique entity in the pairs> do
        for <Every Pair that entity is part of> do
            if Pair is the Pair with the least distance between the entities then
                $BestMatches \leftarrow Pair$
                $Pair.EntityInRecord \leftarrow Pair.DetectedEntity$
            end if
        end for
    end for
    return $BestMatches$
end function

function CleanUpEntitiesInRecord()
    for <Every Entity in the record> do
        $Entity.frames\_to\_live - 1$
        if $Entity.frames\_to\_live = 0$ then
            Remove Entity from the Record
        end if
    end for
end function
```

First, the algorithm makes entity pairs. Each entity pair consists of an entity detected in the current frame and one detected in the record; This represents a possible match, but not

necessarily the best one. To determine which entities, if any, will form a pair, each detected entity will be compared to each entity in the record. If the position of the two entities does not differ more than a certain threshold and the entity class is identical it is considered a possible match. This means that the same entity can be part of different pairs. Pedestrians and cyclists can be paired together, this is supported because the trained YOLOv5n vision model sometimes confuses the two. The pseudo-code for this process can be found in the Function FindEntityPairs in Algorithm 2.
After all the pairs are made the algorithm will determine which of the possible entity pairs is the best match. To determine this the following procedure is used: For each already existing entity in the record, all pairs (which represent possible matches) which contain this specific entity are collected in a subset. From this subset, the pair of entities with the smallest difference in position is selected as the best match. The pseudo-code for this process can be found in the Function FindBestMatches in Algorithm 2.
When an entity in the record is matched to a newly detected entity, the properties in the record are updated with the properties of the newly detected matched entity. After each entity in the record has been matched to a newly detected entity any detected entities which have not been matched are added to the record. This indicates that these are new entities that were previously not known in the record. Because they are now included in the record they might be matched when processing the next image.
The algorithm will then check every match and determine if the entity got closer since the last time it was detected. It does this by comparing the size in the record with the detected size. If the detected size has increased by a certain threshold it is considered approaching and the Entity Matching algorithm will signify that a warning should be given to the SAS user. A schematic view of this process with an example can be seen in Figure 5.

5.6.4 Cleanup of entities

To prevent the record from being filled with entities every time an image has been processed a clean-up step will be run. The pseudo-code for the clean-up step can be found in Algorithm 2. In this function, entities which have not been seen for 5 subsequent frames are removed. The SAS assumes that these entities are not in view anymore and therefore not relevant. For example, a car might have taken a turn which took it out of view. Therefore, it can not pose a threat to the user. The grace period of five frames is chosen because an entity might still be present but not detected in every frame. This might happen because it is occluded by an obstacle or because the YOLOv5 vision model did not detect it (i.e., a false negative). With this approach, the entity remains in the record if it is rediscovered within a maximum of 5 frames.

5.6.5 Comparing the Entity Tracker algorithm to the state of the art

To compare the proposed Entity Tracker Algorithm to the current state of the art, we used the YOLOv8 entity tracking functionality [33]. This is a new feature introduced in the YOLO family which enables YOLOv8 to track any object it can detect between frames. To compare the Entity Tracker Algorithm and YOLOv8, three videos captured during Experiment 1 (found in Section 7) will be evaluated by the YOLOv8 entity tracking functionality. The results of YOLOv8 are compared to the logs made by the SAS Entity Tracking algorithm during Experiment 1. Only entities which pose a possible danger to the user (approaching from behind to overtake the user) are evaluated. Furthermore, to ensure that the comparison is fair, the filtering steps described in the Entity Validity checker (see Section 5.6.2) are also performed on the results of the YOLOv8 tracking algorithm. Note that none of the volunteers during Experiment 1 encountered a dangerous situation with a pedestrian so no comparison was possible for tracking pedestrians.

Tracking model	cars	cyclists
SAS Entity Tracker algorithm	64	**72**
YOLOv8 Tracking functionality [33]	**153**	46

Table 9: The total number of frames dangerous traffic users were tracked

The results can be found in Table 9, which shows how many frames of the same traffic user each of the methods tracked. YOLOv8 can track cars for far more frames than the Entity Tracking Algorithm. However, the Entity Tracker algorithm performs somewhat better on cyclists.

Qualitative analysis
A qualitative comparison of YOLOv8 and the SAS Entity Tracking algorithm is also performed on the videos of Experiment 1. Analysis shows that YOLOv8 generally keeps track of overtaking traffic users. While the Entity Tracking algorithm loses track of the same traffic user two or three times. An example of a takeover manoeuvre with the results of both methods can be found in Figure 6. Shown in the figure is that YOLOv8 consistently tracks the car with the same ID, while the logs for the Entity Tracking algorithm assign the same car three different IDs during the entire manoeuvre. In conclusion, based on this qualitative analysis YOLOv8 has less difficulty tracking cars and cyclists during takeover manoeuvres compared to the Entity Tracking Algorithm.

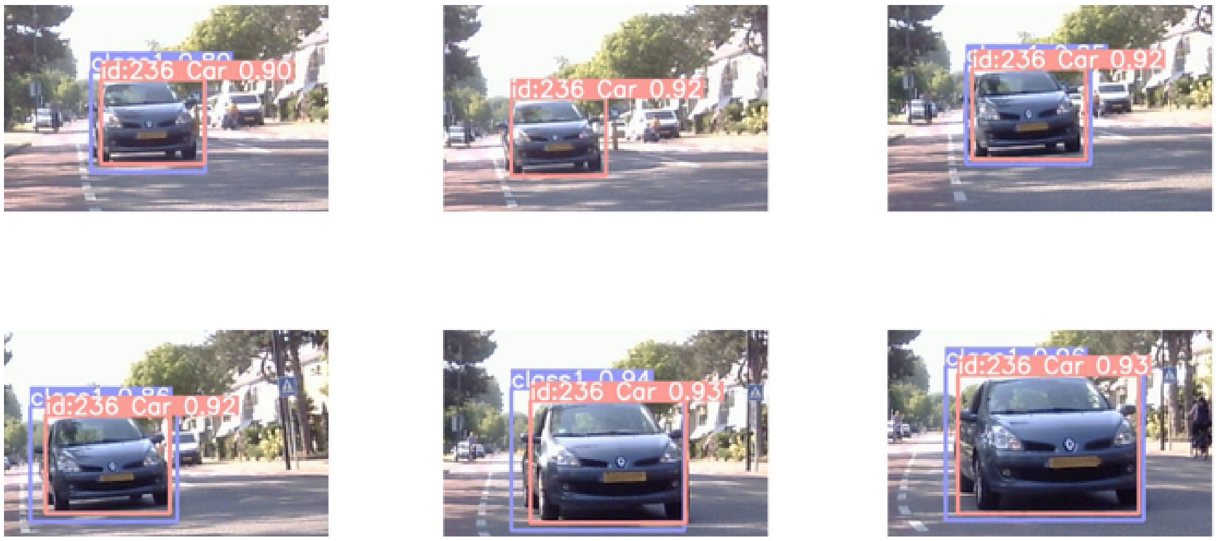

Figure 6: Frames of a video showing the tracking of YOLOV8 (bounding boxes displayed in orange, with a tracking ID) and the SAS Entity Tracking algorithm (bounding boxes displayed in blue without tracking ID).

Conclusion
Based on qualitative and quantitative analysis YOLOv8 is a more consistent and reliable tracker than the Entity Tracker algorithm. A small nuance is that in the quantitative analysis, the SAS Entity Tracker algorithm performed somewhat better in tracking cyclists. In the qualitative analysis, YOLOv8 loses track of a traffic user less often compared to SAS Entity Tracker algorithm. Furthermore, despite the Entity Tracker algorithm tracking more consecutive frames for cyclists it loses track and assigns a new ID to the same cyclist much more often than YOLOv8.

6 Safety Assist System (SAS) Design and implementation

The SAS consist of 4 components: the RGB camera, the SAS vision model software, the actuators and the computation hardware.
The RGB camera observes the environment behind the user and continuously captures images for the SAS vision model which it uses to determine if the current traffic situation is dangerous. The actuators provide a warning to the user if there is danger detected. The hardware provides a platform to connect these components and run the SAS vision model. Figure 7 shows a schematic display of the physical components of the SAS.

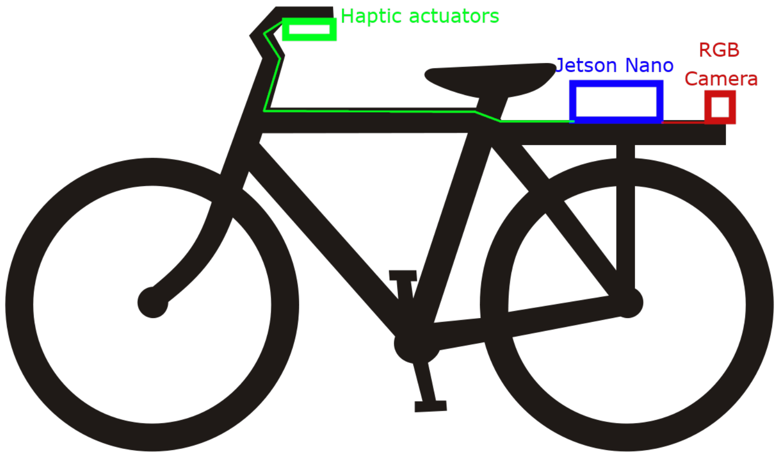

Figure 7: Schematic model of the SAS

Computation Hardware All the necessary computation is done in real-time on a Jetson Nano. For the specification see appendix D. The Jetson Nano is a small computer that is designed to run neural network applications for tasks like image classification or object detection. It is also optimised to run in a 5 or 10 watt configuration. In this project the 10 watt configuration is used to maximise performance at the cost of the lifetime of the battery. When used in the SAS the Jetson Nano is powered by a 7.2 V 2000 mAh battery. The voltage of this battery is converted to 5 V, to satisfy the required supply voltage of the Jetson Nano.
Sensor A video camera will be used as a sensor. This video camera is mounted on the back of the bicycle in such a way that it can observe behind the rider. For the specifications of the camera see Table 10.
Actuators To alert the user to dangerous situations two haptic actuators are used. These haptic actuators are attached to the handlebars of the bicycle. Since each handlebar has an actuator,

Model name	Logitech C270 HD Webcam
Resolution	720p
fps	30
Diagonal field of view	55°

Table 10: Specifications of the camera [57]

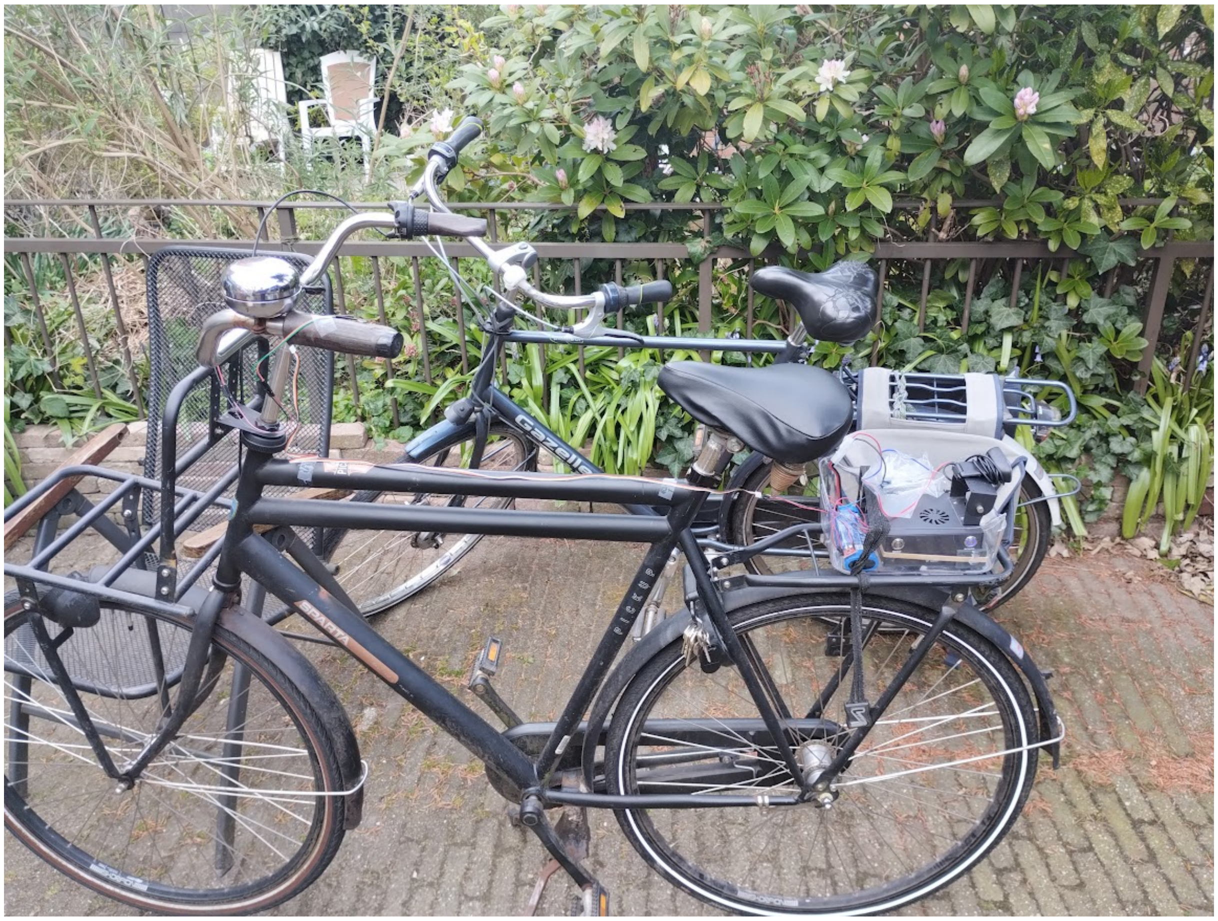

Figure 8: Prototype of the SAS

the SAS can give warnings for dangerous situations both to the right and left of the user. The type of haptic actuator used is the VM 0610 A 3.0 vibration motor. In the SAS these vibration motors will run at their maximum voltage, which results in approximately 10,000 RPM when used to warn for danger [58]. To ensure the safety of the SAS user, the cables of the actuators are attached to the frame and any exposed metal is insulated.
A picture of the final prototype can be found in Figure 8.

6.1 Comparison with other systems

A comparison with the SAS and similar systems found in literature is shown in Table 11. The main difference between the SAS and the other systems is that the SAS can detect and respond to cars, bicycles and pedestrians while the other systems only focus on detecting and warning cars. One other large difference is that none of these systems except the SAS have been tested in real-life situations. They have only been tested in a lab or in simulated conditions, which makes it difficult to compare the performance directly. None of these systems uses object detection on images to track road users except the SAS.

Name	Sensor type	Area covered	Type of vehicles tracked	Notification system
SAS	Video	Behind the user	Cars, cyclists and pedestrians	Haptic
The Cyber-Physical Bike: A Step Towards Safer Green Transportation [8]	Video and audio	Behind the user	Cars	None
"A Novel Collision Avoidance System for a Bicycle" [7]	Laser and sonar	Behind and to the left of the user	Cars	Audio and visual
"Close vehicle warning for bicyclists based on FMCW radar" [9]	Sonar	Behind the user	Cars	Cellphone of the user
Development of a Low-Cost LIDAR System for Bicycles [10]	Lidar and Camera	Behind the user	Cars	None
CycleGuard: A Smartphone-based Assistive Tool for Cyclist Safety Using Acoustic Ranging [11]	Acoustic	Left of the user	Cars	Audio

Table 11: Comparison between the SAS and similar systems

7 Experiment 1: User evaluation of the SAS

In this experiment, a prototype implementation of the Safety Assist System (SAS), is evaluated in a real-world environment. This first experiment consisted of participants cycling a predetermined path through Leiden (a medium-sized city in the Netherlands with a variety of different traffic situations such as busy roads and cycling lanes). Afterwards, they were asked to fill in an evaluation form about their experiences with the SAS. The goal of this user evaluation experiment was to answer two questions: Does the Active Safety Assist System make the participants feel safer during their cycling and is the system intuitive to use?

7.1 The route

The route the participants cycled is displayed in Figure 9. This route was selected to have a variety of road types and traffic activity. Before starting the experiment the participant was provided with a printed version of the route. They were then given time to study the route and were given the opportunity to ask questions if they were uncertain about any aspect of the experiment.

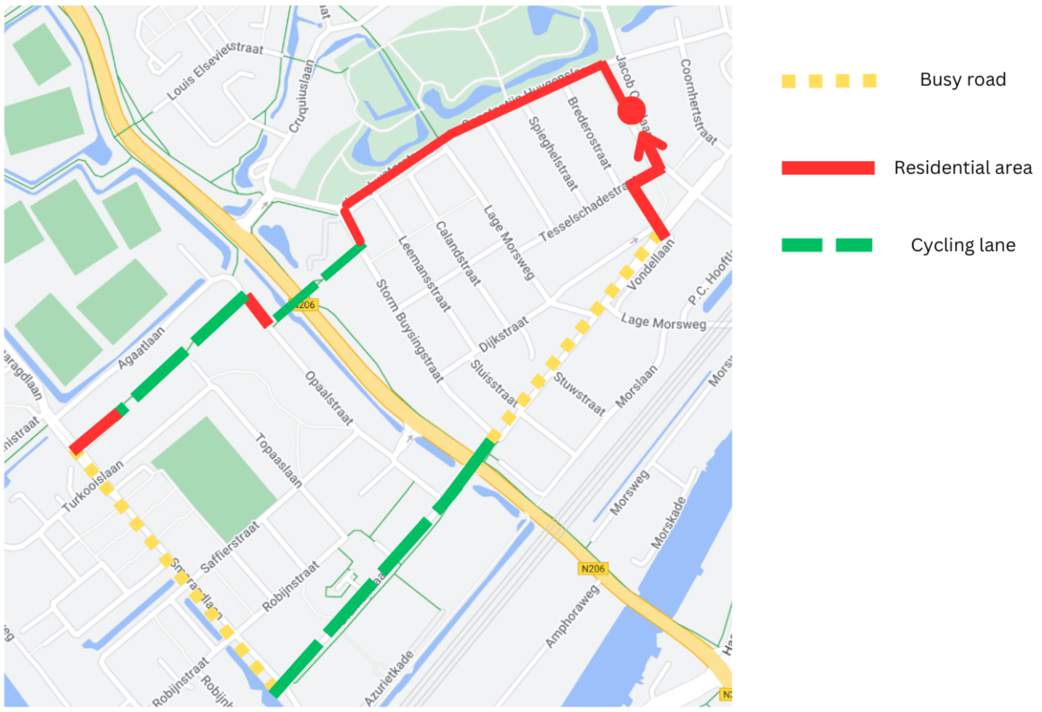

Figure 9: The route the participants took, divided into three categories

The route consisted of various traffic situations divided into three categories:

- Busy road: this category consists of roads with cycling lanes next to car lanes in which the cars have a speed limit of up to 50 kilometres.
- Residential area: this category consists of neighbourhoods with a maximum speed of 15 kilometres and no cycling lanes, There is also a high concentration of parked cars in these neighbourhoods.
- Cycling lanes: this category consists of only cycling lanes and no adjacent car lane.

7.1.1 The participants

To enlist the help of participants several people from the local area were asked to participate. Since the experiment involved a unique piece of hardware and necessary personal instruction, we did not had the capacity to handle a large amount of participants. As the experiment took place in real-life traffic there was a small risk of traffic accidents involved, which are always present when cycling. To minimize this risk, each participant needed to fulfil the following requirements.

- The participant is between 18 and 65 years old.
- The participant has experience with cycling and is comfortable riding a bike in traffic.

- The participant has some familiarity with Leiden so they are not too mentally distracted by following the predetermined route.
- The participant is aware of the relevant traffic laws in the Netherlands.
- The participant is able to bike safely on the bicycle made available for the experiment.
- The participant does not have any health issues which could interfere with safely using a bicycle.

A total of 7 participants used the SAS during the experiment, and each of them fulfilled the requirements listed above.

7.2 The SAS User evaluation form

In Experiment 1 the participants had to fill in two forms: The pre-experiment form and the post-experiment form. The pre-experiment form asked the participants asked about how safe they normally feel when cycling, this was done so there is a baseline to compare their answers after cycling with the SAS.
Immediately after Experiment 1, the participants were asked to fill in the post-experiment form. The post-experiment form consisted of ten questions, which asked about their experiences using the SAS while cycling. Both the pre- and post-experiment forms contained several identical questions to test the reliability. Since we expected most of our participants to be Dutch-speaking, the questions were provided in Dutch. An English form was available if the participants preferred it. The complete list of questions can be found in Appendix A. The closed questions were answered by giving a rating on a 7-point Likert scale (for more information see Section 3.2). The goal was to rate several aspects of the Safety Assist System such as reliability and the feeling of safety. The survey finished with a set of open questions to collect general feedback.

7.3 Experiment protocol

For every run of the experiment the following protocol was followed to ensure the safety of the participant and the correctness of the experiment.

1. The instructor checks with the participant if each of the safety requirements for a participant as described in Section 7.1.1 are fulfilled.
2. The instructor explains to the participant how the SAS works.
3. The participant gets the opportunity to inspect the bicycle used in the experiment and to test if they can ride the bicycle safely.
4. The instructor demonstrates the function of the haptic feedback actuators by letting the participant experience them at their maximum intensity.
5. The instructor explains the route described in Section 7.1 and gives the participant a printed copy to study.
6. The instructor allows the participant to ask questions about the experiment.

7. The participant fills in the pre-experiment form in which they will answer questions about their usual feeling of safety when cycling (See Section 7.2) and confirm that the steps described in this protocol were followed (For the full text of this statement see appendix C).

8. The instructor starts the SAS and the participant starts following the route.

9. After completing the route the participant fills in the post-experiment form. (See Section 7.2)

The values of the parameters of the SAS vision model used in the experiment can be found in Appendix H.

7.4 Results

In this Section the answers to the most relevant questions from the pre- and post-experiment forms are analysed and discussed. An overview of all the questions and the answers of the participants are provided in Appendix F.

7.4.1 Characteristics of the participants

Some of the questions in the form focused on the characteristics of the participants, this was done to put their answers about the SAS into context.

Age range	Participants in age range
18-30	4
30-40	1
40-50	0
50-65	2

Table 12: The age ranges of the participants

Table 12 shows that the age of the participants is spread over three age ranges 18-30, 30-40 and 50-65, with most of the participants being between 18-30 years old. It is beneficial for the experiment to have a spread of age ranges. Ideally, we would have liked to have had more older participants, because they might be an interesting target audience for systems like the SAS which have the goal of increasing road safety.
Figure 10 depicts the answers of the participants when asked if they feel safe while cycling in the city. All the participants agreed or strongly agreed with this statement. This indicates that they have confidence in their safety while cycling in their daily life. The questions about different types of environments to cycle in were answered similarly. All the participants also answered that they were experienced cyclists, as can be seen in Figure 11. The majority of the participants also indicated that they were not interested in a tool which would warn them of dangerous situations. These results imply that the participants might not be the target audience for the SAS.

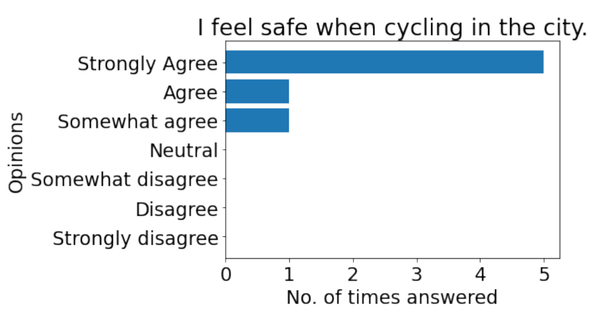

Figure 10: Responses on how safe the participants feel while cycling in the city

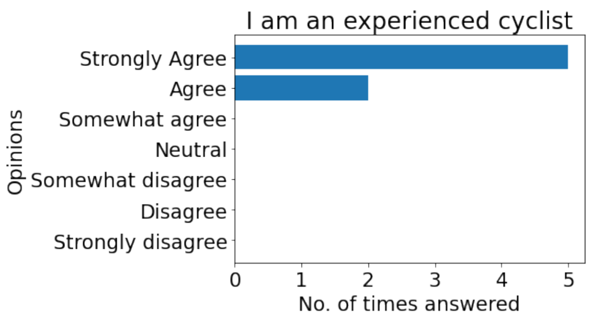

Figure 11: Responses on asking the participants if they are experienced cyclists

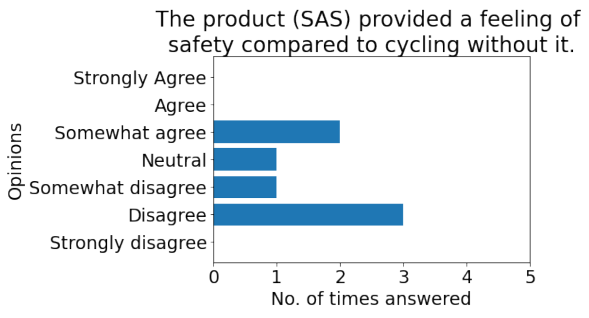

Figure 12: Responses on the feeling of safety experienced while using the SAS

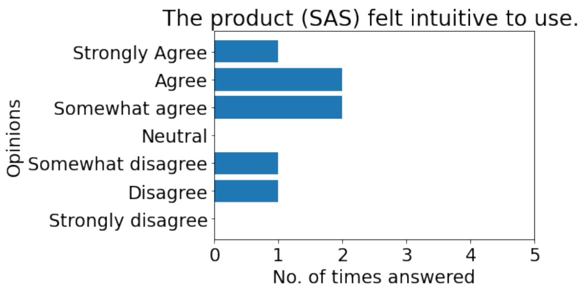

Figure 13: Responses on how intuitive the SAS felt

7.5 User evaluation of the functioning of the SAS

The main question of Experiment 1 is if the SAS provides an increased feeling of safety in comparison to cycling without. Looking at the results in Figure 12, most of the participants disagreed with this statement.
To further assess the performance of the SAS in real-life scenarios two more questions were asked about the warnings given during the experiment: One if they were relevant and two if they were correct. As seen in Figure 14 most of the participants either disagreed or somewhat disagreed with the statement that the warnings were relevant to the current traffic situation. Furthermore, participants found that the SAS gave false warnings during cycling as seen in

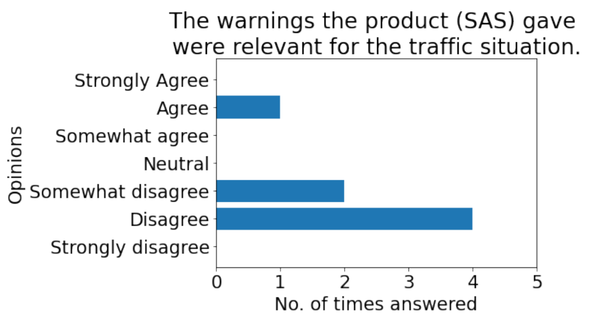

Figure 14: Responses on if the SAS gave relevant warnings

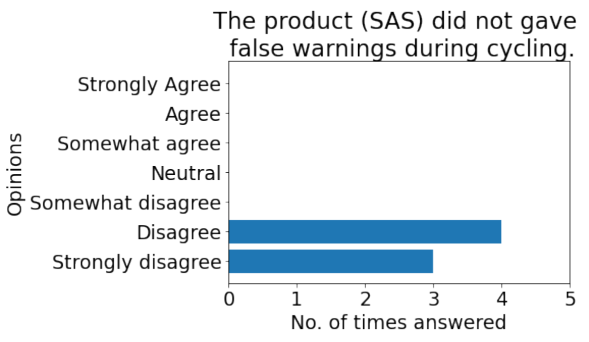

Figure 15: Responses on if the SAS did not gave false warnings during cycling

Figure 15. Also, none of the users stated that they would personally use the SAS for themselves if given the option as seen in Appendix A. The high number of false warnings combined with the already high confidence in their cycling capabilities explains why the participants mostly disagreed that the SAS increased their feeling of safety during their cycling experience.
To assess if the haptic feedback is intuitive to the participants, they were asked how intuitive the SAS felt when cycling. As shown in figure 13 the majority of the participants did feel that the SAS felt intuitive to use. We can therefore assume that haptic feedback is an intuitive way to warn of danger. This is in agreement with the findings of [24].

8 Experiment 2: Real-life Performance Evaluation of the SAS

The goal of experiment 2 is to calculate the precision, recall and F1 score achieved by the SAS during experiment 1. These values will be calculated by manually watching the footage taken during the experiment, and then comparing it with the logs. This data will give an objective estimation of the performance of the SAS to compare to the subjective experiences of the volunteers.

8.1 Results

The amount of True and False Positives and the amount of True and False Negatives can be found in Appendix G. The recall, precision and F1 score can be found in Table 13. Each time the SAS correctly warned the user of a dangerous situation it is counted as a True Positive, if the SAS failed to do so it is counted as a False Negative. When the SAS gave a warning but there was no dangerous situation it is counted as a False Positive. And when the SAS did detect road users but correctly decided that they posed no danger it is counted as a True Negative. In this context a dangerous situation is defined as a road user who is approaching the SAS user from behind intending to overtake the SAS user.

Road type	Recall	Precision	F1 score
All road types	0.74	0.19	0.30
Busy Road	**0.80**	0.40	0.53
Residential Area	0.71	0.07	0.12
Cycling Lane	0.7	**0.5**	**0.58**

Table 13: Performance of the SAS during the experiment

The performance of the SAS is displayed in Figure 13. Overall the recall is quite high but the precision is very low which contributes to an F1 score of 0.3. When splitting the results per road type as described in Section 7.1 it is clear that the residential area poses the highest difficulty to the SAS. The extremely low precision drags down the F1 score to around 0.13. This is mainly caused by the high number of parked cars in residential areas which generate a large number of False Positives. When encountering two parked cards next to each other the SAS often falsely classifies the cars as a single approaching car. The low score indicates that the steps taken in the Entity Validity Checker as described in Section 5.6.2 are insufficient to solve this problem. In comparison to the residential area, the cycling lanes have a higher F1 score. This is probably caused because the cycling lane is a relatively quiet part of the route with a low number of parked cars and other traffic users which decreases the opportunity for False Positives.

9 Conclusion

The goal of this paper is to design and test a Safety Assist System for bicycles, called the SAS. This system aims to be as efficient as possible to decrease cost and complexity. After performing a literature review it was decided that the SAS would consist of a RGB camera setup connected to a custom-made vision model which gives warnings through haptic feedback in the handlebars of the bicycle.
To develop the custom vision model YOLOv5n (a pre-trained version of YOLOv5) was used as a starting point due to its rapid interference speed and support for transfer learning. YOLOv5n was then fine-tuned on the KITTI dataset to significantly improve its performance in recognising cars, pedestrians and cyclists in traffic situations compared to the base model. The other part of the custom vision model is the SAS Entity Tracking algorithm which is developed to keep track of traffic users between frames and to assess if they pose a threat to the user.
Compared to the state of the art the SAS is unique in that it provides a camera-based safety system that is tested in real-life scenarios. Its object recognition is comparable to the state of the art. However, its Entity Tracking algorithm performs below the currently available solutions.
To test the SAS a prototype was constructed consisting of a Jetson Nano, a webcam and two haptic actuators. After several test rides and iterations to the SAS vision model, seven volunteers were asked to cycle a predetermined route with the SAS and were asked questions about their experiences.
The majority of the users did not experience an increased feeling of safety while using the SAS, mostly because due to its high rate of False Positives. This conclusion was reinforced by the manual evaluation of the logs of the SAS, which showed a very low precision score, especially in residential areas. However, most participants did find the haptic feedback intuitive. And the recall score of the warnings was significantly better than the precision.
This leads us to conclude that although the SAS has a high recall score and is intuitive to use it is not yet ready for use in real-life scenarios. This will be the case until the SAS vision model is improved in such a way that the large amount of False Positives is significantly decreased.

10 Future work

The SAS can be improved and or expanded upon in several different ways, this section suggests several new approaches.
Hardware When designing the SAS the goal was to make it as efficient as possible. However, this approach limited the hardware which could be used. For example, a standard webcam was used to provide images to the vision model. This webcam is not designed to capture rapidly moving objects or deal with the lighting in an outside environment. Testing the SAS with, for example, a global shutter camera might give better performance at the trade-off that these cameras are more expensive. Furthermore, the hardware used in the SAS, the Jetson Nano, is limited in computational power compared to more expensive or more current hardware. If the SAS has more computational power available it can run more accurate vision models while not sacrificing the interference speed.

Vision model Several improvements of the vision model are possible, but we expect that training the vision model on a more specific data set might give the largest performance improvement. The current iteration is trained on the KITTI dataset[44]. Although the use of this dataset has significantly improved the detection of traffic users it is not a perfect fit for training of the SAS. This is mostly because it contains traffic users in different orientations. However, the SAS is only interested in detecting approaching traffic users. Therefore, it needs to only recognise the front of traffic users, recognising the sides and the back of traffic users is not necessary and can even lead to False Positives. A dataset which contains only front-facing traffic users would be more useful. Also, the SAS currently struggles to recognise less common traffic users such as children on small bikes or unusually small cars. Adding examples of these traffic users to the dataset would be an improvement. To take this even further it would be interesting to develop a dataset which only contains dangerous situations. This way instead of using object detection to determine a dangerous situation the vision model can be trained to directly recognise dangerous situations. However, this would require a completely different design of the vision model and training process. When using this different vision model other sensors e.g. LIDAR might also be included in this system. This approach is reminiscent of the current approaches in automated driving [15]. Another improvement may be to upgrade the Entity Tracking capabilities of the SAS. As seen in Section 5.6.5, the SAS performs worse at Entity Tracking than some other available methods. Implementing these methods in the SAS vision model might reduce the number of false warnings. However, more recent hardware may be required to run these methods.
Other directions At the moment the SAS can only detect dangers from behind the user. It might be beneficial to also give warnings from the side or the front. An example would be the research done to detect danger from the side with sonar [7].
Experiment The experiment was executed with seven participants. This is a small number and to improve the validity of the experiment more participants should be used. Also, the route was relatively short due to power requirements and to simplify following it for the participants. Testing the SAS with a larger route would give more accurate results. However, we would first recommend improving the SAS further since the results of the experiments show that it is not effective in its current state.

References

[1] M. Stoffers, "Cycling as heritage: representing the history of cycling in the netherlands," *The journal of transport history*, vol. 33, no. 1, pp. 92–114, 2012.

[2] J. Vanparijs, L. Int Panis, R. Meeusen, and B. De Geus, "Exposure measurement in bicycle safety analysis: A review of the literature," *Accident; analysis and prevention*, vol. 84, pp. 9–19, 08 2015.

[3] J. Duan, R. Li, L. Hou, W. Wang, G. Li, S. Li, B. Cheng, and H. Gao, "Driver braking behavior analysis to improve autonomous emergency braking systems in typical chinese vehicle-bicycle conflicts," *Accident; analysis and prevention*, vol. 108, pp. 74–82, 08 2017.

[4] Z. Yang, Z. Yang, J. Smith, and B. A. P. Robert, "Risk analysis of bicycle accidents: A bayesian approach," *Reliability Engineering & System Safety*, vol. 209, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832021000284

[5] L. de Guerre, S. Sadiqi, L. Leenen, C. Oner, and S. Gaalen, "Injuries related to bicycle accidents: an epidemiological study in the netherlands," *European Journal of Trauma and Emergency Surgery*, vol. 46, 04 2020.

[6] G. Savino, R. Lot, M. Massaro, M. Rizzi, I. Symeonidis, S. Will, and J. Brown, "Active safety systems for powered two-wheelers: A systematic review," *Traffic Injury Prevention*, vol. 21, no. 1, pp. 78–86, 2020, pMID: 31914321. [Online]. Available: https://doi.org/10.1080/15389588.2019.1700408

[7] W. Jeon and R. Rajamani, "A novel collision avoidance system for bicycles," in *2016 American Control Conference (ACC)*, 2016, pp. 3474–3479.

[8] S. Smaldone, C. Tonde, V. K. Ananthanarayanan, A. Elgammal, and L. Iftode, "The cyber-physical bike: A step towards safer green transportation," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 56–61. [Online]. Available: https://doi.org/10.1145/2184489.2184502

[9] T. Krejci and M. Mandlik, "Close vehicle warning for bicyclists based on fmcw radar," in *2017 27th International Conference Radioelektronika (RADIOELEKTRONIKA)*, 2017, pp. 1–5.

[10] I. Blankenau, D. Zolotor, M. Choate, A. Jorns, Q. Homann, and C. Depcik, "Development of a low-cost lidar system for bicycles," *SAE Technical Paper*, 04 2018.

[11] W. Jin, S. Murali, Y. Cho, H. Zhu, T. Li, R. Panik, A. Rimu, S. Deb, K. Watkins, X. Yuan, and M. Li, "Cycleguard: A smartphone-based assistive tool for cyclist safety using acoustic ranging," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, pp. 1–30, 12 2021.

[12] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[13] F. Zhang, H. Stähle, G. Chen, C. C. C. Simon, C. Buckl, and A. Knoll, "A sensor fusion approach for localization with cumulative error elimination," in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, pp. 1–6.

[14] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3337

[15] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2017.

[16] G. Gil, G. Savino, S. Piantini, and M. Pierini, "Is stereo vision a suitable remote sensing approach for motorcycle safety? an analysis of lidar, radar, and machine vision technologies subjected to the dynamics of a tilting vehicle," in *7th Transport Research Arena TRA 2018 (TRA 2018)*, 04 2018.

[17] G. Gil, G. Savino, S. Piantini, and M. Pierini "Motorcycles that see: Multifocal stereo vision sensor for advanced safety systems in tilting vehicles," *Sensors (Basel, Switzerland)*, vol. 18, 01 2018.

[18] M. Kirjanov, P. Grzyb, J. Hoffmann, and A. Ozman, "Advanced driver assistance systems for motorcycles: Concept of a lane change assist," *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV))*, 10 2017.

[19] C. Chen and Y. Chen, "Real-time approaching vehicle detection in blind-spot area," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, 2009, pp. 1–6.

[20] S. Muro, I. Yoshida, M. Hashimoto, and K. Takahashi, "Moving-object detection and tracking by scanning lidar mounted on motorcycle based on dynamic background subtraction," *Artificial Life and Robotics*, vol. 26, 08 2021.

[21] M. Liebner, F. Klanner, and C. Stiller, "Active safety for vulnerable road users based on smartphone position data," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 256–261.

[22] V. HUTH and C. GELAU, "Predicting the acceptance of advanced rider assistance systems," *Accident Analysis and Prevention*, vol. 50, pp. pp.578–586, Jan. 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00910238

[23] V. Beanland, M. Lenné, E. Fuessl, M. Oberlader, S. Joshi, T. Bellet, A. Banet, L. Rößger, L. Leden, I. Spyropoulou, G. Yannis, H. Roebroeck, J. Carvalhais, and G. Underwood, "Acceptability of rider assistive systems for powered two-wheelers," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 19, p. 63–76, 07 2013.

[24] K. Touliou, D. Margaritis, P. Spanidis, S. Nikolaou, and E. Bekiaris, "Evaluation of rider's support systems in power two wheelers (ptws)," *SIIV-5th International Congress - Sustainability of Road Infrastructures 2012*, 7 2013.

[25] N. Baldanzini, G. Bencini, and M. Pierini, “Design and preliminary testing of an haptic handle for powered two wheelers,” *European Transport Research Review*, vol. 3, pp. 1–9, 06 2011.

[26] J. Andres, T. Kari, J. von Kaenel, and F. F. Mueller, “”co-riding with my ebike to get green lights”,” in *Proceedings of the 2019 on Designing Interactive Systems Conference*, ser. DIS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1251–1263. [Online]. Available: https://doi.org/10.1145/3322276.3322307

[27] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[28] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A review of yolo algorithm developments,” *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022, the 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922001363

[29] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 04 2020.

[30] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “Yolox: Exceeding yolo series in 2021,” *arXiv preprint arXiv:2107.08430*, 2021.

[31] J. Terven and D. Cordova-Esparza, “A comprehensive review of yolo: From yolov1 and beyond,” 2023.

[32] M. Hussain, “Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection,” *Machines*, vol. 11, no. 7, p. 677, 2023.

[33] Ultralytics, “Yolov8,” Available at https://github.com/ultralytics/ultralytics (2023-09-24).

[34] F. M. Talaat and H. ZainEldin, “An improved fire detection approach based on yolo-v8 for smart cities,” *Neural Computing and Applications*, vol. 35, no. 28, pp. 20 939–20 954, 2023.

[35] Y. Li, Q. Fan, H. Huang, Z. Han, and Q. Gu, “A modified yolov8 detection network for uav aerial image recognition,” *Drones*, vol. 7, no. 5, p. 304, 2023.

[36] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, “Multiple object tracking: A literature review,” *Artificial Intelligence*, vol. 293, p. 103448, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370220301958

[37] D. M. Jiménez-Bravo, Álvaro Lozano Murciego, A. Sales Mendes, H. Sánchez San Blás, and J. Bajo, “Multi-object tracking in traffic environments: A systematic literature review,” *Neurocomputing*, vol. 494, pp. 43–55, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222004672

[38] K. Wang and M. Liu, “Yolov3-mt: A yolov3 using multi-target tracking for vehicle visual detection,” *Applied Intelligence*, vol. 52, 01 2022.

[39] Y. Zou, W. Zhang, W. Weng, and Z. Meng, "Multi-vehicle tracking via real-time detection probes and a markov decision process policy," *Sensors*, vol. 19, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1309

[40] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *4thBMTT MOT Challenge Workshop at the Computer Vision and Pattern Recognition Conference (CVPR) 2019*, 2020.

[41] G. Gil, G. Savino, and M. Pierini, "First stereo video dataset with ground truth for remote car pose estimation using satellite markers," in *Tenth International Conference on Machine Vision (ICMV 2017)*, 04 2018, p. 36.

[42] J.-L. Blanco-Claraco, F. Ángel Moreno-Dueñas, and J. González-Jiménez, "The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014. [Online]. Available: https://doi.org/10.1177/0278364913507326

[43] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017. [Online]. Available: https://doi.org/10.1177/0278364916679498

[44] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[45] R. Bonnin, *Machine Learning for Developers*. Birmingham, UK: Packt Publishing, 2017.

[46] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/3/279

[47] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert scale: Explored and explained," *British Journal of Applied Science & Technology*, vol. 7, pp. 396–403, 01 2015.

[48] K. Finstad, "Response interpolation and scale sensitivity: Evidence against 5-point scales," *J. Usability Studies*, vol. 5, no. 3, p. 104–110, may 2010.

[49] H. Taherdoost, "What is the best response scale for survey and questionnaire design; review of different lengths of rating scale / attitude scale / likert scale," *International Journal of Academic Research in Management*, vol. 8, no. 1, pp. 1–10, 06 2019.

[50] Ultralytics, "Yolov5 github page." [Online]. Available: https://github.com/ultralytics /yolov5 (2023-07-02)

[51] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[52] N. Al-Qubaydhi, A. Alenezi, T. Alanazi, A. Senyor, N. Alanezi, B. Alotaibi, M. Alotaibi, A. Razaque, A. A. Abdelhamid, and A. Alotaibi, "Detection of unauthorized unmanned aerial vehicles using yolov5 and transfer learning," *Electronics*, vol. 11, no. 17, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/17/2669

[53] Nivida, "Nvidia tensorrt." [Online]. Available: https://developer.nvidia.com/tensorrt (2023-06-26)

[54] Ultralytics, "Train custom data." [Online]. Available: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data (2023-09-22)

[55] M. Ju, H. Luo, Z. Wang, B. Hui, and Z. Chang, "The application of improved yolo v3 in multi-scale target detection," *Applied Sciences*, vol. 9, no. 18, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/18/3775

[56] S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, and K. Yu, "Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 345–25 360, 2022.

[57] Logitech, "C270 hd webcam." [Online]. Available: https://www.logitech.com/en-us/products/webcams/c270-hd-webcam.960-000694.html (023-06-26)

[58] EKULIT, "Vm-0610a3.0 vibration motor." [Online]. Available: https://www.ekulit.com/products/vibration-motors/vm-0610a3-0 (2023-06-26)

[59] Nivida, "Jetson nano developer kit." [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (2023-06-26)

A Form questions

A.1 Questions before the experiment

- I am an experienced cyclist. (Answered by a 7 point Likert scale)
- I feel safe when cycling in the city (Answered by a 7 point Likert scale)
- I feel safe when cycling out of town. (Answered by a 7 point Likert scale)
- I feel safe when cycling on roads without cycling lanes. (Answered by a 7 point Likert scale)
- I feel safe when cycling on a cycling lane. (Answered by a 7 point Likert scale)
- I would like a tool that during cycling gives warnings in potentially dangerous traffic situations. (Answered by a 7 point Likert scale)

A.2 Questions after the experiment

- I feel safe when cycling in the city. (Answered by a 7 point Likert scale)
- I feel safe when cycling on roads without cycling lanes. (Answered by a 7 point Likert scale)
- I feel safe when cycling on a cycling lane. (Answered by a 7 point Likert scale)
- I would like a tool that during cycling gives warnings in potentially dangerous traffic situations. (Answered by a 7 point Likert scale)
- The product (The SAS) provided a feeling of safety compared to cycling without it. (Answered by a 7 point Likert scale)
- The warnings the product (The SAS) gave were relevant for the traffic situation. (Answered by a 7 point Likert scale)
- The product (The SAS) did not gave false warnings during cycling. (Answered by a 7 point Likert scale)
- The product (The SAS) felt intuitive to use. (Answered by a 7 point Likert scale)
- I would use the product (The SAS) myself. (Answered by a 7 point Likert scale)
- Would you recommend the product (The SAS) for other road users? If yes then to whom? (Open question)

B In depth training results

Model	Precision	Recall	mAP50	mAp50-95
YOLOv5n trained on KITTI [30]	0.888	0.714	0.831	0.451
YOLOV5n TensorRT trained on KITTI[30]	0.851	0.737	0.823	0.446
YOLOv5s trained on KITTI [30]	0.893	**0.804**	0.879	**0.516**
YOLOv5sTensorRT trained on KITTI [30]	**0.927**	0.797	**0.884**	0.509

Table 14: Results of the pedestrian class with the trained YOLOv5 models

Model	Precision	Recall	mAP50	mAp50-95
YOLOv5n trained on KITTI [30]	0.942	0.902	0.963	0.743
YOLOv5n TensorRT trained on KITTI [30]	0.921	0.919	0.964	0.723
YOLOv5s trained on KITTI [30]	0.945	**0.945**	0.979	**0.82**
YOLOv5sTensorRT trained on KITTI [30]	**0.952**	0.942	**0.978**	0.792

Table 15: Results of the car class with the trained YOLOv5 models

Model	Precision	Recall	mAP50	mAp50-95
YOLOv5n trained on KITTI [30]	0.908	0.762	0.855	0.551
YOLOv5n TensorRT trained on KITTI[30]	0.877	0.737	0.84	0.533
YOLOv5s trained on KITTI [30]	0.935	**0.852**	**0.916**	**0.643**
YOLOv5sTensorRT trained on KITTI [30]	**0.941**	0.830	0.906	0.639

Table 16: Results of the cyclist class with the trained YOLOv5 models

C Disclaimer

I hereby declare that I have inspected the bicycle and can safely ride it during the experiment. I also have been demonstrated how the SAS will warn me of incoming danger. I understand that for this experiment I need to cycle in a city environment and am solely responsible for driving safely and adhering to all the relevant traffic laws. I confirm that I fulfill the safety requirements specified to me. Furthermore I know that I can stop the experiment at any time, and understand that the equipment can be discarded if it would have any negative impact on my safety. I also declare that I know the route and have had the opportunity to ask any questions about the experiment or the functionality of the SAS that I might have.

D Jetson Nano Developer kit specifications

Full name	Jetson Nano Developer Kit
GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s

Table 17: Specifications of the Jetson Nano [59]

E Training desktop specifications

GPU	NVIDIA GeForce RTX 3060 TI
CPU	AMD Ryzen 7 5800X3D 8-core @ 3.4GHz
Memory	32 GB 2 x 64-bit DDR4
OS	Windows 11

Table 18: Specifications of the training desktop

F Form questions results

Statement	Strongly disagree	Disagree	Some-what disagree	Neither agree nor disagree	Some-what agree	Agree	Strongly agree
I am an experienced cyclist	0	0	0	0	0	2	5
I feel safe when cycling in the city	0	0	0	0	1	1	5
I feel safe when cycling on roads without cycling lanes	0	0	0	0	1	1	5
I feel safe when cycling on a cycling lane	0	0	1	0	1	1	4
I would like a tool that during cycling gives warnings in potentially dangerous traffic situations.	1	2	1	1	1	0	1
The product (The SAS) provided a feeling of safety compared to cycling without it.	0	3	1	1	2	0	0
The warnings the product (The SAS) gave were relevant for the traffic situation.	0	4	2	0	0	1	0
The product (The SAS) did not gave false warnings during cycling.	3	4	0	0	0	0	0
The product (The SAS) felt intuitive to use.	0	1	1	0	2	2	1
I would use the product (The SAS) myself.	1	5	1	0	0	0	0

Table 19: Results from the form

G Experiment results

Area	False positives	False negatives	True positives	True negatives
Busy road	12	2	8	1200
Residential area	69	2	5	1852
Cycling lane	7	3	7	421
All areas	88	7	20	3473

Table 20: True/False Positives and True/False Negatives during the experiment

H Parameters of the SAS during Experiment 1 and 2

Parameter	Value
Maximum position difference to match two entities	20 pixels
Minimum size increase to signify approaching	10 pixels
Minimum entity size for car	100 pixels
Minimum entity size for pedestrian	70 pixels
Minimum entity size for cyclist	70 pixels
Left border	250 pixels
Right border	1050 pixels
Maximum size width ratio	1.75 pixels

Table 21: Values of parameters of the SAS vision model during Experiment 1 and 2

I Confusion matrix

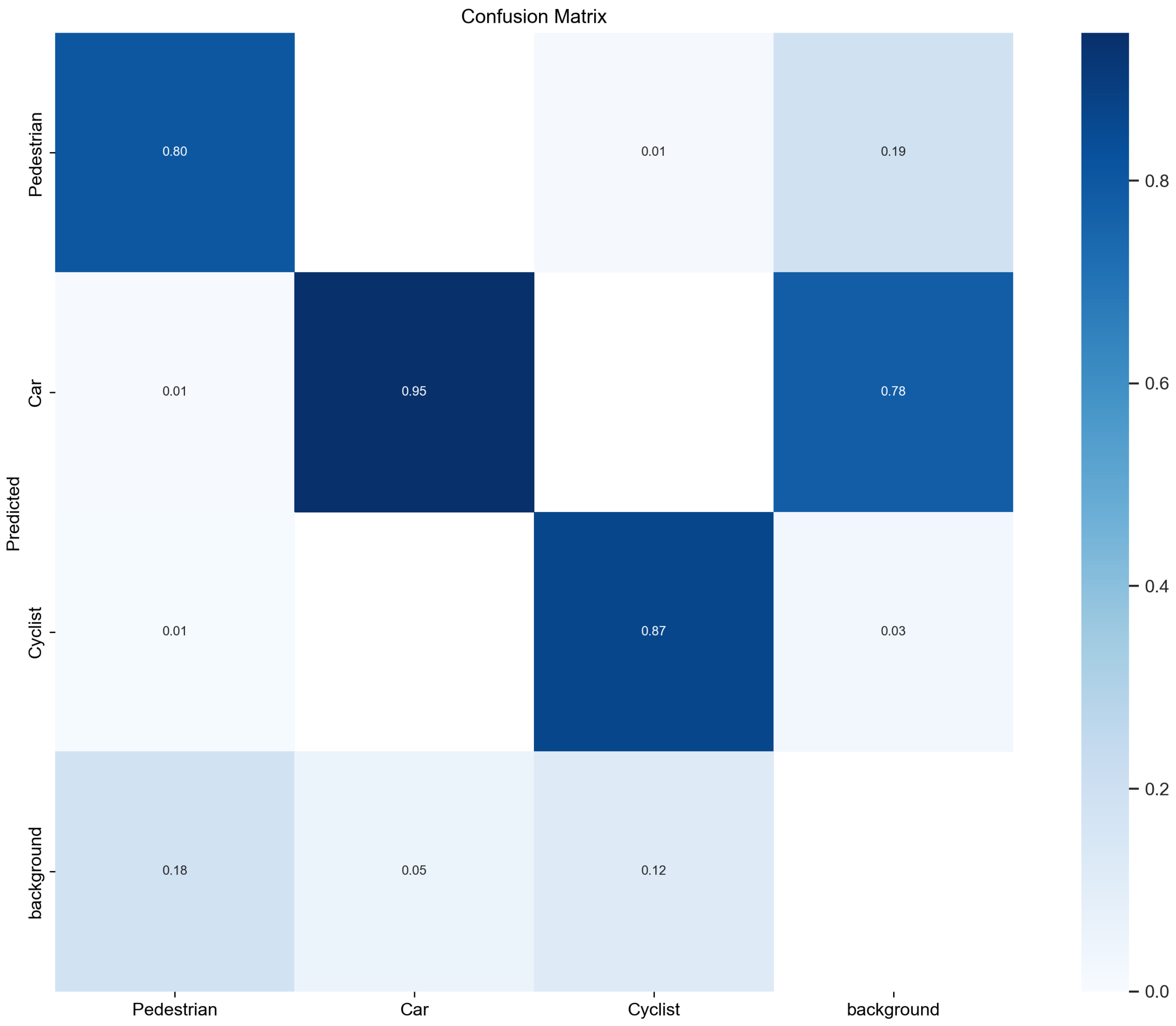

Figure 16: Confusion matrix after training