Investigating Pedestrian-Vehicle Interaction for Autonomous Vehicles

Tim Van den Driesschen, Dennis Heijnen, Julian Kunze, Emanuel Buchholz

January 13, 2019

Contact:	emanuel.buchholz@est.fib.upc.edu
	julian.kunze@est.fib.upc.edu
	dennis.heijnen@est.fib.upc.edu
	tim.van.den.driesschen@est.fib.upc.edu
Instructors:	Cecilio Angulo Bahon
	Anais Garrell Zulueta

Contents

1 Introduction

Nowadays, it is hard to imagine a world without cars. Over 1 billion cars are driving on the street and all these cars need to interact with each other and with other traffic users to create a safe environment. One such example is the interaction with pedestrians when they cross the street. Both car drivers and pedestrians use subtle gestures such as a hand wave, a smile or simple head nod. Gestures which are understood between cultures, gender and age. However, the development of autonomous vehicles is rapidly increasing and will soon make an entrance in the traffic environment [Shaheen and Stocker, 2018]. For the first time, people will have to share the road with computer-controlled cars that can direct their own movements. This is a big change from nowadays where machines are either controlled by humans or are constrained in their range of movement. Two important issues arise with the development of self-driving cars. One, the ability of humans to understand the intent of self-driving cars and two, self-driving cars should understand the intent of humans. Only when both these issues are addressed will self-driving cars safely drive on our roads.

1.1 The relevance of understanding self-driving cars

Autonomous driving will offer a lot of advantages such as an increased comfort for the driver and social impact on both micro and macro level [Winkle, 2016]. However, it will also create a void in social interaction. Autonomous cars can be operated without a driver paying attention to the road or even without being present, which will make it impossible to use the above mentioned gestures. Research has shown that pedestrians feel uncomfortable and worried when the passenger in the driver's seat is distracted during the pedestrian crossing the street [Yang, 2017]. The study also showed that having eye-contact with the passenger in the drivers seat makes the pedestrian have more positive emotions and feel safer to cross the road. A lack of proper interaction between an autonomous car and other traffic users can create unsafe traffic environments. Autonomous cars and pedestrians need to collaborate as teammates to create a safe environment for both entities. Therefore, it is important to research how autonomous cars should interact with pedestrians. Technologies should be designed to make people feel comfortable on the road. An important factor in the design of these technologies comes from the psychology behind how humans view new technologies. It is essential to make people trust new devices or they will not be adopted in the long run [Masahiro Mori and Kageki, 2012].

Recent studies investigated the effect of different gestures of different modalities. The study of Mahadevan and Sharlin [2018] tested different modalities of interaction such as audio, visual and physical cues. The results of the study showed that when testing singular modalities all the modalities were perceived as having a positive impact on the interaction between the car and pedestrian except when the modality is too human-like. For instance, an animated face on the car or a plastic robot hand on top of the car were perceived as negatively impacting the interaction. Studies about the uncanny valley have shown that the more human-like machines become, the more unsettling the machines are perceived [Masahiro Mori and Kageki, 2012]. The study by Mahadevan and Sharlin [2018], also showed that a mixture of modalities is better than using only one modality. However, the study used only one gesture for every modality. Therefore, the current study will focus on testing different gestures for one modality to see which is the most effective. The most common technique to transmit a message is the use of visual cues [Rasouli and Tsotsos, 2018]. After identifiying the relevance of cars understanding humans, the state-of-the-art of the visual gestures will be addressed.

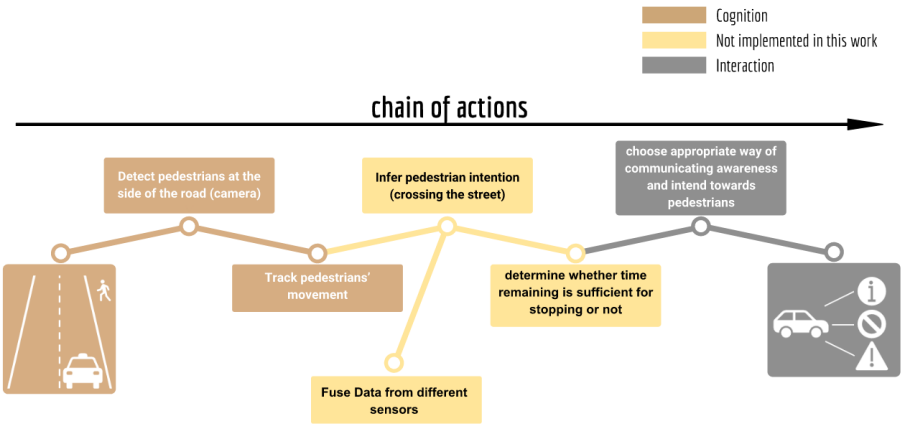


Figure 1: Solution proposal addressed in this work

1.2 The relevance of cars understanding humans

Communication works seldom only in one way, which is why autonomous cars can rather be seen as a teammate in traffic: They need to collaborate with pedestrians. However, before collaboration is possible, it is necessary for self-driving cars to understand pedestrian behaviour. In a survey of pedestrian behaviour and accident risk, it was reported that the majority of pedestrian accidents happen away from road-crossing facilities [Ward and Evans, 1994]. Therefore, it is important for self-driving cars being able to predict beforehand whether a pedestrian is going to cross the street or not at every point of their drive and not only when approaching a designated cross-walk. Predicting this includes not only detecting a human-being but also estimating its intention. Self-driving cars need this capability for two main reasons, one for smooth traffic and two for pedestrian safety. While preventing accidents caused by self-driving cars overlooking a pedestrian that is crossing the street is undoubtedly of utmost importance autonomous cars will also be rejected if they stop for every person standing on the roadside. Therefore, a robust pedestrian detection and intention detection system has to be developed that guarantees an accurate prediction of the pedestrians' actions regarding road crossing. Only then, autonomous cars will be able to drive both smoothly but also safely on the road and thus be publicly accepted.

Of course, we are not able to solve all problems autonomous vehicles face when interacting with pedestrians, in this work. Therefore, we have focused on the two ends of the solution chain you can see in Figure 1, namely the cognition and interaction part of a final implementation.

1.3 State-of-the-art

1.3.1 State-of-the-art of visual gestures

The study of Matthews and Kieson [2017] tested an intent communication system that acts in place of a human driver. The study used a mounted display on a car to communicate with the pedestrian. The display suggested a course of action. For instance, when the car detected the pedestrian and was able to stop in time, the display showed the text "Cross now". The study tested the pedestrian s level of trust towards the system and if they believed that it was safe to cross the road. The results showed that using this particular display could help to resolve 38%

of potentially dangerous situations. The study of Clamann and Cummings [2017] used a display as well but created a different representation of the gesture on the display. The display showed symbols, as shown in Figure 2, indicating if the pedestrian could cross the street or not.

Figure 2: Symbols used to interact with pedestrians in the study of Clamann and Cummings [2017].

The results of the study showed that participants did not trust this method more compared to a car having no display. Another model was tested, where the display showed the speed of the car [Clamann and Cummings, 2017]. Again, participants did not trust this method more compared to a car having no display. Therefore, using text to indicate a gesture seems the most suitable method when using a display.

Another method to communicate with pedestrians is using projection. Figure 3 shows an example of a projection as implemented in the Mercedes-Benz F015 Luxury [Daimler, 2017]. The autonomous car will project a green zebra crossing on the road indicating that the pedestrian can safely walk.

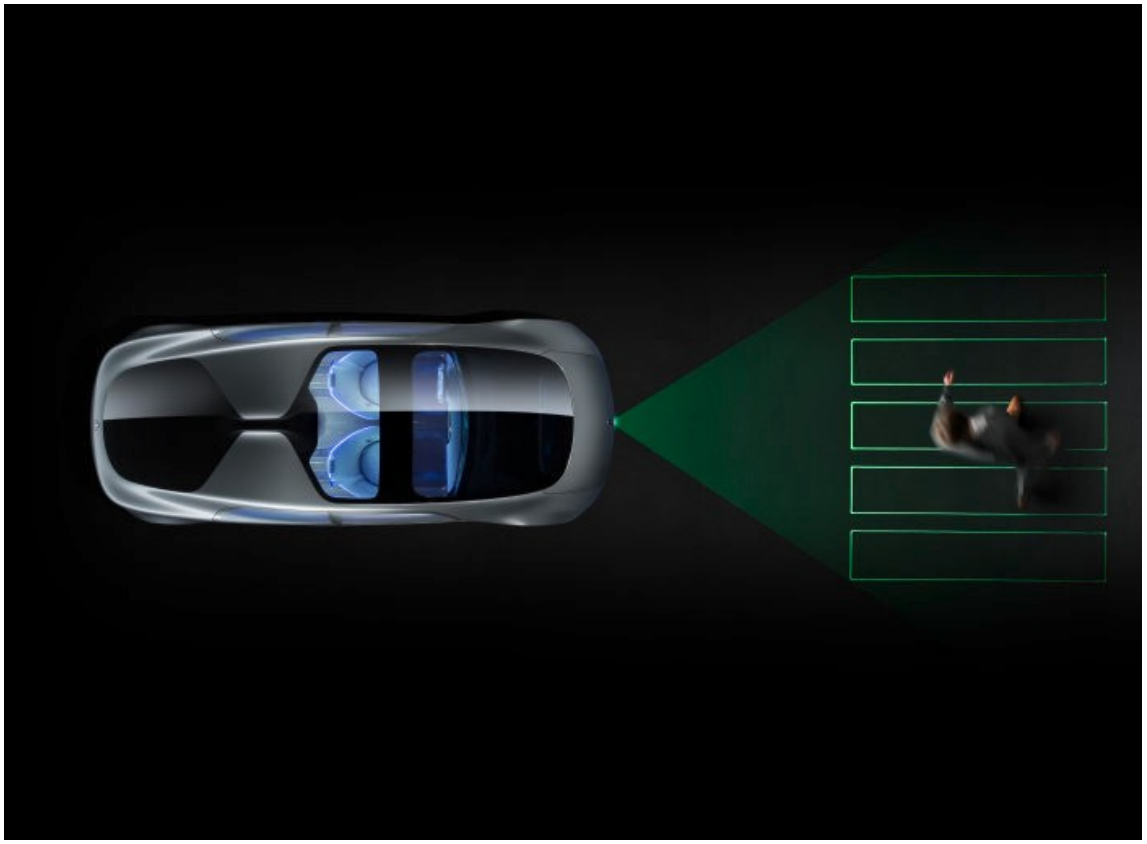


Figure 3: Example of communication via projection. [Daimler, 2017]

No scientific literature research is done so far to investigate how pedestrians perceive this way of communicating. Nevertheless, it is a promising option to include for research in the current study.

The study of Lagstrom [2015] used LED strips as a communication system for interaction with pedestrians. Different positions of the LED strips were tested against each other. LED strips were positioned on either the grill or windshield. The results showed that the intended gesture was more clear when the LED strips were positioned on the grill of the car. The strips can either be blinking in a green color when the pedestrian can cross the street or an array of arrows representing the direction the pedestrian can walk [Lagstrom, 2015].

1.3.2 State-of-the-art pedestrian intention recognition

As described in subsection 1.2 a correct understanding of the pedestrians' intention when approaching a street is essential for increasing safety and traffic fluidity and therefore, to achieve public acceptance. Nevertheless, most intention recognition approaches rely on previous steps e.g. object detection and tracking methods and techniques where different features such as position, body and head posture or movement direction and speed are extracted. These features are then used for the actual intention recognition part. In the following, all steps that are conducted in state-of-the-art approaches and their respective algorithms are presented.

1.3.2.1 Object detection In contrast to image classification and localization of objects in images, object detection describes the task of identifying multiple objects and the respective region they cover in an image. Usually the detected objects are returned by such algorithm as their label together with coordinates of the enclosing axis-aligned bounding box (e.g. coordinates of top-left and bottom-right corner) [Zhao et al., 2018]. Figure 4 shows graphically the difference between object classification and localization, object detection and instance segmentation in images. While the former is only able to analyze the image as a whole are the latter ones capable of distinguishing multiple objects per image. In the following, two approaches for pedestrian detection are presented.

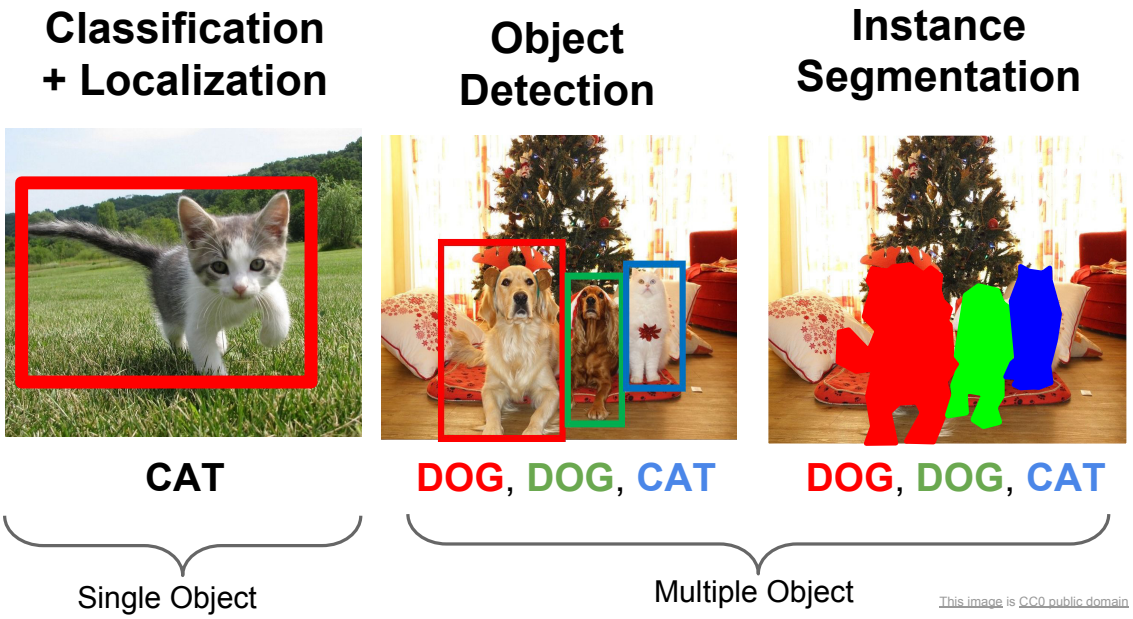


Figure 4: Difference between image classification, object detection and instance segmentation in images [Li et al., 2018]

.

HOG HOG (Histograms of Oriented Gradient) detection essentially is a feature descriptor that uses image gradients and edge direction (edge orientation histograms) to extract scale-invariant features transform descriptors of an image. The idea behind this concept is that object appearance can be described as a distribution of intensity gradients and edge descriptors. For this, the image is divided into small connected regions which are called cells and for each cell, a histogram of gradient directions is compiled. From this histogram, we can extract the HOG descriptor of an image. The process of training a model on HOG descriptors can be quite challenging as described in Dalal and Triggs [2005], but here we focus on a simplified description without including detailed parameter selections. First, an amount of P positive samples are sampled from the training data and the HOG descriptor is extracted from these samples. Then also an amount of N negative samples is sampled with the attendance to the rule $N >> P$. After the successful sampling, the HOG descriptors of all the images are extracted. In order to achieve good results, for this step good image pre-processing is important, see Dalal and Triggs [2005]. Now a linear SVM(Support Vector Machine) is trained on the HOG descriptors and once the SVM is trained, the HOG descriptors of the negative samples are used to do hard-negative mining, which is the extraction of false-positives for retraining the

SVM. This retraining through hard-negative mining can be repeated several times, but according to Dalal and Triggs [2005] it does not bring considerate improvements.
The trained classifier can now be used to classify a test set, drawing a bounding once the detection probability overcomes a certain threshold. This usually leads to an object being identified by several bounding boxes which need to be merged into one (usually the largest) by using non-maximum suppression.
HOG descriptors have the advantage over other object detection algorithms because they are invariant to geometric and photometric transformations, which permits ignoring individual body movements of pedestrians due to strong local photo-metric normalization.

YOLO Especially in the field of autonomous driving, having the detection running in real-time is very important. Because of the car's velocity pedestrians have to be detected as fast as possible to leave enough response time for e.g. breaking or other manoeuvres. One of the fastest state-of-the-art object detection algorithms is YOLO (You Only Look Once) [Redmon et al., 2015] and its successor versions [Redmon and Farhadi, 2016, 2018]. In contrast to other state-of-the-art object detection approaches based on region proposal (e.g. Faster R-CNN [Ren et al., 2015]) this algorithm is based on regression. Region proposal based networks are usually relatively slow because they have to process each image multiple times due to their complex pipeline of generating potential bounding boxes, running the classification and post-processing (refinement for bounding boxes, duplicate elimination, rescoring). YOLO, on the other hand, has to process each image only once (hence the name) enabling its real-time capabilities. Separate components of object detection are unified into one single convolutional neural network that is applied to simultaneously predict multiple bounding boxes and their respective class probabilities. Because it does not rely on sliding windows or region proposal but processes the image as a whole the network is able to use global image features to predict bounding boxes and class probabilities. As a trade-off, accuracies lag a bit behind those of region-based approaches. The neural network architecture is inspired by GoogLeNet model for image classification [Szegedy et al., 2015] and consists of 24 convolutional plus two fully connected layers. As activation function a leaky ReLU is used. Only the final layer differs from this as it uses a linear activation function.
Its output is a $S \times S \times (B \cdot 5 + C)$ tensor, dividing the image into a $S \times S$ grid and predicting for each cell of this grid the class probabilities for every of the C classes. Furthermore, it makes predictions for B bounding boxes in every cell consisting of the x and y coordinates of the box's center, width, height and confidence. The network's first 20 convolutional layers are pre-trained with the ImageNet 1000-class competition data set [Russakovsky et al., 2014] after this, Redmon et al. [2015] train the network on the PASCAL VOC 2007 [Everingham et al., 2007] and 2012 [Everingham et al., 2012]. To circumvent overfitting, a dropout layer between the two fully connected layers and extensive data augmentation (random scaling, random adjustment of exposure and saturation) are applied. Furthermore, for the inference, non-maximal suppression is used to handle multiple detections which may appear for large objects or objects close to the border of multiple cells.
In the experiments, it was found that YOLO outperforms other real-time detectors[1] by a factor of at least 2 in mean average precision (mAP) while still being more than 15 fps faster. Furthermore, performs similarly to state-of-the-art region proposal based approaches like Faster R-CNN [Ren et al., 2015] even though these approaches are far away from performing in real-time. Nevertheless, YOLO makes more localization errors than other predictors. Localization errors are those where the correct class label is assigned but the bounding box is misplaced. On the PASCAL VOC 2012 test set YOLO also performs worse than state-of-the-art models mainly due to not detecting small objects that are close to other objects.

[1] Real-time object detectors perform at 30 frames per second (fps) or better

Thus, in the second version **YOLOv2** [Redmon and Farhadi, 2016] the authors tried to overcome some the initial drawbacks. Batch normalization is introduced to improve convergence and performance while also allowing to get rid of the dropout layer. Furthermore, the architecture is changed so that the fully connected layers predicting directly the coordinates of the bounding boxes are replaced and instead the last convolutional layer predicts an offset and confidence for anchor boxes as used in Ren et al. [2015]. Also, training the network using different image resolutions is incorporated to allow the network to detect objects in images of different dimension sizes. Beside training the network on the Pascal VOC datasets the authors also trained the network on the Microsoft Common Objects in Context (COCO) dataset [Lin et al., 2014]. The final performance on the Pascal VOC 2007 dataset is illustrated in Table 1 where one can see that YOLOv2 generates higher precision scores than prior methods while maintaining real-time performance. On the Pascal VOC 2012 and the COCO dataset YOLOv2 also performs comparably accurate to the other state-of-the-art detectors albeit being faster.

Detection Frameworks	**Train**	**mAP**	**FPS**
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
SSD300	2007+2012	74.3	46
SSD500	2007+2012	76.8	19
YOLOv2 288x288	2007+2012	69.0	91
YOLOv2 352x352	2007+2012	73.7	81
YOLOv2 416x416	2007+2012	76.8	67
YOLOv2 480x480	2007+2012	77.8	59
YOLOv2 544x544	2007+2012	**78.6**	40

Table 1: Performance of different detection frameworks on Pascal VOC 2007. "Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model)." [Redmon and Farhadi, 2016]

1.3.2.2 Object tracking

SORT SORT (Simple online and real-time tracking) is a multiple object tracker focusing on simple and effective algorithms, namely Kalman filters and the Hungarian algorithm [Bewley et al., 2016].
For SORT to work the detection quality of the computer vision algorithm has to be very good, because the tracking part of this implementation relies heavily on the detection implementation. With this, it achieves accuracy comparable to other state-of-the-art trackers such as TDAM and MDP but achieves update rates that are approximately 20 times higher than these trackers. Due to it being an online tracker, it only considers the detection from the previous and the following frame, which leads to a loss of the tracking object due to occlusion or a detection failure.
The performance of this tracker mainly lies in its speed, which is achieved by ignoring everything except the bounding box and speed value of the bounding box for data association between frames. Also, long term and short term occlusion are not managed with this tracker and there is no functionality included for re-identifying objects lost to occlusion. This minimalistic formulation of tracking facilitates both efficiency and reliability for online tracking.

Deep SORT Deep SORT got developed by Wojke et al. [2017] in order to address SORTs troubles with tracking objects trough occlusion as well as a high number of identity switches during detection. This is overcome by replacing the association metric of SORT with a more informed metric that combines motion and appearance information.
The association between detection and measurement tracks is done using the Hungarian equation for matching the association metrics.
The motion metric is based on a Kalman filter with eight-dimensional state space $\{u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h}\}$ considering the bounding box center position (u, v), the aspect ratio γ and the height h. For each detection of an object, a track is initiated and new detections are added to this track, until no detection occurs for a certain time and the track gets deleted. The motion metric with the Kalman filter is computed by computing the Mahalanobis distance between the the predicted and measured Kalman state. The motion metric alone does not provide results for temporarily occluded objects and also fails at rapid unaccounted camera motion which introduces rapid displacements in the image plane that lead to a failure of the Kalman prediction.
For this the second metric, containing appearance information, is used. For each bounding box an appearance descriptor is computed and then the smallest cosine distance between the track (containing the 100 last occurences) and the new detection is calculated. If this calculated distance passes a certain threshold the detection is associated with the track.
Both metrics complement each other, with the Mahalanobis distance working ideally for short term tracking and the cosine distance working ideally for long term tracking and temporal occlusion of an object. For this both metrics are combined into one metric using a weighted sum where an association is accepted if the weighted sum is within a gated region of both metrics.
Additionally a matching cascades is introduced that prioritizes more frequent objects as well as tracks with a smaller age in order to prevent unstable tracks and track fractures.
This network is not as efficient as the SORT algorithm, but with access to a high performance GPU, it is still suitable for online tracking.

ROLO ROLO Ning et al. [2016] is a spatially supervised recurrent convolutional neural network that is based on the YOLO object detection. Like Deep SORT, ROLO is able to exploit the history of locations, as well as distinctive features learned by the neural network in order to improve performance. ROLO is a spatially supervised recurrent convolutional neural network that is based on the YOLO object detection. The temporal history is provided by Long Short Term Memory (LSTM) in its Recurrent Neural Network (RNN) which encodes the temporal context information. Also it uses regression for the prediction of the tracking locations instead of binary classification which further improves its performance. With this it outperforms many other image recognition and tracking algorithms while maintaining a low computational cost. But its current capability is still limited to offline tracking and no online tracking capability has been implemented yet.

1.3.2.3 Pedestrian intention recognition

After detecting and tracking a pedestrian the last crucial step is to determine its intention meaning to decide whether it is going to cross the street or not. This is important because nobody would buy a car that is braking and initiating modes of communication whenever there is a pedestrian walking at the side of the road. Instead, human pose and head orientation can be used to infer what a pedestrian is going to do. In contrast to many approaches that have been conducted using images of humans with good resolutions in order to extract these features in the domain of autonomous vehicles images usually are taken from far distances and thus pedestrians only appear in low resolution. This makes it harder to draw correct conclusions about body pose and head orientation. Nevertheless, these problems have been tackled and the most recent approaches are presented in the following.

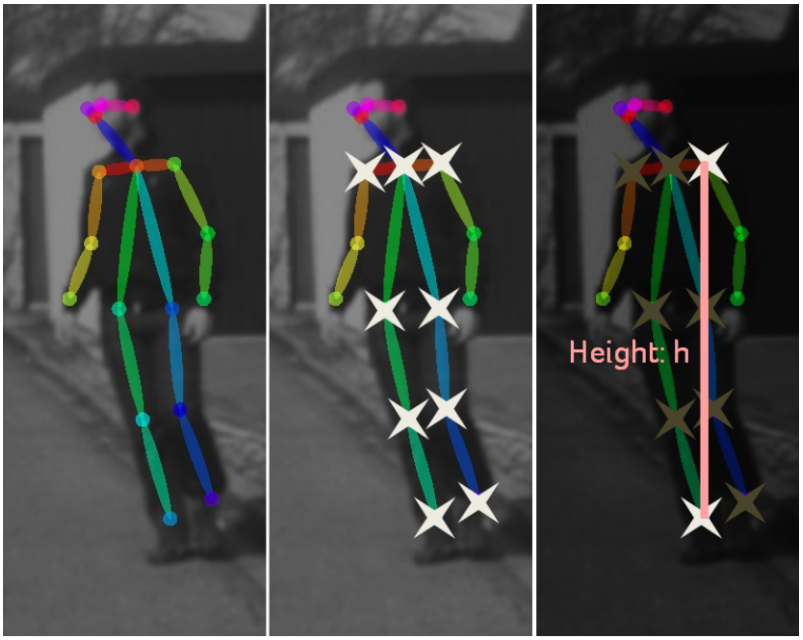


Figure 5: Figure by Fang and López [2018]. The figure shows the skeleton fitting estimating 18 key points. The 9 key points marked with a star are used to predict the pedestrian behaviour. The coordinates are normalized using the height of the person (derived from the topmost and lowermost key point).

Intention from pose estimation Fang and López [2018] proposed a pipeline that "combine[s] CNN-based pedestrian detection, tracking and pose estimation" in order to correctly forecast human behaviour in crossing situations. They train their networks using the Joint Attention for Autonomous Driving (JAAD) dataset (see paragraph 1.3.2.4) to learn from realistic scenarios. For estimating the pedestrian's pose they use a skeleton-fitting approach as described in the work by Cao et al. [2016]. For the feature extraction only the 9 most stable key points of the skeleton (see Figure 5) characterizing the legs and shoulders are used. After normalizing the key points features such as distances between key points as well as angles between couples and triplets of key points are extracted and stored in a feature vector. For the actual behaviour prediction using a Random Forest multiple of those feature vectors are concatenated over time. For a walking pedestrian the authors were able to predict with an accuracy of 0.88 whether a pedestrian was going to cross the street or not. Also for standing pedestrians they were able to determine 250ms after it starts moving if it was going to cross the street or not. A drawback of their described approach, though, is the employed pedestrian detector based on Faster R-CNN which has proven to be not capable of performing in real-time [Zhao et al., 2018].

Intention from head orientation Especially in order to determine whether an approaching car has been seen by the pedestrian the head orientation and gaze are useful cues even though they might not directly correlate with the movement direction of pedestrians as stated by Raza et al. [2018]. Nevertheless, for establishing a better communication between autonomous vehicles and pedestrians this can still be useful to determine. Also, when trying to predict whether pedestrians are going to cross the street after moving parallel to it before (called 'bending' in [Fang et al., 2017])

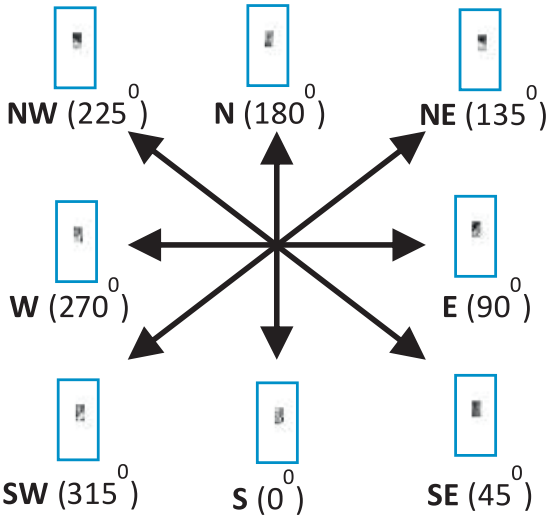


Figure 6: Figure by Raza et al. [2018]. Eight different head orientations that are predicted by the network

detecting a look over the shoulder might indicate this particular action. Most undertaken work until now uses hand-crafted features (see for instance [Rehder et al., 2014]), others such as the most recent research conducted by Raza et al. [2018] use CNNs to learn the features automatically from the dataset. With this approach Raza et al. [2018] accomplish an accuracy of 0.91 for estimating the head orientation (8 discrete classes, Figure 6) outperforming previous approaches. The work only considers the orientation estimation and does not treat the detection of the pedestrian and its respective head itself. However, the orientation estimation can be performed on real-time video sequences.

1.3.2.4 JAAD dataset A dataset introduced especially for the needs of analyzing complex traffic scenes and pedestrian behaviour is the Joint Attention in Autonomous Driving (JAAD) dataset [Kotseruba et al., 2016]. Beside the typically provided bounding box data, it is the first large-scale dataset that also provides behavioral (e.g. standing, crossing, looking, gesture) and contextual (e.g. weather, traffic signals) annotations for the scenes. The dataset contains 346 video clips recorded naturalistic conditions with high-resolution, 30 fps, monocular cameras. Pedestrians are assigned unique labels to allow tracking throughout different camera frames. Furthermore, occlusion information is provided for every bounding box. Some scenes are easier to predict due to good conditions (daytime, clear sky) but the dataset also includes examples of difficult visibility conditions (nightime, sun glare, heavy rain or snow). Figure 7 shows exemplarily the manifoldness of the supplied information. Providing all this information, the JAAD dataset appears to be optimal for the task our work.

1.4 Relevance of the study

The state-of-the-art showed different communication systems for autonomous cars which can be used to interact with pedestrians and how autonomous car can detect pedestrians. The state-of-the-art technologies are focusing on using displays, projections and LED displays. A display indicating a gesture via text, a zebra projection and a LED strips mounted to the grill turned out

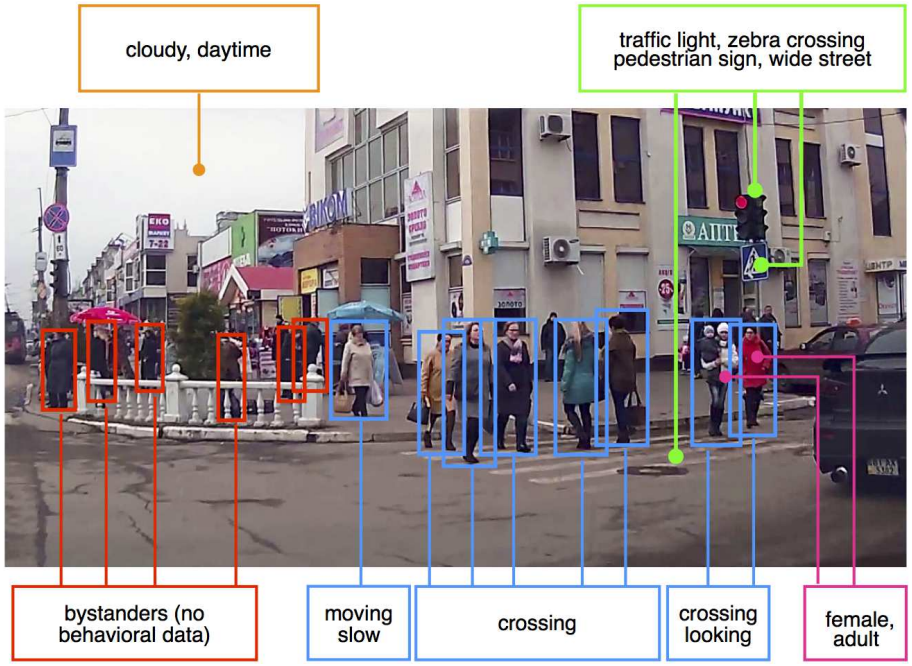


Figure 7: Figure by Kotseruba et al. [2016]. Example of the annotations that are incorporated in the JAAD dataset.

the be the best communication methods. However, all the methods were tested again a null model, a model where no communication system was used. Therefore, the current study will investigate if there one method works better than the other. The best method will be defined as the method where pedestrians perceive the highest amount of effectiveness and trust [Deb, 2017]. Moreover, a system will be designed to detect if a pedestrian wants to cross the street or not. To test which systems work the best we will first test if the different methods work better compared to the null model. Therefore, we will try to answer the following questions:

RQ1: Works the method with LED strips better than null model with regard to effectiveness and trust?
RQ2: Works the method with a projection better than null model with regard to effectiveness and trust?
RQ3: Works the method with a display better than null model with regard to effectiveness and trust?
We will hypothesize based on the state-of-the-art literature that all three the methods will work better than the null model [Clamann and Cummings, 2017],[Daimler, 2017], [Lagstrom, 2015].

Lastly, we will try to find out which models works the best and therefore, try to answer the following question:

RQ4: Which method has the greatest effect size?

2 Requirement Analysis

Because most pedestrian accidents happen when they are crossing the road at an unofficial crossing site [Ward and Evans, 1994], this study will focus on the scenario where a pedestrian is trying to cross the street at an unofficial crossing site, while a self-driving car approaches. A simulation of the self-driving car will be used to test the different proposed gestures. participants will watch a simulation of a self-driving car, meanwhile, they have to imagine they want to cross the road. There is no driver visible in the self-driving car, the pedestrian is sharing the road with a self-driving car.

2.1 Target user

The target user is anybody who uses the roads. Therefore the gestures should be understandable for all road users. Thus it is important to perform the experiment with as many different people. However, because of the relatively small sample size, this study will have, caution is needed when generalizing the results to the general road user. An analysis of the age, gender and nationality have to be done to make sure all groups are represented equally, or otherwise, the results can only be generalized on a particular group. A requirement of the participant is that they should be able to read English and understand the content of the survey.

2.2 Object recognition requirements

For the implementation we plan on implementing an object detection and tracking algorithm that is able to perform in real-time and is robust enough to detect pedestrians throughout changing scenarios and conditions. We will exclude the pedestrian intention recognition part from our work since there are hardly any solutions in current research that meet real-time requirements and implementing such a solution would exceed the scope of this work. Furthermore, in order to train neural networks efficiently we need to make use of a dedicated up-to-date graphics card which none of our laptop computers at home provides. Therefore, will implement our solution in Google Colab. This also brings more advantages with it enables platform independent and shared code development. Section 3.1 provides more information about employing Google Colab.
For the testing of the object recognition capabilities of our implemented system, we will use the average precision value calculated according to the definition by the Pascal VOC challenge 2012 [Everingham et al., 2012]. Since the official code was published in MATLAB, we used a python adaptation that can be found in Cartucho [2019]. The score is calculated by sorting the predicted bounding boxes by decreasing confidence and assigning a ground truth object to the bounding box. Then the intersection over union (IoU) score, also called Jaccard similarity coefficient, is calculated from this match as can be seen in Figure 8. It is considered a match between bounding box and ground truth, if the IoU is $IoU \geq 0.5$. From the IoU criterion a precision recall curve is calculated and then integrated which gives the average precision value. Since we only have one class, no mean average precision value over all classes will be calculated.
We will test the tracking capabilities of our algorithm only visually, since we do not track occlusions and the data-set only contains tracking ids that are consistent throughout occlusions. This makes it impossible to compute a reliable test score without relabeling the data-set or using a different data-set.

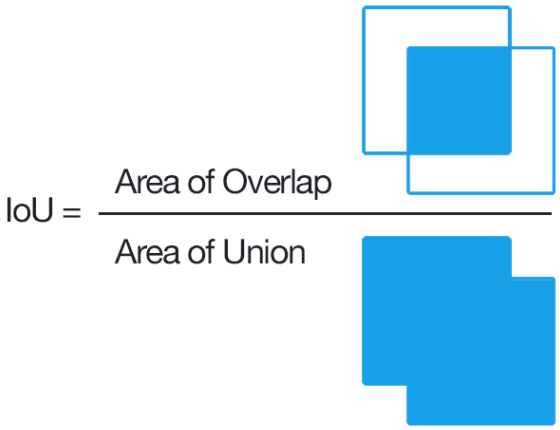


Figure 8: Computation of the IoU score

3 Implementation

3.1 Implementation of the Detection and Tracking Part

The detection and tracking algorithms as presented in section 1.3.2 are implemented using Google Colab, an online Jupyter notebook environment that allows users to use a virtual machine equipped with an Intel Xeon CPU with 2.30GHz, about 13GB RAM and a NVIDIA Tesla K80 GPU with up to 12GB GDDR5 VRAM for free. This set-up enables efficient online training of Machine Learning models with the drawback that the Notebooks runtime is reset every 12 hours so that it is necessary to save checkpoints of the learning process. In the following we will present the steps that we took to get a well working pedestrian detector and tracker. Figure 9 shows our final functional architecture. Furthermore, appendix C explains how to access the code and how it is distributed over the different notebooks.

3.1.1 HOG Detector

For the implementation of the HOG detector we used a pre-trained pedestrian model available in the OpenCV library [Bradski, 2000]. As a first step this detector was implemented and visually checked with images containing pedestrians. As can be seen in Figure 10, the HOG detector gives decent results for images even though bounding boxes are not tightly created around the person but rather loose.
For videos the recognition unfortunately proved to be far worse as can be seen in a video in the HOG folder of the repository, see C. For creating the video of the pedestrian detection, we split the video up into its frames and then performed the HOG pedestrian detection on every single frame and then assembled the collection of frames back into an image. But this does not lead to very accurate results even under good conditions (steady camera, clear view) as can be seen in the video. Also, the frame rate at which images are processed with this approach does not meet our real-time requirements. Therefore, we decide to move on to more recent methods of object detection.

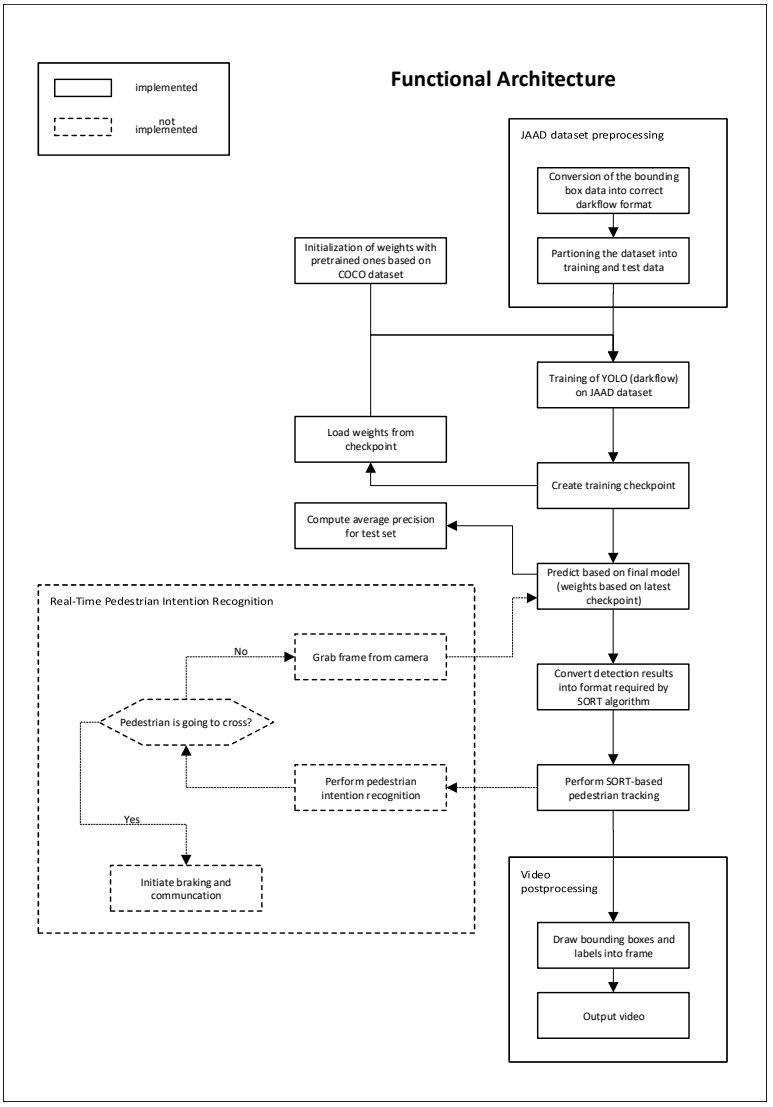


Figure 9: Functional architecture of the pedestrian recognition

3.1.2 YOLO Detector

Preparation The YOLO network was originally written for Darknet [Redmon, 2016], an open-source framework for neural networks that has been written in C and CUDA. In order to make it work with Google Colab it has to be translated into a python framework for neural networks. Trieu [2017] translated Darknet into Tensorflow and called this translation Darkflow. We adapted this implementation for our use. We used the network structure from the YOLOv2 network [Redmon and Farhadi, 2016] and initialized the weights with the by the authors provided weights based on training on the COCO dataset [Lin et al., 2014]. Furthermore, we changed the output layer to only contain neurons for a single class since we are only interested in predicting persons and the number of filters in the second to last (convolutional) layer to $B \times (C + 5)$ with $B = 5$ being the

Figure 10: HOG detector on images containing pedestrians, original (left) and detected (right)

number of bounding boxes predicted per cell and $C = 1$ being the number of classes. The number 5 resembles the 5 parameters that are predicted for each bounding box (center coordinates x and y, width, height and confidence).

Data preprocessing We train our model making use of the JAAD dataset. As previously mentioned, the dataset consists of 346 video clips. The data could not be used right away since some conversion was necessary first to align it with the Darkflow requirements. To train the network, all videos have to be split up into its frames and only the frames containing bounding boxes are kept. This results in 75057 frames showing at least one pedestrian. Also, the annotation data only available in the Matlab Caltech video bounding box (vbb) format needs to be converted into the Darkflow interpretable XML format (see appendix A. The annotation provides information about the directory path and name of the associated image, the image's dimensions and gives for every bounding box on that image label, top left and bottom right corner coordinates, whether it is a difficult object to predict and if the object is occluded or not. Finally, the dataset is split up into a training and test set where the training set contains 67552 frames (90%) and the test set 7505 frames (10%).

Training Due to time limitations training is only performed for roughly 48 hours resulting in 26'500 iterations of batch size 8 (about 3 epochs). The first half of the training a learning rate of $1e-5$ is used, the second half this rate is reduced to $1e-6$. Checkpoints are created every 250 steps to allow interruption of the training and continuation at the exact same spot which is necessary to avoid losing all progress after the 12 hour limit of Google Colab.

3.1.3 Object tracking with SORT

For the SORT implementation we took the output from the YOLO prediction and converted it into the data-structure you can see in Table 2. In the input '.csv' file, the 'Detection ID' field is filled with '-1' and then the SORT algorithms tracks every detection and assigns an ID from its ID counter to it, until it looses the object. IDs increase continually and are not reused once the object is lost.
The three last numbers are ignored by the SORT algorithm and stay on their initial value of '-1'.

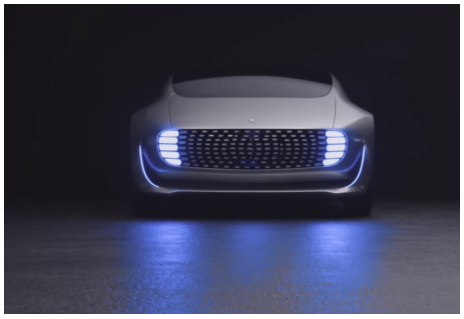
1	2	3	4	5	6	7	8	9	10
Frame ID	Detection ID	top left corner		width	height	Confidence Score	can be ignored		
		x	y						

Table 2: Input and output data structure of the SORT algorithm

From the output file of the SORT prediction the bounding boxes with the tracking IDs are drawn onto the frames with openCV and the frames are then assembled into a video again.

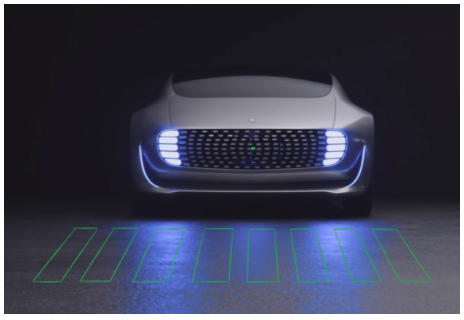
3.2 Implementation of the Interaction Part

To test the interaction of the autonomous car, simulations were created for every interaction we wanted to test including the empty model which refers to the simulation where the car does not have a gesture. The simulation shows a car who approached a pedestrian who wants to cross the street. After the car has stopped, the car will show a gesture. The gesture will be different in every video. Figure 11 shows the four gestures including the empty model.



(a) Car showing no gesture.

(b) Gesture showing arrows.



(c) Gesture showing a projection.

(d) Gesture showing the text: Cross now.

Figure 11: Four gestures that are tested for the interaction

4 Testing

4.1 Testing the Detection Capabilities

The detection capabilities are tested as described in subsection 2.2. Here we achieved an average precision value $AP = 57.19\%$ with our version trained on the JAAD dataset. For comparison, the YOLOv2 algorithm pretrained with the COCO dataset only reached a value $AP = 28.66\%$. This means our training doubled the network's performance.
Unfortunately, we do not have any other networks, that would allow us to directly compare our result for the JAAD dataset. Instead, in order to place our result we looked at the results from the Pascal VOC challenge [Everingham et al., 2013] which range for pedestrians from $AP = 23.4\%$ to $AP = 95.1\%$. This is of course not comparable, since the datasets are different. But it shows that our algorithm provides average performance and that improvement is definitely possible. Our score can be reproduced by loading the Git repository by Cartucho [2019] and the 'testData.zip' mentioned in the SORT notebook, see C.
Our test data-set contains 7505 test frames and YOLO took an average processing time of 0.1133 seconds per image (including the writing process of the file). This results to 8.8 frames per second. In comparison with the framerates reported in the original papers [Redmon et al., 2015; Redmon and Farhadi, 2016] this is significantly slower and not really performing in real-time anymore. The main reason for this is most likely our video data resolution which is 1920x1080 pixels whereas the highest resolution reported is 544x544 pixels.
SORT, on the other hand, took 1.045s to process 270 frames which results in 258.3 fps thus not having any influence on the real-time capabilities of the approach. Examining the tracking capabilities visually indicates that our algorithm is working reliably as can be seen in this video.

4.2 Testing the interaction between vehicle and pedestrian

To test the hypotheses a simulation was created which was shown to the participants. Every participant was shown four videos; one video for every gesture and one video showing no gesture. Participants were asked to fill in a short questionnaire before the videos and one after each video.

4.2.1 Questionnaire

Before the start of the experiment, the participants were asked to fill information about their demographics such as gender, age and nationality. Next, participants had to fill in a questionnaire about their attitude towards autonomous driving, so this factor could be taken into account when analyzing the data. The questionnaire after each video contains questions about the effectiveness of the gesture and the participant's level of trust regarding the system. The questions asked for all three factors (attitude, effectiveness and trust) are from the questionnaire of the study of Deb [2017]. The complete questionnaire can be found in appendix B.
According to Grimm [2010], the collection of data through methods that do not require presence involvement of a researcher can help avoid the social desirability bias to some extent. Since the questionnaire that is used in this research is anonymous and performed online the chances of socially desirable answers is low. For similar reasons the chances of confirmation bias are low. Further, the questions are asked in such a way that they will not hint at the expected outcome. The validation of the questionnaire was performed by one expert in the field of Human-Robot Interaction. The expert concluded that the questionnaire was appropriate for the goal of this experiment.

4.2.2 Procedure

The survey was created using Google Survey and the link was distributed to the participants. Therefore, all the participants received the same instructions. The instruction test was written down as follows:

"For this experiment you will watch four videos. Please pay attention when you watch the video.Before you watch the first video you will have to answer some questions and after each video you will have to answer a couple of questions."

When participants clicked the next-button, the questions appeared. After watching all the videos and answering all the questions a debriefing text was shown. The text stated:

"Thank you for participating in our experiment. If you have any questions, please send a mail to dennis.heijnen@est.fib.upc.edu".

4.2.3 Analysis

After the data was collected, the data was pre-processed for analysis. The questions for each factor of attitude, effectiveness and trust were combined into a single factor. This was done using factor analysis and Cronbachs alpha. The Factor Analysis takes care of combining and checking if it is appropriate to combine the questions into one single factor. The Cronbachs alpha score measures internal consistency, whether several items measure the same general construct.

Next, it was tested if the means of effectiveness and trust were different between gestures and the empty model. Therefore, an ANOVA with repeated measures analysis was performed using effectiveness and trust as continuous independent variables and the categorical variable gestures as the dependent variable. After performing the ANOVA, both assumptions and outliers were checked. Repeated measures were taken into account because the dependent variable is repeatedly tested for every participant. A pairwise comparisons ANOVA was performed to see if and which gestures are different from each other. Lastly, the Cohen's d was calculated for every gesture compared to the empty model to see which gesture has the largest effect size.

4.2.4 Results

4.2.4.1 Demographics For the experiment, 51 participants were recruited. 29 of them were male and 22 were female. Figure 12 shows that most of the participants are between 21 and 25 years old. However, there are participants with a significantly older age.

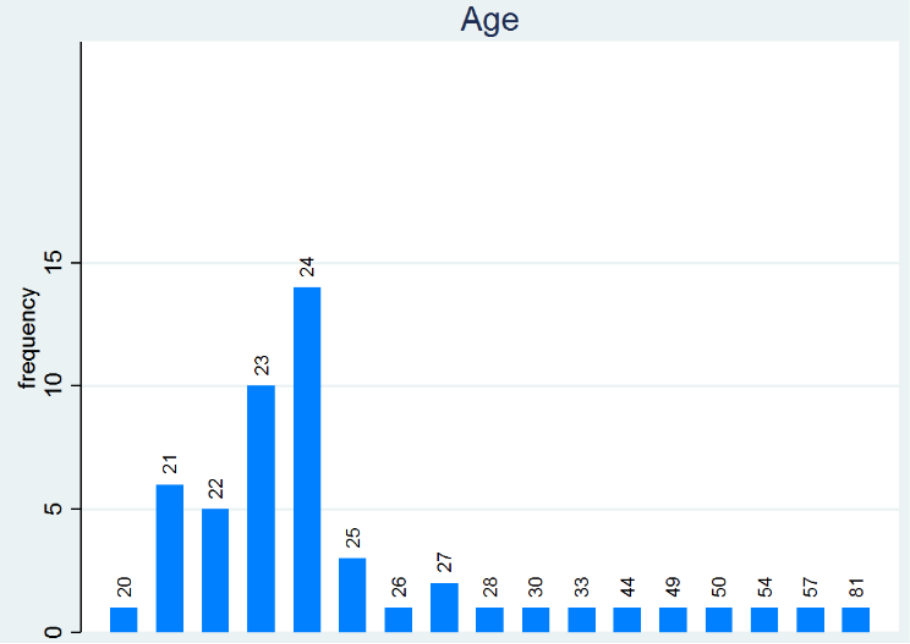


Figure 12: Distribution of age.

Figure 13 shows that most people were recruited from Germany and the Netherlands. This could be expected as the researchers of the experiment are from these countries. Still, 34% are from countries outside Germany and the Netherlands.

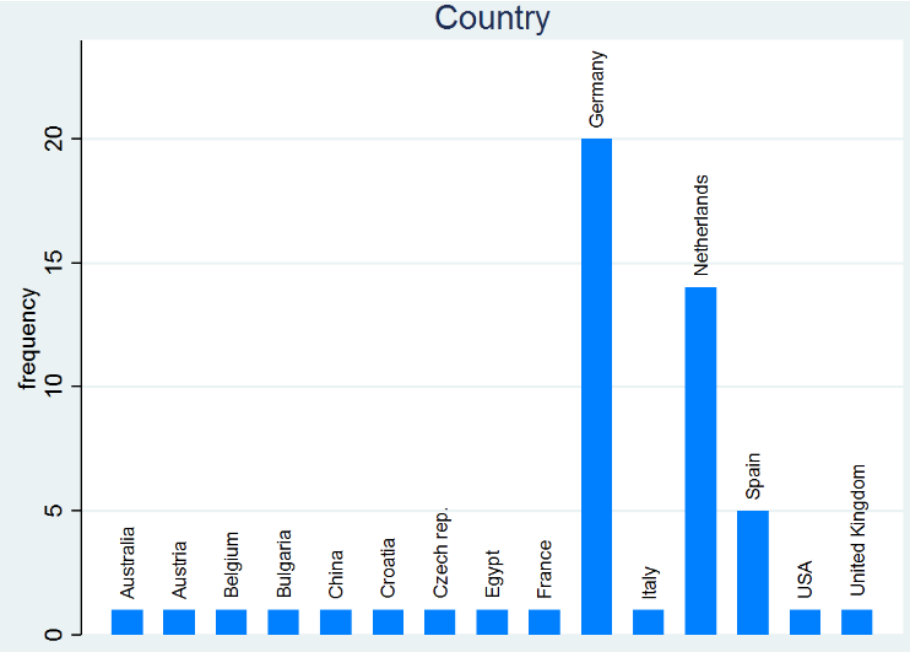


Figure 13: Distribution over countries.

4.2.4.2 Cronbach's alpha The five questions that were used to test the attitude of the participants are combined into a scale that measures the attitude towards self-driving cars. A Cronbach's alpha is calculated for the five questions. An alpha score of 0.64 is obtained, which indicates that the scale is questionable. The item-rest correlation of question 4 seems to be low (0.22), indicating this question does not fit well inside the scale. When looking at the question, "It would take less effort from me to observe the surroundings and cross the road if there are autonomous vehicles involved", the question indeed seems to be different from the rest of the attitude questions (Appendix B). Therefore, question 4 is removed from the scale. The scale for attitude in now improved with an alpha score of 0.68, making it an acceptable scale to measure the attitude of the participants towards self-driving cars.
The distribution of the attitude score of the participants shows that the mean score is 3.49 and ranges from 1 to 5.4. This indicates that there is a lot of difference between participants. Some participants, the ones with high attitude scores, think self-driving cars will have a positive influence and others, with low scores, have a negative point of view. However, most people are still in the middle and thus do not have a strong positive or negative view of self-driving cars.

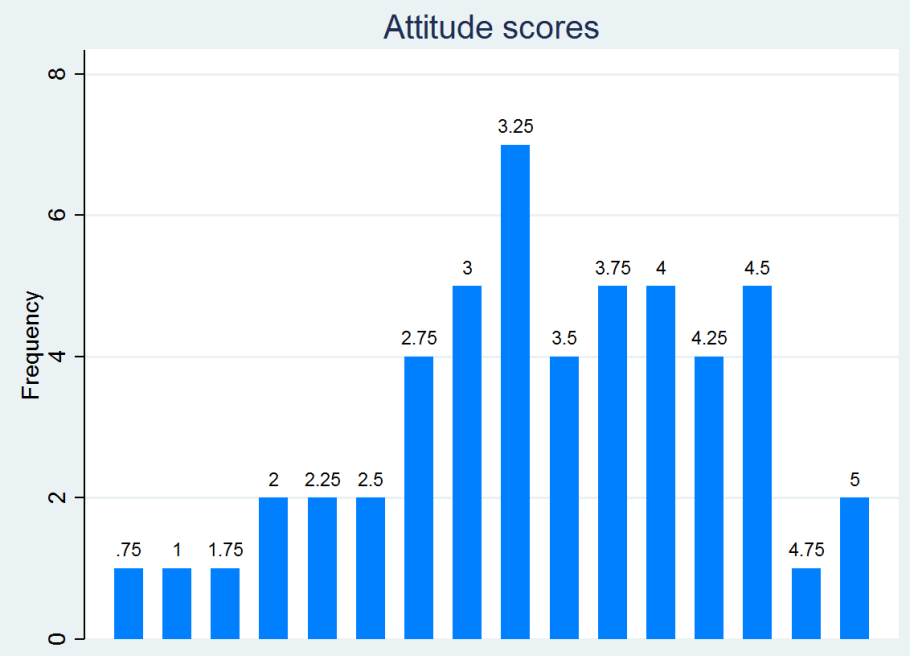


Figure 14: Distribution of attitude.

In order to test the hypotheses, the questions regarding effectiveness and trust need to be combined into scales. The Cronbach's alpha scores for the effectiveness and trust are calculated for each video and shown in the Table 3 below.

Variable	video	Cronbach's α	Reliability
Effectiveness	Empty model	0.75	Good
	Arrows	0.71	Good
	Projection	0.79	Good
	Text	0.70	Good
Trust	Empty model	0.84	Good
	Arrows	0.80	Good
	Projection	0.85	Good
	Text	0.85	Good

Table 3: Cronbach alpha scores for effectiveness and trust

As can be seen in the table all scales have a good reliability and can therefore be used to test the hypotheses.

4.2.4.3 Repeated measures ANOVA Two repeated measures ANOVA tests were performed. In the first test, the influence of gesture on the level of how effective the gesture was was tested. The results are shown in Figure 15. It can be observed that there are differences between the different gestures. The results show that the videos are statistically significant from each other with $F[3, 53] = 66, p < .001$. Moreover, we can observe that the simulation with no gesture is significantly different from the simulations with gestures. The test results of the ANOVA with pairwise comparisons indicates that no gesture is indeed significantly different from the others with $p < .001$ for all three comparisons. In Figure 15 we can see that the gesture showing the text "Cross now" seems to be the most effective. This gesture is indeed significantly different from the gesture showing arrows with $p < .05$. However, there is no significant difference between the textual gesture and the projection ($p = 1.00$).

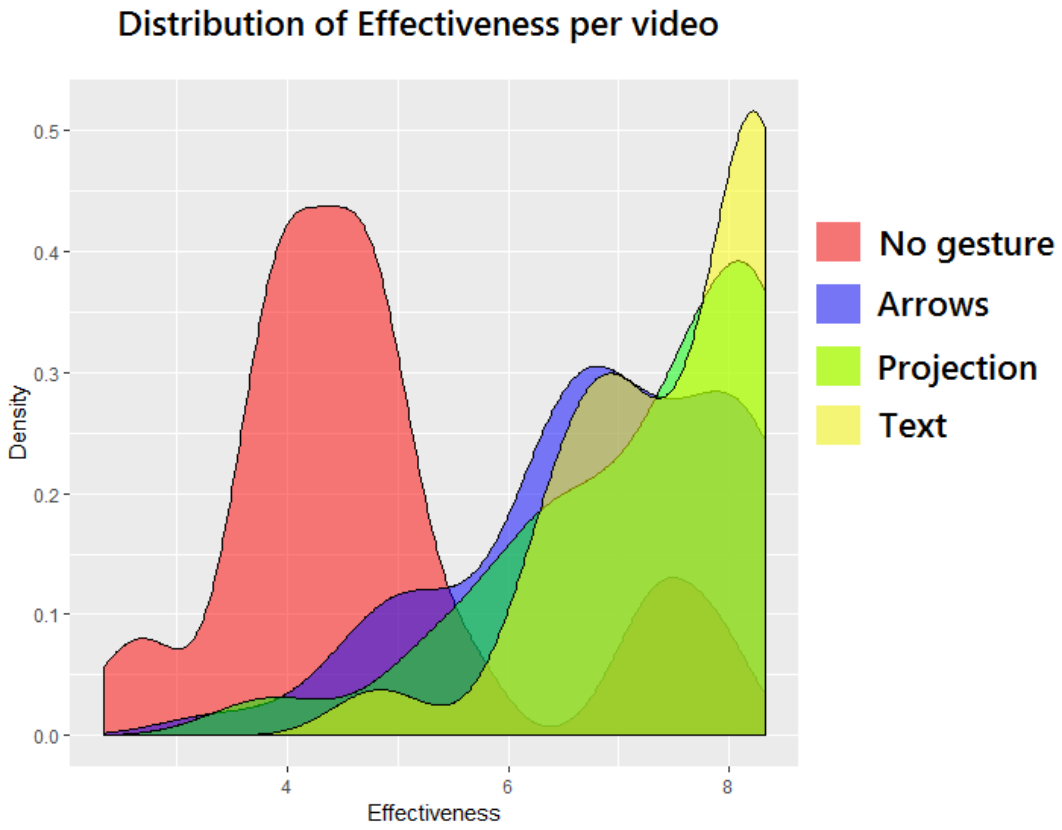


Figure 15: Distribution of effectiveness.

The second repeated measures ANOVA test was used to test if the level of trust depends on the gesture that was shown. Figure 16 shows the results of this ANOVA. Again, we can observe that there are differences between the gestures, which is statistically significant with $F[3, 53] = 23, p < .001$. Furthermore, we see that no gesture is less trustworthy compared to the simulations with gestures. Although, the differences are smaller compared to the differences between gestures regarding the level of effectiveness, the differences are still significant with $p < 0.001$ for all comparisons. However, there are no significant differences between any of the three gestures. For both the first and the second ANOVA test, the assumptions of normality, normality of residuals, independence of residuals and homoskedaticity were met. The standardized residuals did not indicate any outliers for both tests.

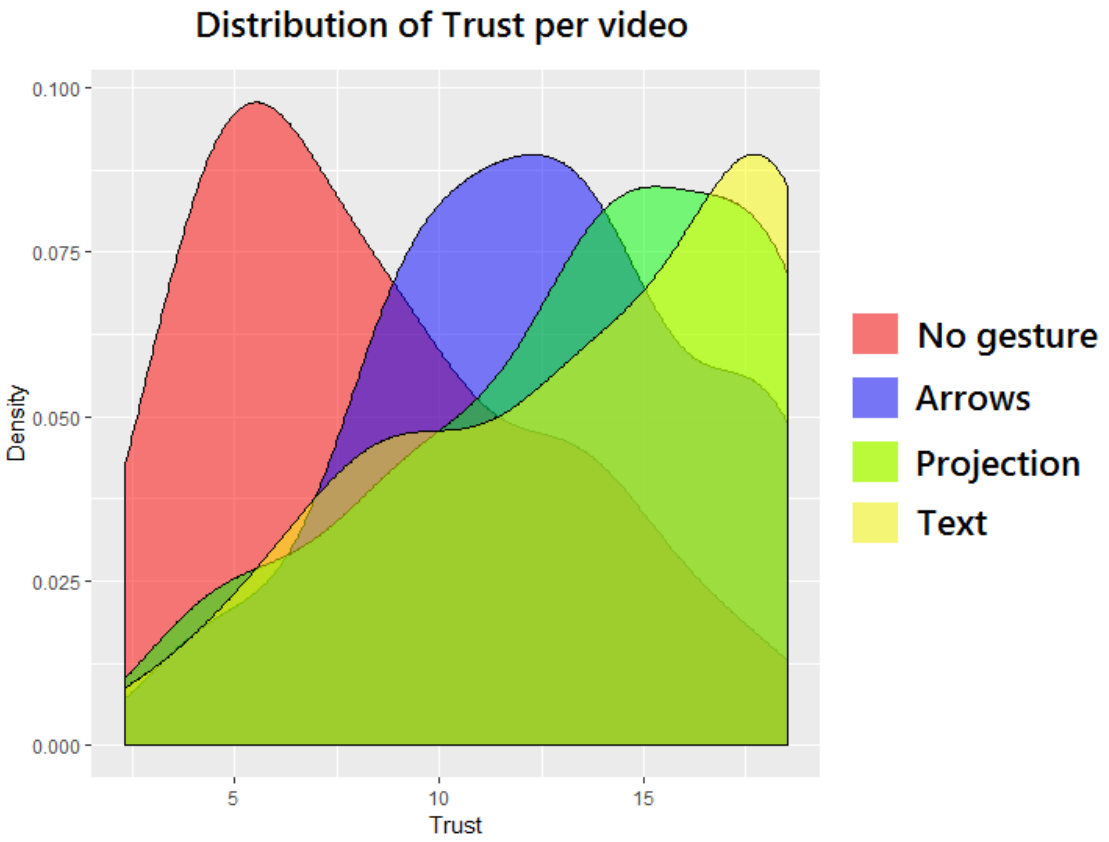


Figure 16: Distribution of trust.

Lastly, we will look at the effect sizes for every gesture compared to the empty model. First, we will look at the effect size for the level of effectiveness. The effect size calculated with Cohen's d of the model using arrows as gesture is 1.58, which indicates that the difference is larger than one standard deviations. The Cohen's d effect size for the model using a projection as gesture is 1.89. The model with a text-display as gesture has a Cohen's d effect size of 2.31.
Next, we will look at the effect sizes for the level of trust. The Cohen's d effect size for the model with arrows as gesture is .97. For the model with a projection as gesture this value is 1.15. The model using a text-display as gesture has a Cohen's d effect size of 1.17.

5 Conclusion

In summary, we have gained useful insights about problems and approaches to overcome them in the field of human-vehicle interaction for autonomous driving. After reviewing current state-of-the-art research, requirements for our project are defined.
Subsequently, the first part of our practical work treats the recognition. For being able to establish a meaningful mode of communication in the first place, the vehicle needs to recognize the pedestrian's

intention. If a pedestrian is going to cross the street the vehicle should behave differently than if the pedestrian is just walking alongside the road. Here, our work focuses on two fundamental steps that enable the intention recognition: Detection and tracking of pedestrians. For this, we employed two state-of-the-art methods that are able to operate in real-time as necessary for autonomous driving. A pre-trained version of the YOLO object detection algorithm has been implemented and fine-tuned, training it with the JAAD dataset which has been designed particularly to improve predictions on pedestrian behaviour. Afterwards, the output of the YOLO network is used by our second module, the SORT tracking method, to track each pedestrian simultaneously over multiple frames. Together they lay the foundations for predicting pedestrian behaviour for example whether a pedestrian is going to cross a street or not. Testing showed, that our fine-tuning significantly improved the performance of the YOLO algorithm on the JAAD dataset. It doubled the average precision (from 28.66% to 57.19%) compared to the baseline, a YOLO network trained on the COCO dataset. Nevertheless, there is still room for improvement and currently the approach is not robust enough to be employed for actual pedestrian detection. In subsection 5.1 we will make proposals on how to overcome the current drawbacks.
The analysis of the interaction experiment will help to test the hypotheses and answer the research questions. The first hypothesis states that the model with arrows as gestures performs better than a model with no gestures regarding levels of effectiveness and trust. The pairwise comparisons ANOVA shows that we cannot reject this hypothesis with our data. Therefore, we can answer the first research question and state that the model with arrows as gesture indeed performs better than the empty model. The same results can be obtained for the second hypothesis. We cannot reject this hypothesis stating that the model with a projection as gesture performs better than the empty model. Hence, we can answer our second research question and say that the model with projection outperforms the empty model regarding effectiveness and trust. The third hypothesis cannot be rejected as well stating that the model with a text-display performs better than the empty model regarding effectiveness and trust. Therefore, the answer to the third research question will be that the model using a text-display performs better than the empty model. From answering the first three questions we can conclude that it is indeed better to use a gesture as a communication tool instead of no gesture. Now we will try to answer the fourth research question to determine which gesture is the most appropriate to use.

When looking at the effect sizes for each model we can observe that the model with a text-display as gesture has the greatest effect size for both the level of effectiveness and trust. Although, the difference between effect sizes regarding the level of trust is small. When only looking at the effect sizes we can conclude that the model using a text-display is the most appropriate model to use when measuring levels of effectiveness and trust. However, the model is not significantly different from the model using a projection regarding the level of effectiveness and not significantly different from both other models regarding the level of trust. Although the text-display gesture seems to be the most appropriate model to use, several drawbacks can be identified. All of our participants were able to read the sign, but in the real world, there might be people who are not able to read the sign such as children or illiterate people. Moreover, people might not be able to understand English or when the sign is presented in a native language in native people might not be able to read the sign. These drawbacks combined with the non-significant differences between other models might be a good reason to not choose the model with a text-display as the best model despite the large effect size. We will address the limitations and possible opportunities for future research which might help to get a better definition of the most appropriate model to use.

5.1 Limitations and Future Research

The testing of our pedestrian detection algorithm showed that it is not yet robust enough to detect pedestrian reliably. Beside changing the network structure, a first step should be to have the network train longer with changing learning rates. We only trained the network for 3 epochs where in the literature networks are often trained for several dozens up to a few hundred epochs. Also, during training it often seemed like the network got stuck in a local minimum. Changing the learning rate could helping escaping those. Furthermore, we found that the network was only operating at around 10 fps which is not performing in real-time anymore. This is mainly due to the fact that our video material is of much higher resolution than the videos used by Redmon and Farhadi [2016]. In future work, the performance should also be examined when trading the good resolution for better framerates. At a resolution of 544×544 pixels the network should perform with 40 fps (see Table 1).
Regarding the interaction between self-driving car and pedestrian, our research was conducted with the help of simulations. This helped giving insights on how to detect pedestrians and how to effectively communicate with them. However, before such systems can be deployed in real-world settings, extensive testing in real situations should be done. Due to time and budget constraints, this research was not able to use real cars. The use of real cars, in a controlled setting, should be the next step in the pedestrian interaction experiments.

References

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. *CoRR*, abs/1602.00763.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cao, Z., Simon, T., Wei, S., and Sheikh, Y. (2016). Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050.

Cartucho, J. (2019). mean average precision - this code evaluates the performance of your neural net for object recognition. https://github.com/Cartucho/mAP.

Clamann, M. Aubert, M. and Cummings, M. L. (2017). Evaluation of vehicle-to-pedestrian communication displays for autonomous vehicles. *arXiv preprint arXiv:1708.07123*, (17-02119).

Daimler (2017). Overview: Mercedes-benz f 015 luxury in motion.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893.

Deb, S., S. L. C. D. W. D. J. S. B. . G. T. M. (2017). Development and validation of a questionnaire to assess pedestrian receptivity toward fully autonomous vehicles. *Elsevier*, 84:178–195.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://host.robots.ox.ac.uk/pascal/VOC/voc2007/.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://host.robots.ox.ac.uk/pascal/VOC/voc2012/.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2013). Assessing the significance of performance differences on the pascal voc challenges via bootstrapping. http://host.robots.ox.ac.uk/pascal/VOC/pubs/bootstrap_note.pdf.

Fang, Z. and López, A. M. (2018). Is the Pedestrian going to Cross? Answering by 2D Pose Estimation. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018-June:1271–1276.

Fang, Z., Vázquez, D., and López, A. M. (2017). On-board detection of pedestrian intentions. *Sensors (Switzerland)*, 17(10):1–14.

Grimm, P. (2010). Social desirability bias. *Wiley International Encyclopedia of Marketing*.

Kotseruba, I., Rasouli, A., and Tsotsos, J. K. (2016). Joint attention in autonomous driving (JAAD). *CoRR*, abs/1609.04741.

Lagstrom, T., L. v. (2015). Avip-autonomous vehicles interaction with pedestrians.

Li, F.-F., Johnson, J., and Yeung, S. (2018). CS231n: Convolutional Neural Networks for Visual Recognition [Course slides, lecture 11].

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Mahadevan, K. Somanath, S. and Sharlin, E. (2018). Communicating awareness and intent in autonomous vehicle-pedestrian interaction. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*.

Masahiro Mori, K. F. M. and Kageki, N. (2012). The uncanny valley. *IEEE*, 19(2):98–100.

Matthews, M. Chowdhary, G. and Kieson, E. (2017). Intent communication between autonomous vehicles and pedestrians. *arXiv preprint arXiv:1708.07123*.

Ning, G., Zhang, Z., Huang, C., He, Z., Ren, X., and Wang, H. (2016). Spatially supervised recurrent convolutional neural networks for visual object tracking. *arXiv preprint arXiv:1607.05781*.

Rasouli, A. and Tsotsos, J. K. (2018). Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *arXiv preprint arXiv:1805.11773*.

Raza, M., Chen, Z., Rehman, S.-U., Wang, P., and Bao, P. (2018). Appearance based pedestrians head pose and body orientation estimation using deep learning. *Neurocomputing*, 272:647 – 659.

Redmon, J. (2013–2016). Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Rehder, E., Kloeden, H., and Stiller, C. (2014). Head detection and orientation estimation for pedestrian safety. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pages 2292–2297.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2014). Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575.

Shaheen, Susan, H. T. and Stocker, A. (2018). Future of mobility white paper.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Trieu, T. H. (2017). Darkflow. https://github.com/thtrieu/darkflow.

Ward, H., C. J. M. A. A. R. and Evans, A. (1994). Pedestrian activity and accident risk. *IEEE*.

Winkle, T. (2016). Safety benefits of automated vehicles: Extended findings from accident research for development, validation and testing. *Springer, Berlin, Heidelberg*, pages 335–364.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402.

Yang, S. (2017). Driver behavior impact on pedestrians' crossing experience in the conditionally autonomous driving context. *DiVA*.

Zhao, Z., Zheng, P., Xu, S., and Wu, X. (2018). Object detection with deep learning: A review. *CoRR*, abs/1807.05511.

Appendices

A Ground truth bounding box XML format

```
<annotation>
    <folder>/content/JAAD/test/vid</folder>
    <filename>video_0119_180.jpg</filename>
    <source>
        <database>JAAD_clips</database>
        <annotation>JAAD_pedestrian</annotation>
        <image>JAAD</image>
        <url>http://data.nvision2.eecs.yorku.ca/JAAD_dataset/</url>
    </source>
    <size>
        <width>1920</width>
        <height>1080</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>person</name>
        <bndbox>
            <xmin>459</xmin>
            <ymin>453</ymin>
            <xmax>640</xmax>
            <ymax>952</ymax>
        </bndbox>
        <difficult>0</difficult>
        <occlusion>0</occlusion>
    </object>
    <object>
        <name>person</name>
        <bndbox>
            <xmin>1520</xmin>
            <ymin>558</ymin>
            <xmax>1572</xmax>
            <ymax>682</ymax>
        </bndbox>
        <difficult>0</difficult>
        <occlusion>0</occlusion>
    </object>
</annotation>
```

B Questionnaire

The questionnaire as presented to the participants. Every question was answered on a 7-point Likert scale ranging from 1 is totally disagree until 7 is totally agree.

Attitude:
1: Autonomous vehicles will enhance the overall transport system.
2: Autonomous vehicles will make the roads safer
3: I would feel safe to cross the road in front of an autonomous vehicle
4: It would take less effort from me to observe the surroundings and cross the road if there are autonomous vehicles involved
5: Automated vehicles are a danger for the road safety

Effectiveness:
1: Interacting with the system would not require a lot of mental effort
2: The autonomous vehicle has stopped for me
3: The gestures are easy to understand
4: I do not know the intention of the system

Trust:
1: I feel safe to cross the street
2: I can now cross the street
3: I have to wait to cross the street
4: I do not feel comfortable to cross the street

C Accessing the code

The code is written in Google Colab notebooks, in order to make it easily executable. All notebooks are written in a way to make them executable without changing parameters. For this to work, every notebook loads the required intermediary results provided by us. If you would like to use your own results you need to save the results from the previous step in the functional architecture and provide it to the notebook as input accordingly. If there are any questions regarding the code, please contact Emanuel or Julian. You can find the contact information in the title page.
The code can be accessed here (a google account is required):
`https://drive.google.com/open?id=1MZr5uucCcfi_AcU28f9WxDC8o6CKf6WT`
The folder contains a PDF file with the functional architecture of our code and four folders.
The folder 'Additional Code' contains our initial experimentation with the HOG detector and also its resulting video file. Also our attempted implementation of the ROLO code can be found here.
The folder 'JAAD Dataset Preprocessing' contains our code for the dataset processing such that the videos and annotations can be interpreted by the YOLO Darkflow network. For this, the notebook first downloads the dataset from the respective websites, then the set is split up in smaller batches that are converted individually. This is due to instabilities that have been encountered when processing the whole dataset at once. After conversion, the dataset is split up into training and test set and broken links between annotation and video files that occur due to moving the files after conversion are restored.
Then the 'YOLO Pedestrian Detection' folder contains two notebooks, one with the training of the network and prediction of the bounding boxes ('YOLO_PedDetection.ipynb') and another one for

saving checkpoints ('SaveCheckpoints.ipynb'), as Google Colab only allows for 12h of continuous computing time and our training took longer. The YOLO notebook first installs all darkflow dependencies. Then it contains code for training the network on the JAAD dataset which includes modifying the model structure, setting training options (batch size, number of epochs, learning rate and GPU usage), loading weights from pretrained networks or restoring checkpoints and starting the actual training. After training is finished, the prediction section can be executed where a pretrained model and our self-trained model are both applied to the test samples of the JAAD dataset. Results are stored and exported.
In order to compute the average precision scores as desribed in section 4.1 one needs to make use of the code by Everingham et al. [2013]. For this, the predicted bounding boxes are already exported in the correct format but the ground truth boxes need to be converted first. The respective converter can be found in the repository's 'extra' directory as is called 'convert_gt_yolo.py'. After conversion the files need to be moved into the correct folders as specified in the repository's readme file. The AP score can the simply be computed by executing the 'main.py' file.
Last but not least in the fourth folder 'SORT Object tracker', the SORT tracker is implemented. This notebook takes the YOLO prediction on the test videos and is currently implemented in a way that it outputs one of the test videos if it is run. The other test videos can easily be shown by modifying the code as instructed in the notebook. The result video is also shown in the folder.