Future Urban Roads

Bente ter Borg
Lodewijck Foorthuis
Julian Tas
Tim van Zee

Delft University of Technology
Faculty of Mechanical, Maritime and Materials Engineering
Department of Biomechanical Engineering

December 21, 2018

First Supervisor: F.A. Dreger, MSc — TU Delft
Second Supervisor: Dr. ir. J.C.F. de Winter — TU Delft
Third Supervisor: J.C.J. Stapel, MSc — TU Delft

This page is intentionally left blank.

A Study into the Scenarios where Motorized Vehicles and Vulnerable Road Users Communicate

Bente ter Borg, Lodewijck Foorthuis, Julian Tas and Tim van Zee

***Abstract*—Autonomous Vehicles (AVs) do not contain a human driver at the driver's seat. Therefore, AVs may need to communicate with Vulnerable Road Users (VRUs) by means of artificial gestures, sounds or lights. However, it is currently unknown in which types of traffic scenarios human drivers communicate with VRUs. The aim of this research is twofold: (1) to examine in which types of scenarios motorized vehicle-VRU communication is prevalent, and using this data (2) to examine in which traffic scenario communication is worthwhile. This second traffic scenario is built as candidate scenario in an experimental setup in virtual reality, wherein an AV communicates with a VRU. Past research suggests for the second aim a situation with unclear traffic laws, low speeds and an unclear visibility. In total 101 YouTube videos were analyzed using observation metrics: *severity of the conflict, traffic situation, location, visibility, type of communication, mobility, lanes to cross, priority, violation of the law and vehicle type*. Also *relative velocity*, *conflict time* and *absolute velocity* were measured. Results showed that *no communication* occurred more at high speed scenarios ($\geq$ 51 [kph]) than at low ($\leq$ 30 [kph]) and intermediate (31 - 50 [kph]) speed scenarios. When there was *no violation of the traffic law* more communication takes place by VRUs and vehicles compared to when there was *violation of the traffic law*. In the case of blocked view, accidents were more likely to happen. Blocked view mostly occurred due to a stationary vehicle. Accidents happen more at intersections than at the other traffic situations. In conclusion, based on the YouTube analysis, communication is worthwhile in a low and intermediate speed scenario, where there is no violation of the traffic law and where the VRU's visibility is blocked by a stationary vehicle. This scenario is built in Unity and could be tested with different situations in further research.**

***Keywords*—Autonomous Vehicles, Communication Scenario, Vulnerable Road Users, Unity.**

1 Introduction

More and more car manufacturers, like Mercedes and Toyota, are joining the development of Autonomous Vehicles (AVs), whereas companies like Google and Uber are already testing their AVs on the public road. It can be expected that these vehicles will be exemplary road users who will obey the traffic rules and drive safely. The development of the AV, however, raises the question how Vulnerable Road Users (VRUs) will respond to and behave around AVs even if they drive safely. Someone behind the wheel reading a newspaper or even nobody driving the vehicle might cause confusion. Therefore, car manufacturers are adding displays or warning lights to their AVs to improve their communication with VRUs. This paper aims to investigate in which types of traffic scenarios AV-VRU communication may be worthwhile. The main focus of this paper is to investigate traffic scenarios where communication between motorized vehicles and VRUs is prevalent. The second focus is to investigate where communication could be worthwhile in traffic scenarios. The scenario where communication could be worthwhile is built in Unity.

2 Background

The Institute for Road Safety Research in the Netherlands, SWOV, stated that the decision-making and behaviour of pedestrians and cyclists in interaction with AVs have received little attention in the research community [1]. Past research mostly focused on the type of communication, rather than on the scenario in which the communication took place. In most cases a scenario is chosen with a single vehicle and a single pedestrian on a crosswalk. There is a lack of focus on different types of scenarios that can occur on the road. It is important to research multiple road situations. Otherwise the research will not give a useful insight into the real-life road situation.

However, past research about communication could give insight in possible scenarios. The research of Zimmermann shows that VRUs interpret driving behaviour, like deceleration, as positive communication [2]. This conclusion was also drawn by research done by Rothenbücher [3]. This research indicates that if an AV showed a significant deceleration, pedestrians were confident to cross the road. However, when the AV behaved unexpectedly in terms of speed or yielding, pedestrians showed discomfort with the AVs and were looking for a way to communicate. Furthermore, it has been shown that pedestrians seemed comfortable with the AVs as long as the AV behaves according to the traffic laws [4]. VRUs are less comfortable at a scenario where traffic laws are unclear and are more willing to communicate.

Apart from dealing with AVs, pedestrians seem to have difficulties with recognizing danger. When vehicles approach at higher speeds, pedestrians find it more difficult to determine whether a vehicle is decelerating, since the relative acceleration is harder to perceive [5]. These factors suggest that a way of communication between AVs and VRUs is necessary to guarantee safety and make VRUs comfortable with the AV. Looking at the cause of accidents in Europe, 14% is due to a communication error and 24% is due to interpretation and planning errors [6]. Interpretation and planning errors imply the analyzing of the current and predicted behaviour of other road users. A factor that has been identified as a contributory factor in the causation of pedestrian crashes and injuries is the speed of motorized vehicles [7]. An experiment from Katz et al. [8] showed that drivers slowed down or stopped more often for a pedestrian when their approaching speed was low. Another factor that contributes to the causation of pedestrian crashes is the visibility of the pedestrian [7].

Thus, a scenario where communication between VRUs and AVs occurs can be compared to the current situation between VRUs and vehicle drivers. From past research, it seems this is a situation with unclear traffic laws, low speeds and clear visibility.

Since there is a need to better understand which traffic situations require communication, we did a structured analysis of road videos. To ensure a vast database, a broad analysis of YouTube content was performed. YouTube is an international platform that allows people to upload videos. Every minute, more than 48 hours of video content is uploaded [9]. YouTube is also connected to other big social networks, such as Facebook, Twitter and Google+. In this way, the off-site sharing of YouTube videos is made easy. Due to the broad range and the huge amount of uploaded videos on YouTube every minute, YouTube is used in our research.

3 Method

Despite not being part of the initial goal of the research, we have started a real-life observation pilot with a mounted camera on the side of the road to get more insight in communication. This pilot showed that communication rarely takes place in real-life (an estimated one in twenty road users). Since other research into real-life situations has already been done [4] and due the lack of time, this pilot was not integrated any further. This research is therefore limited to YouTube analysis.

3.1 YouTube Search

Firstly, we selected YouTube videos which displayed communication. These videos were found by the use of search terms. The search terms were based on the severity of the conflict. The four categories in increasing order of severity were: (1) *No delay and no discomfort*, (2) *Confusion and waiting*, (3) *Agitation, near miss evasion* and (4) *Angry, accident*. The conflict is linked with an emotion which is sorted in increasing order of severity from no discomfort to angry. The search terms are shown in Table 3 in Appendix A. The search terms per category were combined to find the videos. The goal was to find 20 to 30 videos in total for each category and not to find a certain amount of videos per search. The research resulted in a total of 101 videos that were analyzed (Appendix F).

3.2 observation metrics for YouTube Data Extraction

We analyzed the videos using categories, so all the data extracted from YouTube is stored in a structured way. All the categories we used are observation metrics (Table 1 on the next page). Due to the possibility of multiple vehicles and VRUs in a video the observation metrics were divided in a general part, a VRU part and a vehicle part. Therefore, multiple types and number of communication can occur in one video. When VRUs or multiple vehicles on the same lane were acting the same or as a group, we counted them as one.

3.3 Inclusion Conditions of YouTube Videos

In order to make the video search more reliable we filtered the selected videos. The following criteria were used to filter the videos:

- At least one vehicle;
- At least one VRU;
- No compilations.

Compilations were filtered out, because most compilations consist of sensational videos and not real-life situations. In order to filter out compilations, we used the YouTube search filter. The filter applied to the search was < 4 minutes, since most compilations > 4 minutes. The < 4 minute compilations were filtered out post-hoc. The YouTube filter is also used to sort the videos by number of views in order to keep the same sequence when the search is reproduced. For every combination of search terms, the number of watched videos and the number of videos that were actually useful for this research were noted down, to keep track of the usefulness of the different search terms and to make it easier to retrieve subsequent videos.

Compilations were allowed in case not enough videos are found in a certain category which suffice to the inclusion criteria. Since the videos on YouTube where good interactions take place are less common, this exception is applied for *No delay and no discomfort* and *Confusion, waiting*.

Table 1: observation metrics for interface with left the categories, centered the option(s) and right the description.

General observation metrics		
Severity of the conflict	No delay and discomfort Confusion and waiting Agitation, near miss or evasion Angry, accident	The state of the VRU after the interaction.
Traffic situation	Crosswalk Intersection Roundabout Not defined	The road situation where the VRU is crossing. Crosswalk is a crossing over a road without another crossing. By intersection and roundabout is meant crosswalk adjacent to these situations. A not defined situation is a road without a designated crossover.
Conflict time	Quantity [s]	The moment a VRU starts crossing until the moment the VRU leaves the road or is hit by a vehicle.
Number of VRUs	Quantity	Number of VRUs in the video. Multiple VRUs who acted as a group are counted as one.
Number of vehicles	Quantity	Number of vehicles in the video. Multiple vehicles are counted as one if they acted as a group.
Location	Country	The country where the situation takes place.
VRU observation metrics		
Visibility	Clearly visible Difficult to notice Hardly visible, blocked view	The options are clearly visible, difficult to notice (still visible, but not right away) or hardly visible, blocked view (not visible due to darkness or another vehicle blocking the view)
Communication	Eye contact Positive hand gesture Negative hand gesture No communication	The communication options are eye contact (the VRU looks at the vehicle), positive hand gesture, negative hand gesture and no communication.
Mobility	Child Mobile Walking aid	The ability of the VRU to cross the road: a child could behave unpredictable, someone with walking aid needs more time to cross the road.
Lanes to cross	Quantity	If multiple lanes are split by a traffic island, only the lanes up to the traffic island are considered.
Priority	Yes No Unclear	Whether the VRU had priority following the traffic rules.
Violation of the law	Yes No	Whether the VRU obeys the traffic rules.
Vehicle observation metrics		
Relative velocity	Lower than max According to max Higher than max	The velocity of the vehicle relative to the allowed maximum speed.
Absolute velocity	Quantity [kph]	The velocity extracted from YouTube.
Communication	Driving behaviour Sound or horn Light signals No communication	The kind of communication used by the vehicle. Options are sound or horn, light signals and no communication. Multiple options are possible for one vehicle.
Vehicle type	Motorcycle Car Truck or Bus	Type of vehicle.
Priority	Yes No Unclear	Whether the vehicle had priority following the traffic rules.
Violation of the law	Yes No	Whether the vehicle obeys the traffic rules.

3.4 Speed Extraction of YouTube Videos

We created a database with the videos and the observations were annotated using a custom Matlab interface. With the built graphical user-interface (see Figure 13 in Appendix B) the observations can easily be analyzed and will automatically be uploaded to the corresponding Matlab table. We used a second interface (see Figure 14 in Appendix B) for a correction of the data regarding communication. This way, the country and the communication per VRU and per vehicle was added (Table 1).

The absolute velocity in kilometers per hour was measured using estimated distances from the videos. The exact location of the situation in the video was found by either entering the coordinates in Google Maps or by looking at the environment. When an exact location of the situation could be found, we used the Google Maps distance measuring tool to determine the distance between the two landmarks. This is also shown in Appendix C. When this was not possible, the lengths of other vehicles in the video were used. When the distance was determined, we measured the time in [s] it took the vehicle to cover this distance. Then the velocity is calculated by dividing the covered distance by passed time.

Both the conflict time and time used for calculation of the velocity are measured using a stopwatch. The video playback speed was slowed down four times in order to make the time measurement more accurate.

3.5 Error of Distance and Speed Measurements

The use of Google Maps results in a distance error of 0.44% [10]. Distances estimated using other vehicles can be expected to be reasonably accurate as the length of a certain vehicle type could be easily extracted from the internet. Also, there is an error due to inaccuracy of the time measurement. The time measurement had to be started and stopped manually when the vehicle crossed the start and end marks of the known distance. Distortions in the video and reaction time of the people who measured may result in an error when measuring the time. Because an average velocity is taken over a small distance, it will be an approximation of the absolute speed. The absolute speed is used to make a distinction between low, intermediate and high speed scenarios. According to the urban speed limits, the low speed scenario is set as ≤ 30 $[kph]$ and the high speed scenario as ≥ 51 $[kph]$. The intermediate speed scenario is set as 31 - 50 $[kph]$. The speed measurement was done twice for each video by different annotators and the measurements are accurate within a 10% margin.

To measure the conflict time we used a stopwatch with an accuracy of *0.01 [s]*. To obtain an error margin for the time the measurement was done by two people and the error margin of *0.5 [s]* is assumed. The video playback speed was slowed down four times, which decreases the error margin to *0.125 [s]*. This margin is set to account for reaction time errors from the people who measure.

Most of the data will be on a nominal scale. For this reason we will mostly use bar graphs and pie charts. In these kind of graphs it is hard to plot the margin of error. To address this we will also plot graphs with the outer boundaries of the error margins. These will be compared to the graphs with the unedited data to see if it significantly influences our results.

4 Results

During the analysis of the data we found that the traffic situation *roundabout* occurred only once, so this scenario is excluded from the results.

4.1 Influence of Vehicle Velocity on Communication

From the box plot with the vehicle velocity plotted against communication (Figure 1), it appears that *no communication* is most common at high speeds. *Sound or horn* occurred mostly at high speeds too. Low and intermediate speed scenarios seem to have more communication for both the vehicle and the VRU.

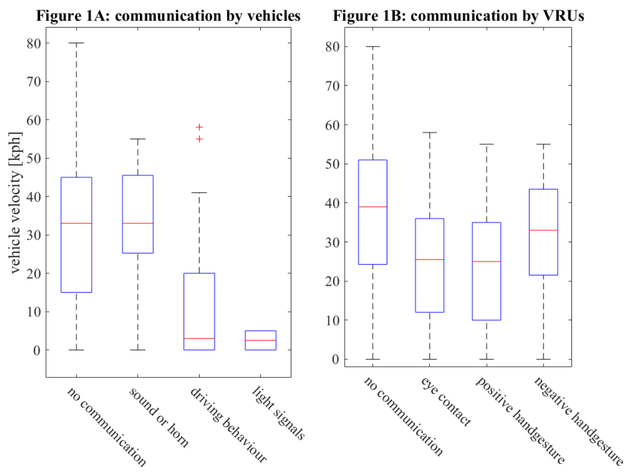

Figure 1: Vehicle velocity plotted against A. vehicle communication and B. VRU communication.

In Figure 2 it can be seen that the frequency of *no communication* increased when the velocity of the vehicle increased. The *no communication* by the VRU is higher for 0 $[kph]$ than for $1 - 40$ $[kph]$.

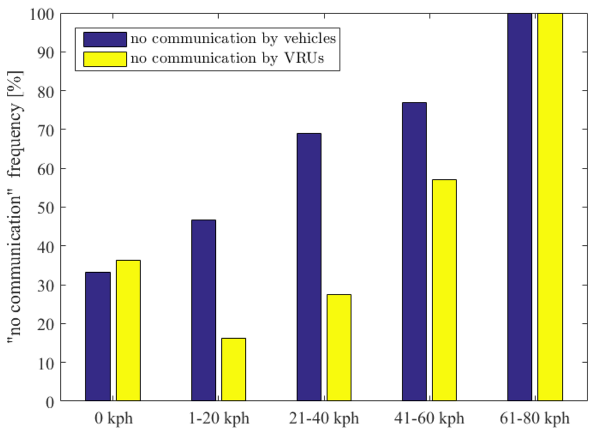

Figure 2: The occurrence of no communication plotted with a 20 *kph* interval, as percentage of the total number of counted communication cases.

In Figure 3 the types of communication are shown at different speed scenarios, namely low, intermediate and high. In the low speed scenario, 73% of the cases showed communication and 27% showed *no communication*. In the intermediate speed scenario, 66% of the cases showed communication and 34% showed *no communication*. In the high speed scenario, the main type of communication was *no communication* with 75% of the total frequency. *Eye contact, positive and negative hand gestures* made up 25% of the total.

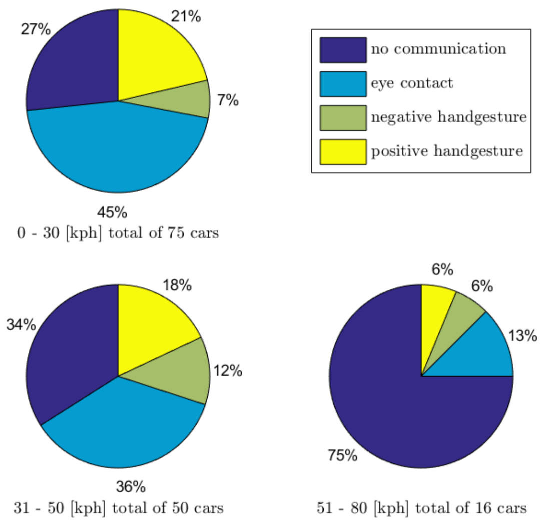

Figure 3: The different types of communication plotted against the three speed scenarios; low, intermediate and high for the three different types of traffic situations, by number of conflicts.

To take the error in speed measurement into account three additional plots were made. In Figure 19 in Appendix D the VRU part of Figure 2 is plotted with an error of 10%. The same has been done for Figure 3. In Figure 20 in Appendix D the plot can be seen with an adjusted speed of 90% and in Figure 21 in Appendix D the plot can be seen with an adjusted speed of 110%.

4.2 Types of communication when (no) Violation of the Law occurred

Figure 4 shows the difference in communication for VRUs in case there was *(no) violation of the law*. From this figure, it seems that in case there was *no violation of the law*, *no communication* occurred less by VRUs compared to a situation when there was *violation of the law*. In case there was *no violation of the law*, *positive hand gestures* made up 35% of the total. In case of *violation of the law*, there was only 13% *positive hand gestures*. In case of *no violation of the law*, *no communication* made up 12% of the total. In case of *violation of the law* there was 42% *no communication*.

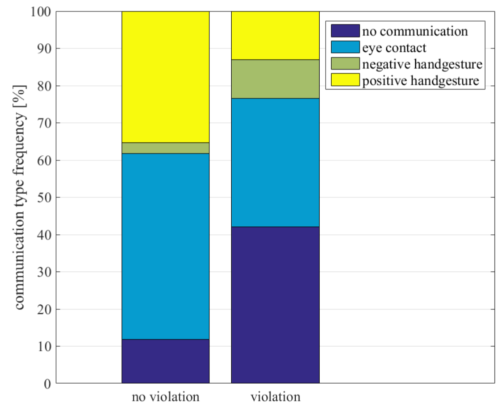

Figure 4: Communication by the VRU in case there is (no) violation of the law, as percentage of the total number of counted communication cases.

Figure 5 shows the difference in communication for vehicles in case of *(no) violation of the law*. From this figure, it seems that in case of *no violation of the law*, *driving behaviour* made up 65% of the total. In case of *violation of the law*, *driving behaviour* made up 18% of the total. In case of *no violation of the law*, *no communication* made up 32% of the total. In case of *violation of the law*, there was *no communication* in 69% of the cases.

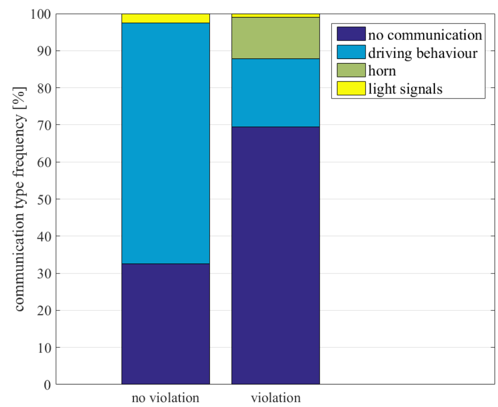

Figure 5: Communication by the vehicle towards the VRU in case there is (no) violation of the law, as percentage of the total number of counted communication cases.

4.3 The Stationary Car Scenario

As shown in Figure 6, in 52% of the cases VRUs were *clearly visible* and 24% of the cases showed a *blocked view*. In the scenario with a stationary vehicle on the adjacent road, the *blocked view* scenario occurred in 51% of the cases and the *clearly visible* outcome occurred in 22% of the cases.

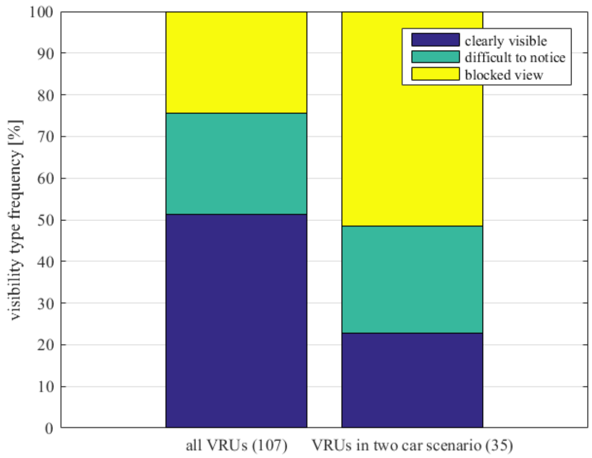

Figure 6: Visibility of VRU (left) and the visibility of the VRU in a scenario with at least two vehicles (right), as percentage of the total number of counted communication cases.

Figure 7 shows the visibility of the VRU with at least two vehicles in the scenario combined with *the severity of the conflict*. In this scenario, there is at least one stationary vehicle and at least one non-stationary vehicle. It seems that for *accident* and *near miss*, there is more *blocked view* than the other two categories.

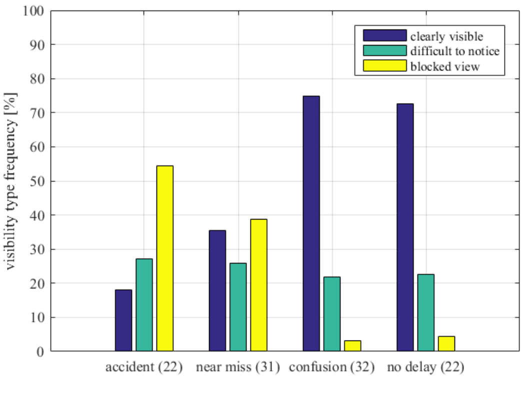

Figure 7: Visibility of VRUs plotted against the severity of the conflict by (number of VRUs).

4.4 Traffic Situation

Figure 8 shows that there was mostly *no communication* by the VRU when an accident occurred. *Eye contact* occurred substantially more at *near miss*, *confusion* and *no delay*. *Positive hand gestures* were most common at *no delay*.

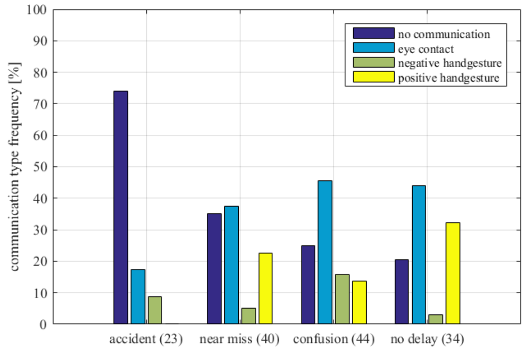

Figure 8: Communication plotted against the VRU per severity of the conflict by (number of interactions).

From Figure 9 it seems that *accidents* occurred most at *intersections*. Most *near misses* occurred at a *not defined* situation. The total communication resulted in 86% for *not defined*, while at *intersection* the total communication resulted in 65%.

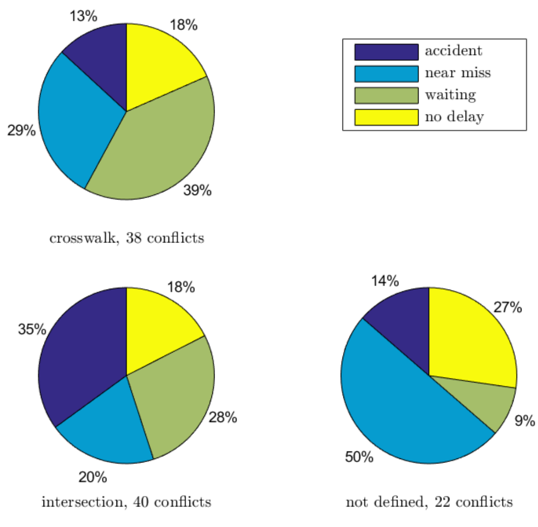

Figure 9: Severity of the conflict plotted against the three different types of traffic situations, by number of conflicts.

5 Discussion

The goal of the research was to find scenarios in which communication between a VRU and vehicle takes place. From the self made footage it can be seen that direct communication between drivers and VRUs rarely occurs in real-life situations. Therefore, the discussion only focuses on the YouTube analysis.

5.1 High Speed Scenarios

In low speed scenarios, the frequency of *eye contact* is high. From this it can be concluded that when VRUs are crossing the road they are looking for confirmation that oncoming vehicles are stopping or yielding. *No communication* occurred more in high speed scenarios compared to low and intermediate speed scenarios. Communication using a *horn or sound* was more present in high speed scenarios. From this result it can be concluded that in high speed scenarios there is either no time to communicate or only time to use a horn. It could be concluded from this that an AV trying to communicate with a VRU at low and intermediate speed scenarios could be more useful than at high speed scenarios.

5.2 Traffic Situation

Although there is more communication at lower speeds, there is no clear difference between the communication in *traffic situations*. *Accidents* happened most at *intersections*. This could be due to intersections being crowded and there is traffic coming up from multiple directions. The *not defined* traffic situation has most *near misses*, so could be pointed out as the situation where the least communication takes place by VRUs. However, since it consists of many sub-types of traffic situations (for instance straight urban road, highway, traffic jam), it is hard to conclude if there is lack of communication in all sub-types. Therefore, the situation where the least communication takes place by the VRU is said to be *intersection*.

5.3 Situation with a Stationary Vehicle

Stationary vehicles caused a blocked view in a significant number of cases. It could be concluded that in situations with a stationary vehicle and a multiple lane road, the clear view is compromised. From this it could be concluded that these blocked view situations resulted in less use of communication and caused a higher occurrence of accidents and near misses. Therefore, this situation was used in Unity as the experimental situation. This will be further described in chapter 6.

5.4 Margin of Distance and Speed Error

When looking at Figures 19, 20 and 21 in Appendix D it can be seen that there are no significant differences between the plots without error taken into account and the plot with error taken into account. The frequency of communication still declines when the speed increases. Therefore it will not be necessary to adjust the earlier made observations. It should be mentioned that the way the margin of error was taken into account still leaves some uncertainty. For the error plots all the speed measurements were either decreased or increased by 10%. This means that the patterns will stay roughly the same. When some speed measurements are lower while others are higher, the patterns may change more.

5.5 Improvements and Recommendations

Firstly, YouTube is a sensational platform, so the sample group does not fully represent the real-life situation. Users of the platform upload videos in order to gain as many views as possible. The consequence is a bigger representation of accidents and near misses compared to the real-life appearances. If there is more time available for research it is recommended to record at a location, so the sample group is a better representation of the reality.

Secondly, in this research the measured conflict time was not useful. This was measured from the moment the first VRU entered the road until the moment the VRU left the road or was hit by a vehicle. In some videos the recording had to be stopped before one of this criteria was met. These differences in time measurement resulted in faulty data. Furthermore, the conflict time did not give a representation of the time VRUs and drivers had to react to the conflict. In some cases, a VRU would stay on the road after a near miss due to possible shock. An actual response time would be hard to measure from YouTube videos since it would be hard to determine the moment a driver sees a VRU. A well measured response time could give useful information for the study but this has to be done in a different way. To improve the accuracy of the conflict time, it is advised to only use High Definition (4K) videos to extract the gaze direction data from both the VRU and the vehicle driver.

The sample size of this research was 101 videos, which is not sufficient to cover all possible traffic situations with enough videos. For example there is only one video containing a roundabout. A larger sample size would give more insight in specific traffic situations and would provide more reliable results.

The two lane scenario with one stationary car resulted in VRUs who were not visible. Further research could look into a way to warn upcoming cars for the hardly visible VRU or warn the VRU for the approaching car.

6 Proof of Concept for an Experiment in Unity

From the discussion above the certain scenario factors are defined to construct a scenario where communication may be most worthwhile. This scenario is built in a virtual reality environment using the game-engine Unity. This scenario is built using a follow-up on previous TU Delft research from L. Kooijman and K. de Clerq [11]. Our objective with Unity is to build the scenario and give a proof of concept for an experiment in Unity. An experiment should be conducted to confirm the influence of communication on the scenario to increase the safety of VRUs.

6.1 Proof of Scenario for Unity Experiment

The location of the environment is the first factor for the scenario. The country that is chosen as the location for the Unity scenario is the United Kingdom (UK), since the UK is representative for a Western-European environment and most of the YouTube videos were filmed in the UK (Figure 22 in Appendix D). To apply it to Dutch standards we changed the scenario to a right-side driving environment.

In the discussion it was concluded that high speed scenarios in most cases lack communication. Therefore, the Unity scenario takes place in an urban environment, which consists of a low and intermediate speed scenario.

In the discussion it was concluded that scenarios with a blocked view by a stationary vehicle resulted in a higher number of accidents and near misses. This scenario has a multiple lane road to allow space for a stationary vehicle and a driving vehicle on the adjacent lane. The most common scenario from the YouTube data is a two lane road, which also is the lane situation with the most communication by the VRU-Vehicle. (Figure 23 in Appendix D). Therefore, the Unity scenario takes place on a two lane road with at least one stationary vehicle and at least one driving vehicle on the adjacent lane, which is showed in Figures 10 - 12 on the next page. Appendix E shows additional specifications regarding the Unity environment. These specifications are used as a standard scenario. Further research could look into varying these parameters.

6.2 Concept for an Experiment

In a possible experiment, multiple scenarios could be tested on participants using Virtual Reality (VR) Goggles and Suit like the Oculus Rift and X-Sens. The Oculus point of view is attached to the head of a virtual puppet and the suit movement is connected to the movement of the puppet. First, the participants should be asked to get familiar with the environment. For the experiment, the participants should be asked to cross the road when a driving car is approaching. This experiment could be repeated so that all the participants cross the road once in a certain scenario. The data that could be extracted from this experiment are the physical reaction of the participant from the VR-Suit sensors and a questionnaire about the crossing experience. This experience could include a safe feeling while crossing, the confidence in crossing and general emotional state after crossing.

To provide a form of communication for the crossing participant, a screen could be applied on the front side of the first stationary vehicle. The screen could have two output options: “DON’T WALK” and a black screen (no output). The initial state of the screen is no output. The screen could be triggered to the state “DON’T WALK” if a approaching car is located at a certain distance from the back of the stationary vehicle (Appendix E, and returns to the initial state if the driving vehicle has passed. This trigger is also the trigger for the deceleration of the approaching vehicle. Symbols could also be used as outputs for the screen, but to avoid confusion about the exact meaning of the symbols we used text. Further research should look into possible benefits of the use of symbols. The scenarios could differ in form of communication and driving behaviour of the driving vehicles. Four different scenarios that could be tested are:

1. Vehicle approaches without braking and there is no communication from the stationary vehicle;
2. Vehicle slows down to 10 *kph* on approach and there is no communication from the stationary vehicle;
3. Vehicle approaches without braking and there is communication from the stationary vehicle;
4. Vehicle slows down to 10 *kph* on approach and there is communication from the stationary vehicle.

7 Conclusion

It is found that direct communication between drivers and VRUs rarely occurs in real-life situations. The scenario in which communication between VRUs and vehicles is worthwhile contains multiple vehicles where at least one of the vehicles is stopping for the VRU and blocks the view for the other vehicles on the adjacent lane. Also, in low and intermediate speed scenarios (< 50 [kph]) VRUs take more advantage of communication, because at high speed scenarios there is not sufficient time to communicate. It can be concluded that when VRUs are crossing the road, they are looking for confirmation that oncoming vehicles are stopping or yielding. These conclusions should be tested in the described experimental setting in Unity.

8 Acknowledgements

We would like to thank Joost de Winter and Jork Stapel for their help, support and motivation during these last three months, Lars Kooijman for his time to help us understand Unity and especially Felix Dreger for his unlimited time and feedback he has given us throughout our project.

Figure 10: The Unity crossing scenario as seen from the puppet, with the van sign off.

Figure 11: The scenario seen from above, with a car about to pass the crossing. The sign is set to “DON’T WALK”.

Figure 12: The scenario top down view with a passing car incoming.

References

[1] L. Vissers, S. van der Kint, I. van Schagen, and M. Hagenzieker. Safe interaction between cyclists, pedestrians and automated vehicles. *SWOV*, 2016.

[2] R. Zimmermann and R. Wettach. First step into visceral interaction with autonomous vehicles. 2017.

[3] D. Rothenbücher, J. Li, D. Sirkin, B. Mok, and W. Ju. Ghost driver: A field study investigating the interaction between pedestrians and driverless vehicles. Columbia University, 2016.

[4] M. Risto, C. Emmenegger, E. Vinkhuyzen, M. Cefkin, and J. Hollan. Human-vehicle interfaces: the power of vehicle movement gestures in human road user coordination. University of Iowa, 2017.

[5] C. Ackermann, M. Beggiato, and J. Krems. Vehicle movement and its potential as implicit communication signal for pedestrians and automated vehicles. Proceedings of the 6th Humanist Conference, 2018.

[6] R. Talbot P. Thomas, A. Morris and H. Fagerlind. Identifying the causes of road crashes in europe. 2013.

[7] M. Šucha. Road users' strategies and communication: Driver-pedestrian interaction. 2014.

[8] A. Elgrishi A. Katz, D. Zaidel. An experimental study of driver and pedestrian interaction during the crossing conflict. 1975.

[9] M. Wattenhofer, R. Wattenhofer, and Z. Zhu. The youtube social network. Sixth International AAAI Conference on Weblogs and Social Media, 2012.

[10] E. Lopes and R. Nogueira. Proposta metodológica para validação de imagens de alta resolução do google earth para a produção de mapas. January 2011.

[11] K. de Clercq. The effects of external human-machine interfaces of automated vehicles on the crossing behaviour of pedestrians. Master's thesis, Delft University of Technology, December 2017.

A Appendix - Tables

Table 2: Search terms used for YouTube.

Search words per category	
No delay and no discomfort	Pedestrian Cyclist Good interaction Communication Dashcam
Confusion and waiting	Pedestrian Cyclist Annoyed Not Seen Waiting Crossing
Agitation, near miss evasion	Pedestrian Cyclist Near miss Near accident Lucky
Angry because of an accident	Pedestrian Cyclist Accident Hit by car

B Appendix - Interface

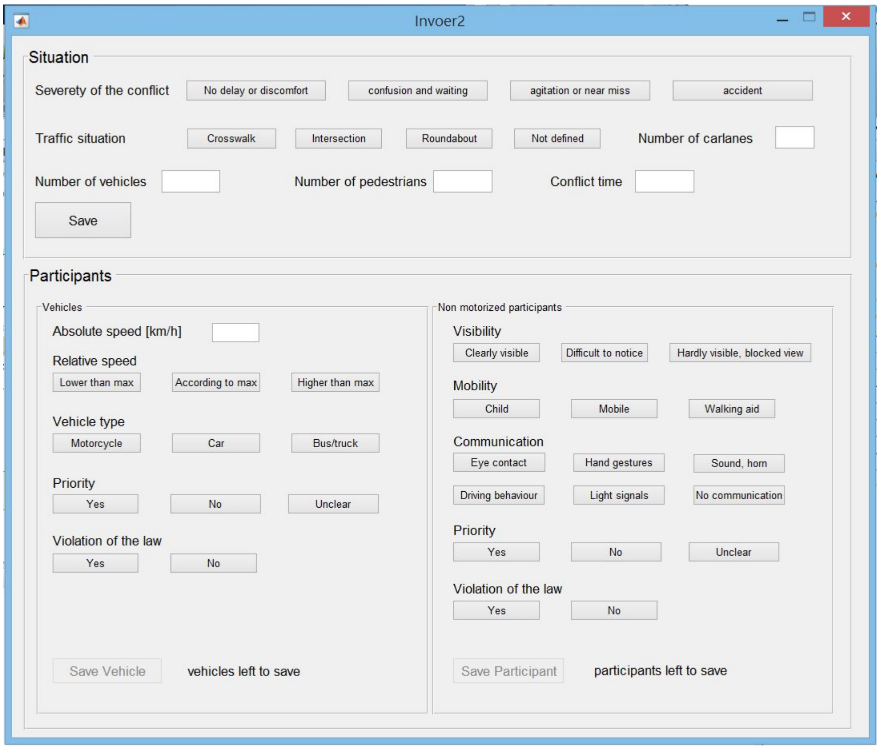

Figure 13: The used Matlab Interface.

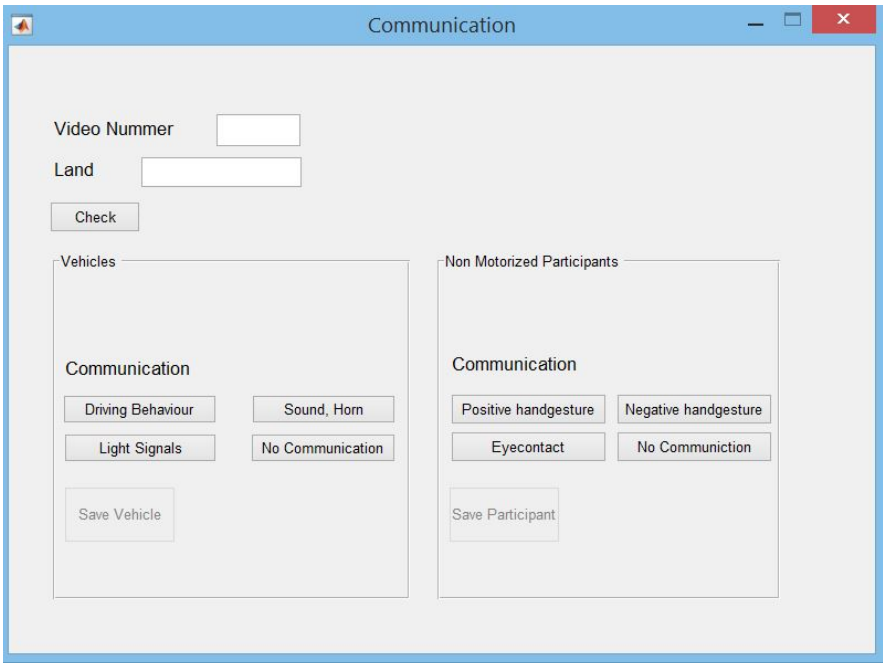

Figure 14: The used Matlab Interface.

C Appendix - YouTube measurements

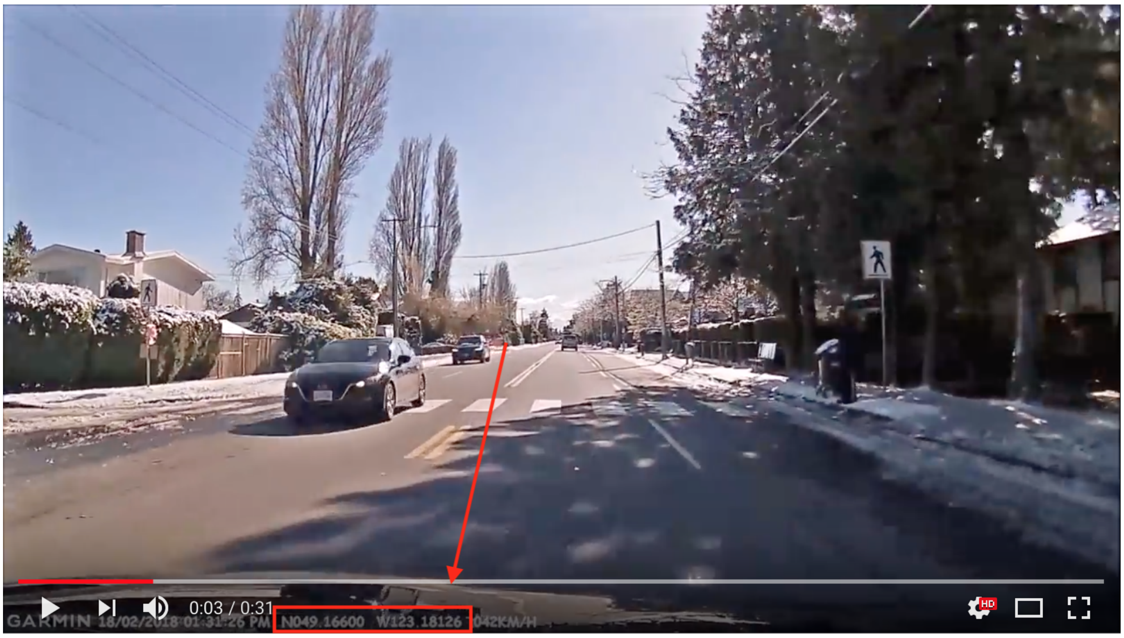

Figure 15: YouTube video with exact position.

Figure 16: YouTube video with starting point measurement (red dot).

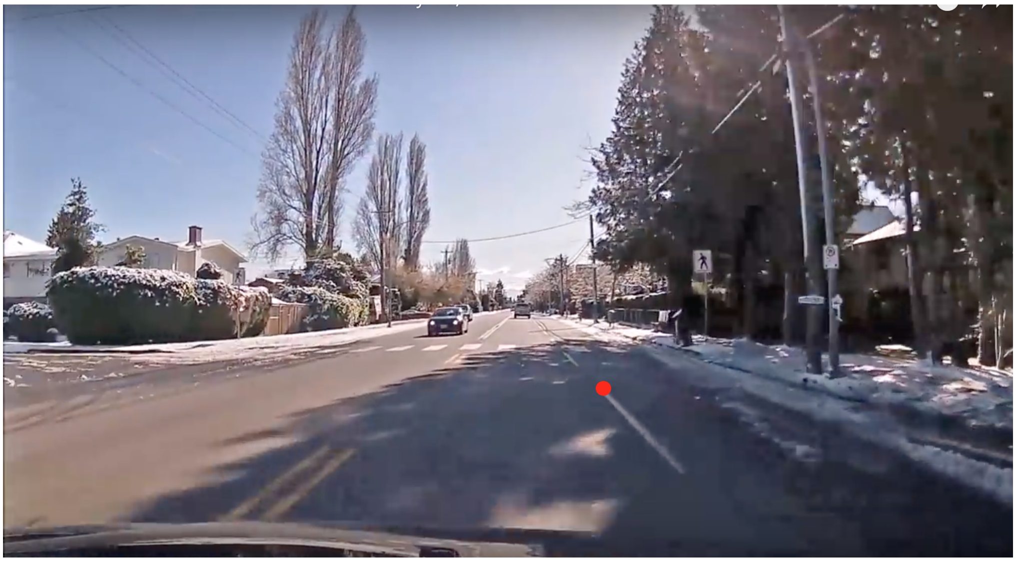

Figure 17: YouTube video with ending point measurement (red dot).

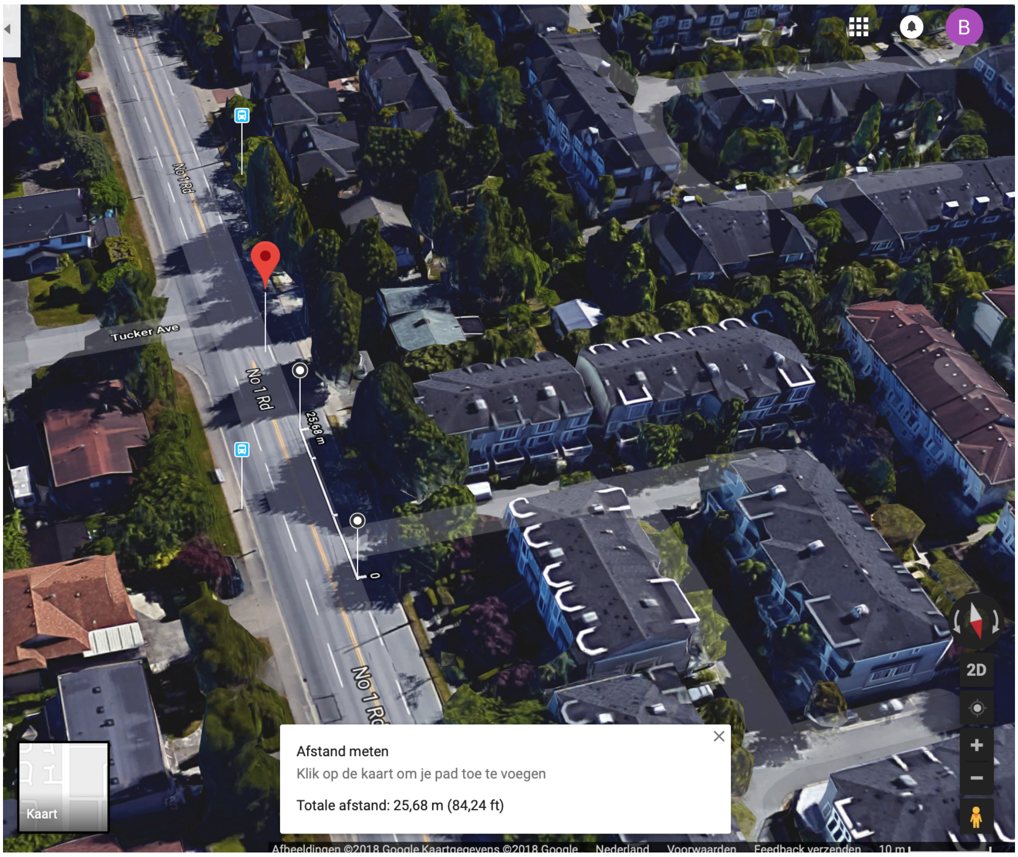

Figure 18: Measurement via Google Maps measure tool of starting and ending point.

D Appendix - Additional Graphs

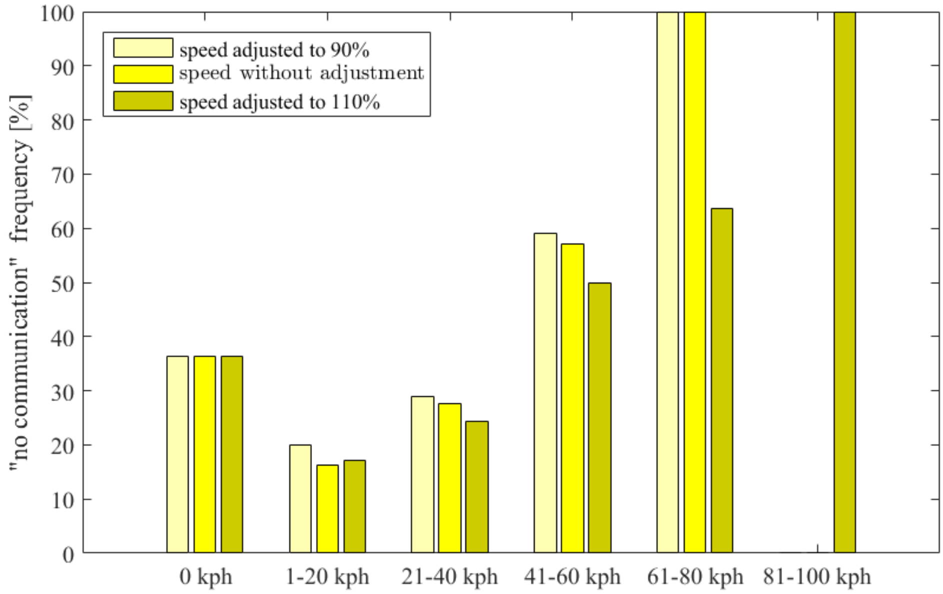

Figure 19: Occurrence of "no communication" by a VRU plotted against 20 *kph* intervals with addition of plots of the speed error margins.

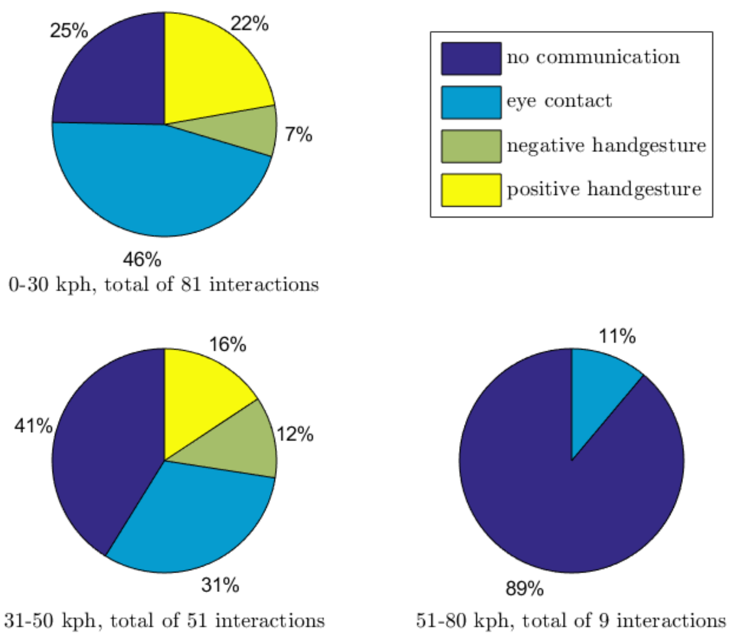

Figure 20: The different types of communication plotted for the three speed scenarios. The speed is adjusted to 90% to take the error into account.

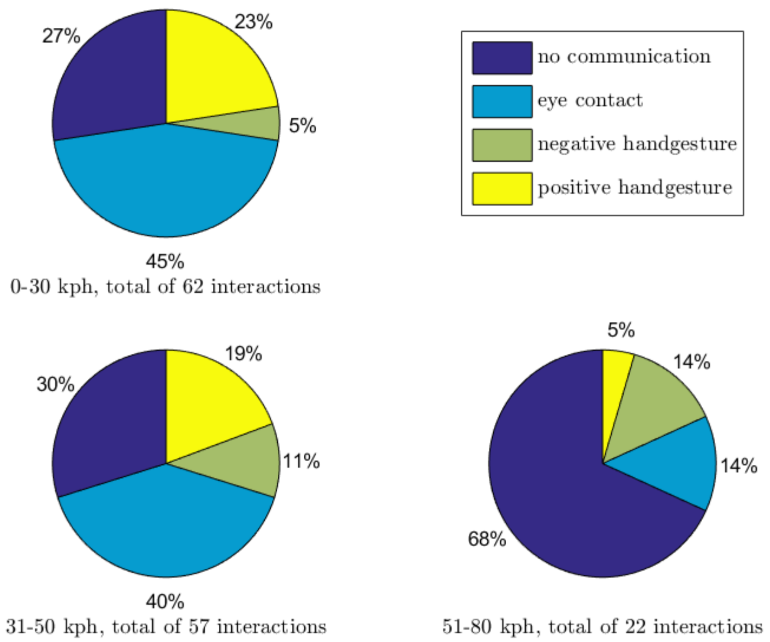

Figure 21: The different types of communication plotted for the three speed scenarios. The speed is adjusted to 110% to take the error into account.

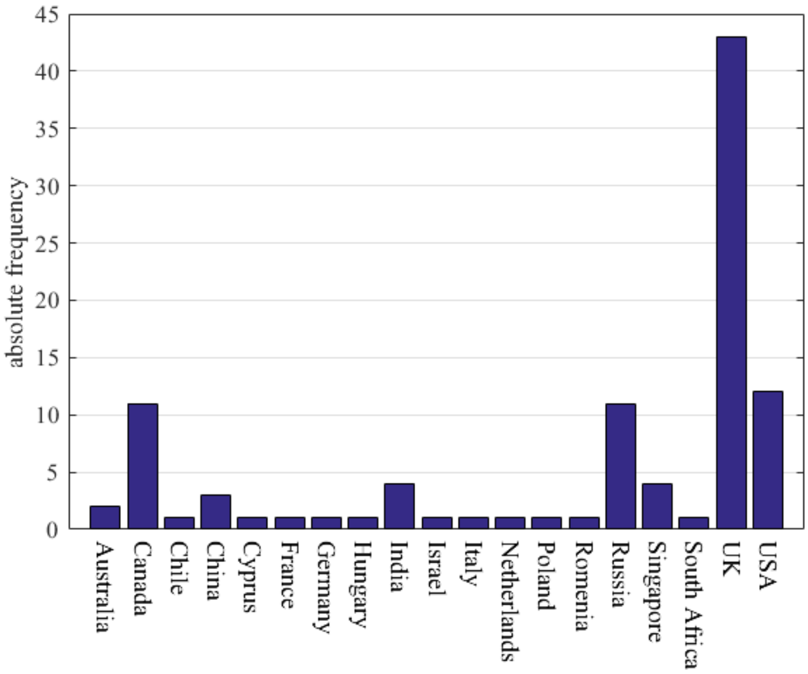

Figure 22: Countries used in YouTube video analysis.

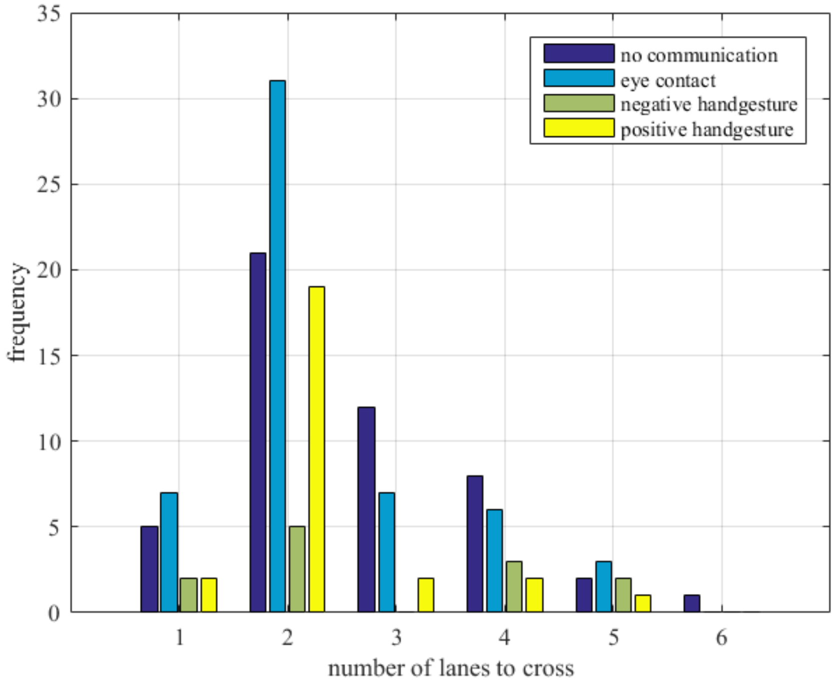

Figure 23: Communication by the VRU plotted against the number of lanes to cross. One count in the frequency means one as percentage of the total number of counted communication cases.

E Appendix - Unity

Table 3: specifications for the Unity Environment, as shown in chapter 6.

Specifications for Unity	
Stationary vehicles crosswalk	One van, 1 meter before the crosswalk. One bus, 1 meter behind the van. One van, 1 meter behind the bus.
Stationary vehicles intersection	Two cars, 1 meter from the intersection. One truck, 1 meter from the intersection. One van, 1 meter from the intersection.
Triggers	20 meters from the back of the van, triggers vehicle speed to 10 [kph]. 1 meter before the crosswalk, triggers a yielding movement of 1 meter sideways. 1 meter past the crosswalk, triggers vehicle speed to 50 [kph].
Vehicle speed	From the crosswalk to the first trigger, 50 [kph]. From the first trigger to the crosswalk, deceleration to 10 [kph]. During the passing of the crosswalk, 10 [kph]. After the passing of the crosswalk, acceleration to 50 [kph].

The environment described above is a standard environment built for this research. As stated in Chapter 6, further research could focus on using different dimensions and different speeds. The number of vehicles is a variable and can be changed to fit different types of research. The specifications above are meant for further research. This is described in our recommended experiment in Chapter 6. The specifications could be adjust to fit purpose for different research. The files can be accessed at https://bit.ly/2EINPEq.

F Appendix - Video list

Agitation, near miss or evasion (total 81 videos, 24 used)
Pedestrian Near Miss

1. https://www.youtube.com/watch?v=jRjfhL87vNg
2. https://www.youtube.com/watch?v=RbCUgPdvJb4
3. https://www.youtube.com/watch?v=8SeVc3itItI
4. https://www.youtube.com/watch?v=n1YRAT_da04
5. https://www.youtube.com/watch?v=4PpkffWGER4
6. https://www.youtube.com/watch?v=0X3K5b8QrDU
7. https://www.youtube.com/watch?v=YHLfjyHJtZA
8. https://www.youtube.com/watch?v=73Sj0AxTpFs
9. https://www.youtube.com/watch?v=_eOX7BHCsy0
10. https://www.youtube.com/watch?v=hxs3-SjrIYk
11. https://www.youtube.com/watch?v=_gOsearXKlY
12. https://www.youtube.com/watch?v=zh1dsvbpck0
13. https://www.youtube.com/watch?v=v1sbq3epLnU
14. https://www.youtube.com/watch?v=DPGbwmP3w9M
15. https://www.youtube.com/watch?v=0vaNhqCNq3g
16. https://www.youtube.com/watch?v=wBFXsUKjqfg
17. https://www.youtube.com/watch?v=cDo9_UVKhcw
18. https://www.youtube.com/watch?v=IRykUrl7s4Y
19. https://www.youtube.com/watch?v=wMDc8HRnVic
20. https://www.youtube.com/watch?v=Ao_K7NFuBGE
21. https://www.youtube.com/watch?v=_cD7aIQgPvg
22. https://www.youtube.com/watch?v=qfsvK5Vo2Uo
23. https://www.youtube.com/watch?v=8YCXeb6uN9g
24. https://www.youtube.com/watch?v=m5nLZH92UpY

Agitation, near miss or evasion (total 62 videos, 7 used)
Cyclist Near Miss crossing

25. https://www.youtube.com/watch?v=QyYcflOHHxk
26. https://www.youtube.com/watch?v=VDojW84zXXI
27. https://www.youtube.com/watch?v=pGX8WJOegwU
28. https://www.youtube.com/watch?v=Avop1Jf8RcA
29. https://www.youtube.com/watch?v=tX8rWTJXSuo
30. https://www.youtube.com/watch?v=a7Xlu7vDVfw
31. https://www.youtube.com/watch?v=KCBeSOtEEBQ

Agitation, near miss or evasion (total 42 videos, 2 used)
Pedestrian Near Miss vehicle

32. https://www.youtube.com/watch?v=kSHJDw_bIJA
33. https://www.youtube.com/watch?v=E6-zxaeGRtI

Confusion and waiting (total 28 videos, 11 used)
Crossing pedestrian waiting

34. https://www.youtube.com/watch?v=7jyr7-dW--g
35. https://www.youtube.com/watch?v=7jyr7-dW--g
36. https://www.youtube.com/watch?v=4sObXWe-RXU
37. https://www.youtube.com/watch?v=Ik6_lmzaCjA
38. https://www.youtube.com/watch?v=aET3ADHNF2Y
39. https://www.youtube.com/watch?v=n6ReKF5tbnI
40. https://www.youtube.com/watch?v=emZtVv3BXOM
41. https://www.youtube.com/watch?v=Hc3ApgLOjUY
42. https://www.youtube.com/watch?v=Ww3vDRW1M50
43. https://www.youtube.com/watch?v=VJCHeDVsApo
44. https://www.youtube.com/watch?v=l0NTyYASg6I

Confusion and waiting (total 17 videos, 4 used)
Pedestrian waiting

45. https://www.youtube.com/watch?v=0Tsy1sOf0rw
46. https://www.youtube.com/watch?v=7AOXzCs-gCg
47. https://www.youtube.com/watch?v=hL94BMQIM8g
48. https://www.youtube.com/watch?v=5VgzfFw8rg8

Confusion and waiting (total 14 videos, 3 used)
Pedestrian crossing not seen

49. https://www.youtube.com/watch?v=EMCn7mrDd0k
50. https://www.youtube.com/watch?v=_aaZl3bryMg
51. https://www.youtube.com/watch?v=dSH5-T3ZeNE

Confusion and waiting (total 8 videos, 4 used)
Annoyed pedestrian crossing

52. https://www.youtube.com/watch?v=DlEOoJGfd2Y
53. https://www.youtube.com/watch?v=d6VQCjQxVvc
54. https://www.youtube.com/watch?v=vKM1hJRCUBk
55. https://www.youtube.com/watch?v=YWx4cuazhRk

Angry because of an accident (total 43 videos, 7 used)
Pedestrian accident

56. https://www.youtube.com/watch?v=0GjwEWUOyEg
57. https://www.youtube.com/watch?v=J3XgG3aIW7A
58. https://www.youtube.com/watch?v=i6LwdZ9ZsAE
59. https://www.youtube.com/watch?v=IzWqbbTgQk4
60. https://www.youtube.com/watch?v=6wgrtlNHDwM
61. https://www.youtube.com/watch?v=a0MqtfnZpeQ
62. https://www.youtube.com/watch?v=WoqgWlA4utU

Angry because of an accident (total 52 videos, 10 used)
Pedestrian hit by car

63. https://www.youtube.com/watch?v=FydJucnb2Rk
64. https://www.youtube.com/watch?v=ZbuX6MpPcX0
65. https://www.youtube.com/watch?v=EYQEQsx-DZk
66. https://www.youtube.com/watch?v=rjetfAKG_K0
67. https://www.youtube.com/watch?v=yK-jxbYpUYw
68. https://www.youtube.com/watch?v=woet-C3icZA
69. https://www.youtube.com/watch?v=gWA7O3Uz7Lc
70. https://www.youtube.com/watch?v=YG413xzn7c0
71. https://www.youtube.com/watch?v=fVewInQ7ViE
72. https://www.youtube.com/watch?v=WzXz_8gEqnc

Angry because of an accident (total 33 videos, 2 used)
Cyclist accident

73. https://www.youtube.com/watch?v=tGIaskdYXd4
74. https://www.youtube.com/watch?v=xWSw3PHwR-o

Angry because of an accident (total 20 videos, 3 used)
Cyclist hit by car

75. https://www.youtube.com/watch?v=bs8VWWi8I7s
76. https://www.youtube.com/watch?v=8TGSWcSfK_U
77. https://www.youtube.com/watch?v=NDGhezpj5o4

No delay and no discomfort (total 35 videos, 7 used)
Communication pedestrian dashcam

78. https://www.youtube.com/watch?v=BCJ8LByIx6Y
79. https://www.youtube.com/watch?v=1EgAzdFXKK8
80. https://www.youtube.com/watch?v=OZ-ZTYF3kyA
81. https://www.youtube.com/watch?v=XpM5q1VUpKE
82. https://www.youtube.com/watch?v=Cu0sm4JDh1Q
83. https://www.youtube.com/watch?v=TKzlRA5-ycs
84. https://www.youtube.com/watch?v=BPBUd_C_dcE

No delay and no discomfort (total 7 videos, 3 used)
Pedestrian good interaction

85. https://www.youtube.com/watch?v=oFK-XzrCY3g
86. https://www.youtube.com/watch?v=oFK-XzrCY3g
87. https://www.youtube.com/watch?v=lMphb6cNScM

No delay and no discomfort (total 44 videos, 7 used)
Pedestrian road observations

88. https://www.youtube.com/watch?v=2YXivh_lxhw
89. https://www.youtube.com/watch?v=E3afai7Jtfg
90. https://www.youtube.com/watch?v=xrLnrcvGEfo
91. https://www.youtube.com/watch?v=Jd9TNMxuD5g
92. https://www.youtube.com/watch?v=YDnLKs-xPTI
93. https://www.youtube.com/watch?v=f2FoiSzbgIo
94. https://www.youtube.com/watch?v=y9tJk6Y_j2A

No delay and no discomfort (total 15 videos, 3 used)
Polite pedestrian/driver

95. https://www.youtube.com/watch?v=rnZBb7_c-w0
96. https://www.youtube.com/watch?v=PJN7D6PdIE0
97. https://www.youtube.com/watch?v=valICtuwnwQ

No delay and no discomfort (total 27 videos, 4 used)
Polite driver

98. https://www.youtube.com/watch?v=uZgmGSQ4yh0
99. https://www.youtube.com/watch?v=0y8H50DUJHs
100. https://www.youtube.com/watch?v=ZT9LPjYBRDs
101. https://www.youtube.com/watch?v=gVUhL9NaF2w

G Appendix - Matlab Scripts

```matlab
function varargout = Invoer2(varargin)
% INVOER2 MATLAB code for Invoer2.fig
%      INVOER2, by itself, creates a new INVOER2 or raises the existing
%      singleton*.
%
%      H = INVOER2 returns the handle to a new INVOER2 or the handle to
%      the existing singleton*.
%
%      INVOER2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INVOER2.M with the given input arguments.
%
%      INVOER2('Property','Value',...) creates a new INVOER2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Invoer2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Invoer2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Invoer2

% Last Modified by GUIDE v2.5 08-Nov-2018 14:21:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Invoer2_OpeningFcn, ...
                   'gui_OutputFcn',  @Invoer2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Invoer2 is made visible.
function Invoer2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Invoer2 (see VARARGIN)
```

```
% Choose default command line output for Invoer2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Invoer2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Invoer2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function numveh_Callback(hObject, eventdata, handles)
% hObject    handle to numveh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.nv = str2double(get(hObject,'String'));
handles.vl = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of numveh as text
%        str2double(get(hObject,'String')) returns contents of numveh as a double


% --- Executes during object creation, after setting all properties.
function numveh_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numveh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function numped_Callback(hObject, eventdata, handles)
% hObject    handle to numped (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.np = str2double(get(hObject,'String'));
handles.pl = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of numped as text
%        str2double(get(hObject,'String')) returns contents of numped as a double
```

```
% --- Executes during object creation, after setting all properties.
function numped_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numped (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function numlanes_Callback(hObject, eventdata, handles)
% hObject    handle to numlanes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.nlanes = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of numlanes as text
%        str2double(get(hObject,'String')) returns contents of numlanes as a double


% --- Executes during object creation, after setting all properties.
function numlanes_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numlanes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function time_Callback(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.t = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of time as text
%        str2double(get(hObject,'String')) returns contents of time as a double


% --- Executes during object creation, after setting all properties.
function time_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```matlab

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in sev1.
function sev1_Callback(hObject, eventdata, handles)
% hObject    handle to sev1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.sev2,'value',0);
    set(handles.sev3,'value',0);
    set(handles.sev4,'value',0);
    handles.sev = 1;
    guidata(hObject, handles);
end

% --- Executes on button press in sev2.
function sev2_Callback(hObject, eventdata, handles)
% hObject    handle to sev2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.sev1,'value',0);
    set(handles.sev3,'value',0);
    set(handles.sev4,'value',0);
    handles.sev = 2;
    guidata(hObject, handles);
end


% --- Executes on button press in sev3.
function sev3_Callback(hObject, eventdata, handles)
% hObject    handle to sev3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.sev1,'value',0);
    set(handles.sev2,'value',0);
    set(handles.sev4,'value',0);
    handles.sev = 3;
    guidata(hObject, handles);
end


% --- Executes on button press in sev4.
function sev4_Callback(hObject, eventdata, handles)
% hObject    handle to sev4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.sev1,'value',0);
```

```
    set(handles.sev2,'value',0);
    set(handles.sev3,'value',0);
    handles.sev = 4;
    guidata(hObject, handles);
end


% --- Executes on button press in trafsit1.
function trafsit1_Callback(hObject, eventdata, handles)
% hObject    handle to trafsit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.trafsit2,'value',0);
    set(handles.trafsit3,'value',0);
    set(handles.trafsit4,'value',0);
    handles.trafsit = 1;
    guidata(hObject, handles);
end

% Hint: get(hObject,'Value') returns toggle state of trafsit1


% --- Executes on button press in trafsit2.
function trafsit2_Callback(hObject, eventdata, handles)
% hObject    handle to trafsit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.trafsit1,'value',0);
    set(handles.trafsit3,'value',0);
    set(handles.trafsit4,'value',0);
    handles.trafsit = 2;
    guidata(hObject, handles);
end

% Hint: get(hObject,'Value') returns toggle state of trafsit2


% --- Executes on button press in trafsit3.
function trafsit3_Callback(hObject, eventdata, handles)
% hObject    handle to trafsit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.trafsit1,'value',0);
    set(handles.trafsit2,'value',0);
    set(handles.trafsit4,'value',0);
    handles.trafsit = 3;
    guidata(hObject, handles);
end

% Hint: get(hObject,'Value') returns toggle state of trafsit3


% --- Executes on button press in trafsit4.
function trafsit4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to trafsit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.trafsit1,'value',0);
    set(handles.trafsit2,'value',0);
    set(handles.trafsit3,'value',0);
    handles.trafsit = 4;
    guidata(hObject, handles);
end

% Hint: get(hObject,'Value') returns toggle state of trafsit4



function absspd_Callback(hObject, eventdata, handles)
% hObject    handle to absspd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.as = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of absspd as text
%        str2double(get(hObject,'String')) returns contents of absspd as a double


% --- Executes during object creation, after setting all properties.
function absspd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to absspd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in relspd1.
function relspd1_Callback(hObject, eventdata, handles)
% hObject    handle to relspd1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.relspd2,'value',0);
    set(handles.relspd3,'value',0);
    handles.relspd = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of relspd1


% --- Executes on button press in relspd2.
function relspd2_Callback(hObject, eventdata, handles)
% hObject    handle to relspd2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```matlab
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.relspd1,'value',0);
    set(handles.relspd3,'value',0);
    handles.relspd = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of relspd2


% --- Executes on button press in relspd3.
function relspd3_Callback(hObject, eventdata, handles)
% hObject    handle to relspd3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.relspd1,'value',0);
    set(handles.relspd2,'value',0);
    handles.relspd = 3;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of relspd3


% --- Executes on button press in veh1.
function veh1_Callback(hObject, eventdata, handles)
% hObject    handle to veh1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.veh2,'value',0);
    set(handles.veh3,'value',0);
    handles.veh = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of veh1


% --- Executes on button press in veh2.
function veh2_Callback(hObject, eventdata, handles)
% hObject    handle to veh2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.veh1,'value',0);
    set(handles.veh3,'value',0);
    handles.veh = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of veh2


% --- Executes on button press in veh3.
function veh3_Callback(hObject, eventdata, handles)
% hObject    handle to veh3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
if get(hObject,'Value') == 1
    set(handles.veh1,'value',0);
    set(handles.veh2,'value',0);
    handles.veh = 3;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of veh3


% --- Executes on button press in prio1.
function prio1_Callback(hObject, eventdata, handles)
% hObject    handle to prio1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio2,'value',0);
    set(handles.prio3,'value',0);
    handles.priov = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio1


% --- Executes on button press in prio2.
function prio2_Callback(hObject, eventdata, handles)
% hObject    handle to prio2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio1,'value',0);
    set(handles.prio3,'value',0);
    handles.priov = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio2


% --- Executes on button press in prio3.
function prio3_Callback(hObject, eventdata, handles)
% hObject    handle to prio3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio1,'value',0);
    set(handles.prio2,'value',0);
    handles.priov = 3;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio3


% --- Executes on button press in vio1.
function vio1_Callback(hObject, eventdata, handles)
% hObject    handle to vio1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
```

```
    set(handles.vio2,'value',0);
    handles.viov = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vio1


% --- Executes on button press in vio2.
function vio2_Callback(hObject, eventdata, handles)
% hObject    handle to vio2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vio1,'value',0);
    handles.viov = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vio2


% --- Executes on button press in prio11.
function prio11_Callback(hObject, eventdata, handles)
% hObject    handle to prio11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio12,'value',0);
    set(handles.prio13,'value',0);
    handles.priop = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio11


% --- Executes on button press in prio12.
function prio12_Callback(hObject, eventdata, handles)
% hObject    handle to prio12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio11,'value',0);
    set(handles.prio13,'value',0);
    handles.priop = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio12


% --- Executes on button press in prio13.
function prio13_Callback(hObject, eventdata, handles)
% hObject    handle to prio13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.prio11,'value',0);
    set(handles.prio12,'value',0);
    handles.priop = 3;
```

```
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of prio13


% --- Executes on button press in vio11.
function vio11_Callback(hObject, eventdata, handles)
% hObject    handle to vio11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vio12,'value',0);
    handles.viop = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vio11


% --- Executes on button press in vio12.
function vio12_Callback(hObject, eventdata, handles)
% hObject    handle to vio12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vio11,'value',0);
    handles.viop = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vio12


% --- Executes on button press in vis1.
function vis1_Callback(hObject, eventdata, handles)
% hObject    handle to vis1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vis2,'value',0);
    set(handles.vis3,'value',0);
    handles.vis = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vis1


% --- Executes on button press in vis2.
function vis2_Callback(hObject, eventdata, handles)
% hObject    handle to vis2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vis1,'value',0);
    set(handles.vis3,'value',0);
    handles.vis = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vis2
```

```
% --- Executes on button press in vis3.
function vis3_Callback(hObject, eventdata, handles)
% hObject    handle to vis3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.vis1,'value',0);
    set(handles.vis2,'value',0);
    handles.vis = 3;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of vis3


% --- Executes on button press in mob1.
function mob1_Callback(hObject, eventdata, handles)
% hObject    handle to mob1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.mob2,'value',0);
    set(handles.mob3,'value',0);
    handles.mob = 1;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of mob1


% --- Executes on button press in mob2.
function mob2_Callback(hObject, eventdata, handles)
% hObject    handle to mob2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.mob1,'value',0);
    set(handles.mob3,'value',0);
    handles.mob = 2;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of mob2


% --- Executes on button press in mob3.
function mob3_Callback(hObject, eventdata, handles)
% hObject    handle to mob3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(hObject,'Value') == 1
    set(handles.mob1,'value',0);
    set(handles.mob2,'value',0);
    handles.mob = 3;
    guidata(hObject, handles);
end
% Hint: get(hObject,'Value') returns toggle state of mob3
```

```
% --- Executes on button press in com1.
function com1_Callback(hObject, eventdata, handles)
% hObject    handle to com1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com1


% --- Executes on button press in com2.
function com2_Callback(hObject, eventdata, handles)
% hObject    handle to com2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com2


% --- Executes on button press in com3.
function com3_Callback(hObject, eventdata, handles)
% hObject    handle to com3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com3


% --- Executes on button press in com4.
function com4_Callback(hObject, eventdata, handles)
% hObject    handle to com4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com4


% --- Executes on button press in com5.
function com5_Callback(hObject, eventdata, handles)
% hObject    handle to com5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com5


% --- Executes on button press in com6.
function com6_Callback(hObject, eventdata, handles)
% hObject    handle to com6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of com6


% --- Executes on button press in savepar.
function savepar_Callback(hObject, eventdata, handles)
```

```
com = zeros(1,6);
com(1,1) = get(handles.com1, 'Value');
com(1,2) = get(handles.com2, 'Value');
com(1,3) = get(handles.com3, 'Value');
com(1,4) = get(handles.com4, 'Value');
com(1,5) = get(handles.com5, 'Value');
com(1,6) = get(handles.com6, 'Value');
if handles.pl > 0
    nonmotorized(handles.VidNum, handles.vis, handles.mob, handles.priop, handles.viop, com)
    resetpar(handles)
    handles.pl = handles.pl-1;
    set(handles.parleft, 'String', handles.pl);
    if handles.pl == 0
            set(handles.savepar, 'enable', 'off');
    end
    if handles.vl == 0 && handles.pl == 0
        set(handles.save, 'enable', 'on');
    end
    guidata(hObject, handles);
end

% --- Executes on button press in saveveh.
function saveveh_Callback(hObject, eventdata, handles)
if handles.vl > 0
    vehicles(handles.VidNum, handles.as, handles.relspd, handles.veh, handles.priov, handles.viov)
    resetveh(handles)
    handles.vl = handles.vl-1;
    set(handles.vehleft, 'String', handles.vl);
    if handles.vl == 0
            set(handles.saveveh, 'enable', 'off');
    end
    if handles.vl == 0 && handles.pl == 0
        set(handles.save, 'enable', 'on');
    end
    guidata(hObject, handles);
end



% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)
handles.VidNum = situations(handles.sev, handles.trafsit, handles.nlanes, handles.nv, handles.np, handles.t);
set(handles.saveveh, 'enable', 'on');
set(handles.savepar, 'enable', 'on');
set(handles.vehleft, 'String', handles.vl);
set(handles.parleft, 'String', handles.pl);
resetsit(handles)
set(handles.save, 'enable', 'off');
guidata(hObject, handles);


function resetsit(handles)
set(handles.sev1,'value',0);
set(handles.sev2,'value',0);
```

```
set(handles.sev3,'value',0);
set(handles.sev4,'value',0);
set(handles.trafsit1,'value',0);
set(handles.trafsit2,'value',0);
set(handles.trafsit3,'value',0);
set(handles.trafsit4,'value',0);
set(handles.numlanes, 'String', '');
set(handles.numveh, 'String', '');
set(handles.numped, 'String', '');
set(handles.time, 'String', '');


function resetveh(handles)
set(handles.absspd, 'String', '');
set(handles.relspd1,'value',0);
set(handles.relspd2,'value',0);
set(handles.relspd3,'value',0);
set(handles.veh1,'value',0);
set(handles.veh2,'value',0);
set(handles.veh3,'value',0);
set(handles.prio1,'value',0);
set(handles.prio2,'value',0);
set(handles.prio3,'value',0);
set(handles.vio1,'value',0);
set(handles.vio2,'value',0);


function resetpar(handles)
set(handles.vis1,'value',0);
set(handles.vis2,'value',0);
set(handles.vis3,'value',0);
set(handles.mob1,'value',0);
set(handles.mob2,'value',0);
set(handles.mob3,'value',0);
set(handles.com1,'value',0);
set(handles.com2,'value',0);
set(handles.com3,'value',0);
set(handles.com4,'value',0);
set(handles.com5,'value',0);
set(handles.com6,'value',0);
set(handles.prio11,'value',0);
set(handles.prio12,'value',0);
set(handles.prio13,'value',0);
set(handles.vio11,'value',0);
set(handles.vio12,'value',0);
```

```
%% saving the general data from the interface to the cell

function [VidNum] = situations(A, B, C, D, E, F)
input = [A, B, C, D, E, F];%storing the input
% defining the options per category
Sev = {'no delay or discomfort', 'confusion and waiting', 'agitation near miss', '
    accident'};
TrafSit = {'crosswalk', 'intersection', 'roundabout', 'Not Defined'};
NumLane = [];
NumVeh = [];
NumPed = [];
Time = [];
%adding all the options to one array
Options = {Sev, TrafSit, NumLane, NumVeh, NumPed, Time};

%% loading or creating the general datacell
%check if the cell already exists
if exist('Situations.mat') == 0;
    %create cell if it does not exist
    Situations = cell(100, 7);
    else load('Situations.mat') %load cell if it does not exist
end
%% storing video number and input A until F
%saving the video number
VidNum = find(cellfun(@isempty,Situations),1);
Situations{VidNum, 1} = VidNum;
%saving input A until F according to the earlier defined options
for i = 1:6;
    if isempty(Options{i}) == 1
        Situations{VidNum, i+1} = input(i);
    else
        Situations{VidNum, i+1} = Options{1, i}(1, input(i));

    end
end
%save the datacell
save('Situations.mat','Situations')

end
```

```matlab
%% saving the vehicle data from the interface to the cell
function vehicles(VidNum, A, B, C, D, E)
input = [A, B, C, D, E];%storing the input
% defining the options per category
AbsSpd = [];
RelSpd = {'slower than maximum','according to maximum','faster than maximum'};
Veh = {'motorcycle', 'car','bus or truck'};
Prio = {'yes', 'no', 'unclear'};
Vio = {'yes', 'no'};
%adding all the options to one array
Options = {AbsSpd, RelSpd, Veh, Prio, Vio};

%% loading or creating the vehicle datacell
%check if the cell already exists
if exist('Vehicles.mat') == 0;
    %create cell if it does not exist
    Vehicles = cell(1, 6);
    RowNum = 1;
else
    %load cell if it does not exist
    load('Vehicles.mat')
    sz = size(Vehicles);
    RowNum = sz(1) + 1; %finding the next open row
end

%% storing video number and input A until E
%saving the video number
Vehicles{RowNum, 1} = VidNum;
%saving input A until E according to the earlier defined options
for i = 1:5;
    if isempty(Options{i}) == 1
        Vehicles{RowNum, i+1} = input(i);
    else
        Vehicles{RowNum, i+1} = Options{1, i}(1, input(i));

    end
end
%save the datacell
save('Vehicles.mat','Vehicles')

end
```

```
%% saving the VRU data from the interface to the cell
function nonmotorized(VidNum,A,B,C,D,E)
input = [A, B, C, D]; %storing the input
% defining the options per category
Vis = {'clearly visible', 'difficult to notice', 'hardly visible or blocked view'};
Mob = {'child','mobile','walking aid'};
Prio = {'yes', 'no', 'unclear'};
Vio = {'yes', 'no'};
ComOptions = {'eye contatc', 'hand gestures', 'sound or horn', 'driving behaviour', 'lights signals', 'no communication'};
%adding all the options to one array
Options = {Vis, Mob, Prio, Vio};

%% loading or creating the VRU datacell
%check if the cell already exists
if exist('NonMotorized.mat') == 0;
    %create cell if it does not exist
    NonMotorized = cell(1, 6);
    RowNum = 1;
else
    %load cell if it does not exist
    load('NonMotorized.mat')
    sz = size(NonMotorized);
    RowNum = sz(1) + 1; %finding the next open row
end

%% storing video number and input A until D
%saving the video number
NonMotorized{RowNum, 1} = VidNum;
%saving input A until D according to the earlier defined options
for i = 1:4;
    NonMotorized{RowNum, i+1} = Options{1, i}(1, input(i));
end



%% Saving the communication. This is done seperately because one VRU can communicate in multiple ways
cts = find(E); %check which communication types are used and returns the indices of where the
%communication type is in ComOptions
%save the communication types to Com
for i = 1:length(cts);
    Com{1,i} = ComOptions{1,cts(i)};
end
%add Com to the datacell
NonMotorized{RowNum,6} = Com;

%save the datacell
save('NonMotorized.mat','NonMotorized')
end
```

```
function varargout = Communication(varargin)
% COMMUNICATION MATLAB code for Communication.fig
%      COMMUNICATION, by itself, creates a new COMMUNICATION or raises the existing
%      singleton*.
%
%      H = COMMUNICATION returns the handle to a new COMMUNICATION or the handle to
%      the existing singleton*.
%
%      COMMUNICATION('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in COMMUNICATION.M with the given input arguments.
%
%      COMMUNICATION('Property','Value',...) creates a new COMMUNICATION or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Communication_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Communication_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Communication

% Last Modified by GUIDE v2.5 19-Nov-2018 14:54:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Communication_OpeningFcn, ...
                   'gui_OutputFcn',  @Communication_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Communication is made visible.
function Communication_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Communication (see VARARGIN)

% Choose default command line output for Communication
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Communication wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Communication_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function VidNum_Callback(hObject, eventdata, handles)
% hObject    handle to VidNum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.VN = str2double(get(hObject,'String'));
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of VidNum as text
%        str2double(get(hObject,'String')) returns contents of VidNum as a double


% --- Executes during object creation, after setting all properties.
function VidNum_CreateFcn(hObject, eventdata, handles)
% hObject    handle to VidNum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Land_Callback(hObject, eventdata, handles)
% hObject    handle to Land (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.L = get(hObject,'String');
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of Land as text
%        str2double(get(hObject,'String')) returns contents of Land as a double


% --- Executes during object creation, after setting all properties.
function Land_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to Land (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in CP1.
function CP1_Callback(hObject, eventdata, handles)
% hObject    handle to CP1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CP1


% --- Executes on button press in CP2.
function CP2_Callback(hObject, eventdata, handles)
% hObject    handle to CP2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CP2


% --- Executes on button press in CP3.
function CP3_Callback(hObject, eventdata, handles)
% hObject    handle to CP3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CP3


% --- Executes on button press in CP4.
function CP4_Callback(hObject, eventdata, handles)
% hObject    handle to CP4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CP4


% --- Executes on button press in SaveP.
function SaveP_Callback(hObject, eventdata, handles)
% hObject    handle to SaveP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
com = zeros(1,4);
com(1,1) = get(handles.CP1, 'Value');
com(1,2) = get(handles.CP2, 'Value');
com(1,3) = get(handles.CP3, 'Value');
```

```
com(1,4) = get(handles.CP4, 'Value');

if handles.ParCount <= handles.NumP
    CS(handles.iP(handles.ParCount), 'P', com)
    set(handles.CP1,'value',0);
    set(handles.CP2,'value',0);
    set(handles.CP3,'value',0);
    set(handles.CP4,'value',0);
    handles.ParCount = handles.ParCount+1;
    set(handles.Ptext, 'String', handles.ParCount);
    if handles.ParCount > handles.NumP
            set(handles.SaveP, 'enable', 'off');
            set(handles.Ptext, 'String', '');
    end
    if handles.CarCount > handles.NumV && handles.ParCount > handles.NumP
        set(handles.Check, 'enable', 'on');
        set(handles.VidNum, 'enable', 'on');
        set(handles.VidNum, 'string', '');
    end
    guidata(hObject, handles);
end

% --- Executes on button press in CV1.
function CV1_Callback(hObject, eventdata, handles)
% hObject    handle to CV1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CV1


% --- Executes on button press in CV2.
function CV2_Callback(hObject, eventdata, handles)
% hObject    handle to CV2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CV2


% --- Executes on button press in CV3.
function CV3_Callback(hObject, eventdata, handles)
% hObject    handle to CV3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CV3


% --- Executes on button press in CV4.
function CV4_Callback(hObject, eventdata, handles)
% hObject    handle to CV4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CV4

```

```
% --- Executes on button press in SaveV.
function SaveV_Callback(hObject, eventdata, handles)
% hObject    handle to SaveV (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
com = zeros(1,4);
com(1,1) = get(handles.CV1, 'Value');
com(1,2) = get(handles.CV2, 'Value');
com(1,3) = get(handles.CV3, 'Value');
com(1,4) = get(handles.CV4, 'Value');

if handles.CarCount <= handles.NumV
    CS(handles.iV(handles.CarCount), 'V', com)
    set(handles.CV1,'value',0);
    set(handles.CV2,'value',0);
    set(handles.CV3,'value',0);
    set(handles.CV4,'value',0);
    handles.CarCount = handles.CarCount+1;
    set(handles.Vtext, 'String', handles.CarCount);
    if handles.CarCount > handles.NumV
            set(handles.SaveV, 'enable', 'off');
            set(handles.Vtext, 'String', '');
    end
    if handles.CarCount > handles.NumV && handles.ParCount > handles.NumP
        set(handles.Check, 'enable', 'on');
        set(handles.VidNum, 'enable', 'on');
        set(handles.VidNum, 'string', '');
    end
    guidata(hObject, handles);
end


% --- Executes on button press in Check.
function Check_Callback(hObject, eventdata, handles)
Result = Check(handles.VN, handles.L);
handles.NumV = cell2mat(Result(1));
handles.NumP = cell2mat(Result(2));
handles.iV = cell2mat(Result(3));
handles.iP = cell2mat(Result(4));
handles.CarCount = 1;
handles.ParCount = 1;
set(handles.SaveV, 'enable', 'on');
set(handles.SaveP, 'enable', 'on');
set(handles.Vtext, 'String', handles.CarCount);
set(handles.Ptext, 'String', handles.ParCount);
set(handles.VidNum, 'enable', 'inactive');
set(handles.Land, 'String', '');
set(handles.Check, 'enable', 'off');
guidata(hObject, handles);
% hObject    handle to Check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Save the country and return necessary info
function [Result] = Check(VidNum, Land)
%load all the data cells
load('Situations.mat');
load('Vehicles.mat');
load('NonMotorized.mat');
% save the country
Situations{VidNum,8} = Land;

%% obtain data
%find the indices in the vehicle cell with the input video number
rowV = cell2mat(Vehicles(:,1));
iV = find(rowV==VidNum);
%find the indices in the VRU cell with the input video number
rowP = cell2mat(NonMotorized(:,1));
iP = find(rowP==VidNum);
% return the number of VRUs and vehicles
NumV = length(iV);
NumP = length(iP);
%save the situations datacell
save('Situations.mat', 'Situations')
% return the data
Result = {NumV, NumP, iV, iP};

end
```

```
%% correcting the communication
function CS(index, VP, com) %obtain the index, whether it is a vehicle or VRU and the communication types
%load data cells
load('Vehicles.mat');
load('NonMotorized.mat');

%% edit the communication
if VP == 'V' % check if it is a vehicle
   ComV = {'driving behaviour', 'sound or horn', 'lights signals', 'no communication'}; %defining the options
   %fill in the communication types
   CV = find(com);
    for i = 1:length(CV);
    Com{1,i} = ComV{1,CV(i)};
    end
Vehicles{index,7} = Com; % add the communication to the datacell
end

if VP == 'P' % check if it is a VRU
   ComP = {'positive handgesture', 'negative handgesture', 'eye contact', 'no communication'}; %defining the options
   %fill in the communication types
   CP = find(com);
    for i = 1:length(CP);
    Com{1,i} = ComP{1,CP(i)};
    end
NonMotorized{index,6} = Com; % add the communication to the datacell
end

%% save the datacells
save('NonMotorized.mat','NonMotorized')
save('Vehicles.mat','Vehicles')
```

```
%% use the three different data cells to create one structure to use in data analysis
% load the three cells
load('Situations.mat');
load('Vehicles.mat');
load('NonMotorized.mat');
%define the category names
SitNames = {'PedestrianState', 'TrafficSituation', 'LanesToCross', 'NumberVehicles',
    'NumberPedestrians', 'ConflictTime', 'Country'};
VehNames = {'VideoNumber','AbsoluteSpeed', 'RelativeSpeed', 'VehicleType', 'Priority'
    , 'ViolationOfTheLaw', 'Communication'};
NMNames = {'VideoNumber','Visibility', 'Mobility', 'Priority', 'ViolationOfTheLaw', '
    Communication'};
% create structures from the cells and add the category names
S = cell2struct(Situations(:,2:8), SitNames, 2);
V = cell2struct(Vehicles, VehNames, 2);
NM = cell2struct(NonMotorized, NMNames, 2);
%% connecting the strucures
%searching for the video number in the vehicle structure and storing it
for i = 1:length(Vehicles);
    vnv(i) = V(i).VideoNumber;
end

for j = 1:length(Situations);
    indices = find(vnv==j); %searching for the video number
    %add the vehicles to the situation
    if length(indices) > 1 %check if there are multiple vehicles in the video
       for k = 1:length(indices)
           S(j).Vehicle(k) = V(indices(k));
       end
    else
        S(j).Vehicle(1) = V(indices);
    end
end

%searching for the video number in the VRU structure and storing it
for i = 1:length(NonMotorized);
    vnp(i) = NM(i).VideoNumber;
end

for j = 1:length(Situations);
    indices = find(vnp==j);%searching for the video number
    %add the VRUs to the situation
    if length(indices) > 1 %check if there are multiple VRUs in the video
       for k = 1:length(indices)
           S(j).Participant(k) = NM(indices(k));
       end
    else
        S(j).Participant(1) = NM(indices);
    end
end

%% Changing all textual data to the char type so all data is uniform and usable
for i = 1:length(S);
    S(i).PedestrianState = char(S(i).PedestrianState);
    S(i).TrafficSituation = char(S(i).TrafficSituation);
    for j = 1:length(S(i).Vehicle);
        S(i).Vehicle(j).Priority = char(S(i).Vehicle(j).Priority);
```

```
            S(i).Vehicle(j).ViolationOfTheLaw = char(S(i).Vehicle(j).ViolationOfTheLaw);
            S(i).Vehicle(j).RelativeSpeed = char(S(i).Vehicle(j).RelativeSpeed);
            S(i).Vehicle(j).VehicleType = char(S(i).Vehicle(j).VehicleType);
            S(i).Vehicle(j).Communication = char(S(i).Vehicle(j).Communication);
        end
        for k = 1:length(S(i).Participant);
            S(i).Participant(k).Priority = char(S(i).Participant(k).Priority);
            S(i).Participant(k).Visibility = char(S(i).Participant(k).Visibility);
            S(i).Participant(k).Mobility = char(S(i).Participant(k).Mobility);
            S(i).Participant(k).ViolationOfTheLaw = char(S(i).Participant(k).
                ViolationOfTheLaw);
            S(i).Participant(k).Communication = char(S(i).Participant(k).Communication);
        end
end

% saving the structure
save('MasterStruct.mat','S');
```

```
%% Boxplot with speed plotted against the different comunication types
%%loading the neccesary data
load('MasterStruct.mat') %The datastucture with all the data from the video analysis
load('MaxSpeed.mat') %vector with the highest car velocity per video
%% obtaining all the communication types of all the cars
ComV = cell(1,1);
SpeedV = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Vehicle); %per video all cars
        C = cellstr(S(i).Vehicle(j).Communication); %Obtain the communication per car
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComV{index,1} = char(C{q,1});
            SpeedV(index,1) = S(i).Vehicle(j).AbsoluteSpeed;
            index = index+1;
        end
    end
end

%% obtaining all the communication types of all the pedestrians
ComP = cell(1,1);
SpeedP = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %all VRUs
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per VRU
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComP{index,1} = char(C{q,1});
            SpeedP(index,1) = MaxSpeed(i);
            index = index+1;
        end
    end
end

%% change 'lights signals' into 'light signals' so the xlabels in the plot are right
for ii = 1:length(ComV);
    if strcmp(ComV{ii,1},'lights signals') == 1
        ComV{ii,1} = 'light signals';
    end
end

%% defining and assigning the communication types
CatVeh = {'no communication', 'sound or horn', 'driving behaviour', 'light signals'};
CatPed = {'no communication', 'eye contact', 'positive handgesture', 'negative handgesture'};
CatComV = categorical(ComV,CatVeh);
CatComP = categorical(ComP,CatPed);

%% plotting the boxplots
%plotting the boxplot for the vehicles
figure('OuterPosition', [300 150 800 700])
subplot('position', [0.1 0.2 0.35 0.7])
```

```
boxplot(SpeedV,CatComV)
ylabel('vehicle velocity [kph]')
set(gca,'XTickLabelRotation',315,'FontName', 'Times New Roman','FontSize',14)
title('Figure 1A: communication by vehicles','FontSize',12)

%plotting the boxplot for the VRUs
subplot('position', [0.53 0.2 0.35 0.7])
boxplot(SpeedP,CatComP)
set(gca,'YLim',[-2 85])
set(gca,'XTickLabelRotation',315,'FontName', 'Times New Roman','FontSize',14)
title('Figure 1B: communication by VRUs','FontSize',12)
```

```
%Occurance of "no communication" per velocity interval
%loading the neccesary data
load MasterStruct %The datastucture with all the data from the video analysis
load('MaxSpeed.mat') %vector with the highest car velocity per video
%% Obtaining all the car communications and adding the according car velocity
ComV = cell(1,1);
SpeedV = zeros(1,1);
index = 1;
for i = 1:length(S); %Look into every video
    for j = 1:length(S(i).Vehicle); %look into all the cars per video
        C = cellstr(S(i).Vehicle(j).Communication); %Obtain the communication per car
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComV{index,1} = char(C{q,1});
            SpeedV(index,1) = S(i).Vehicle(j).AbsoluteSpeed;
            index = index+1;
        end
    end
end

%% Obtaining all the pedestrian communications and adding the according car velocity
ComP = cell(1,1);
SpeedP = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %all participants
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per car
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComP{index,1} = char(C{q,1});
            SpeedP(index,1) = MaxSpeed(i);
            index = index+1;
        end
    end
end
%% Sort the communication cell and speed vector according to the speed in an ascending order
[SortSpeedV, indexx] = sortrows(SpeedV);
SortComV = ComV(indexx,:);

[SortSpeedP, indexx] = sortrows(SpeedP);
SortComP = ComP(indexx,:);

%save the sorted speed and communication vectors for the error margin plots
save('Spd&Com.mat', 'SortSpeedV', 'SortComV', 'SortSpeedP', 'SortComP')

%% splitting the communication and velocity in intervals of 20 kph
%
low = 0;
up = 20;
%finding and storing the indices of communication at 0kph
idxv(:,1) = (SortSpeedV == 0);
countv(1) = sum(idxv(:,1));
idxp(:,1) = (SortSpeedP == 0);
```

```
countp(1) = sum(idxp(:,1));
%finding and storing the indices of communication at the intervals
for x = 1:4;
    idxv(:,x+1) = (SortSpeedV>low & SortSpeedV<=up);
    countv(x+1) = sum(idxv(:,x+1)); %finding the number of interactions within the interval in order to calculate the percentages
    idxp(:,x+1) = (SortSpeedP>low & SortSpeedP<=up);
    countp(x+1) = sum(idxp(:,x+1));
    low = low+20;
    up = up+20;
end

%% Counting the number of times 'no communication' occurs and calculating the percentages
hist = zeros(5,2);
nocomV = 0;
nocomP = 0;
for y = 1:5; %all intervals
    tempV = SortComV(idxv(:,y)); %obtain vehicle communication within the speed interval
    tempP = SortComP(idxp(:,y)); %obtain VRU communication within the speed interval
    % check whether there is 'no communication' for the vehicle and count when yes
    for z = 1:countv(y);
        if strcmp(tempV{z},'no communication') == 1
            nocomV = nocomV+1;
        end
    end
    % check whether there is 'no communication' for the VRU and count when yes
    for z = 1:countp(y);
        if strcmp(tempP{z},'no communication') == 1
            nocomP = nocomP+1;
        end
    end
    %calculating percentages
    hist(y,1) = nocomV/countv(y)*100;
    hist(y,2) = nocomP/countp(y)*100;
    nocomV = 0;
    nocomP = 0;
end
%% Drawing pie charts
figure
bar(hist)
legend({'no communication by vehicles','no communication by VRUs'}, 'Location', 'northwest','FontSize',11,'Interpreter', 'latex')
ylabel('"no communication" frequency [%]','FontName', 'Times New Roman','FontSize', 12)
set(gca,'XTickLabel',{'0 kph', '1-20 kph', '21-40 kph', '41-60 kph', '61-80 kph'},'FontName', 'Times New Roman','FontSize', 12 )
```

```
%% communication types plotted against whether the traffic rules are violated
%loading the neccesary data
load('MasterStruct.mat')
%% obtaining vehicle communication and checking whether the traffic rules are violated
Com = cell(1,4);
index = 1;
indexx = 1;
VioV = zeros(101,1); %storing whether one of the vehicles in the video is violating traffic laws
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Vehicle); %per video all vehicles
        C = cellstr(S(i).Vehicle(j).Communication); %Obtain the communication per vehicle
        if strcmp(S(i).Vehicle(j).ViolationOfTheLaw, 'yes') == 1 %per vehicle checking if there is violation of traffic laws
            VioV(i) = 1;
        end
        %split multiple communication of a single car into multiple cells.
        %When nobody is violating traffic laws it is stored in collumn 1 otherwise in 2
        for q = 1:length(C);
            if strcmp(S(i).Participant(1).ViolationOfTheLaw, 'no') == 1 && strcmp(S(i).Vehicle(j).ViolationOfTheLaw, 'no') == 1
                Com{index,1} = char(C{q,1});
                index = index+1;
            else
                Com{indexx,2} = char(C{q,1});
                indexx = indexx+1;
            end
        end
    end
end

%% obtaining VRU communication and checking whether the traffic rules are violated
index = 1;
indexx = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %per video all vehicles
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per VRU
        %split multiple communication of a single car into multiple cells.
        %When nobody is violating traffic laws it is stored in collumn 3 otherwise in 4
        for q = 1:length(C);
            if strcmp(S(i).Participant(j).ViolationOfTheLaw, 'no') == 1 && VioV(i) == 0
                Com{index,3} = char(C{q,1});
                index = index+1;
            else
                Com{indexx,4} = char(C{q,1});
                indexx = indexx+1;
            end
        end
    end
end
```

```
%% counting communication types and prepare matrices for the bar graphs
hist = zeros(4,4);
% define communication types for comparison
CV = {'no communication', 'driving behaviour', 'sound or horn', 'lights signals'};
CP = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};

% check which communication type was used by the vehicles and counting them
for x = 1:2;
    for k = 1:length(Com);
        if isempty(Com{k,x}) == 0
            y = find(ismember(CV,Com(k,x))); %comparing to the definition arrays
            hist(x,y) = hist(x,y)+1; %counting
        else
            break %stopping when all communication types are done (not all collumns are the same length)
        end
    end
end
% check which communication type was used by the VRUs and counting them
for x = 3:4;
    for k = 1:length(Com);
        if isempty(Com{k,x}) == 0
            y = find(ismember(CP,Com(k,x)));
            hist(x,y) = hist(x,y)+1;
        else
            break
        end
    end
end

%calculating percentages
for ii = 1:4;
    hist(ii,:) = hist(ii,:)./sum(hist(ii,:)).*100;
end

%% plotting the bar graphs
figure
bar(hist(1:2,:),'stacked')
legend({'no communication', 'driving behaviour', 'horn', 'light signals'},'FontName', 'Times New Roman','FontSize',16)
set(gca,'XTickLabel',{'no violation','violation'},'FontName', 'Times New Roman','FontSize',15)
ylabel('communication type frequency [%]','FontSize',15)
axis([0 4 0 100])
grid on

figure
bar(hist(3:4,:),'stacked')
legend({'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'},'FontName', 'Times New Roman','FontSize',15)
set(gca,'XTickLabel',{'no violation','violation'},'FontName', 'Times New Roman','FontSize',15)
ylabel('communication type frequency [%]','FontSize',16)
axis([0 4 0 100])
grid on
```

```
%% visibility types plotted for every scenario and scenarios with a stationary car
load('MasterStruct.mat')
%% Checking if there is a stationary second car and if yes storing visibility type
Vis = cell(1,1);
index = 1;
for i = 1:length(S); %all videos
    if length(S(i).Vehicle) > 1 %checking if there are more than one car
        for j = 1:length(S(i).Vehicle) %per video all cars
            %checking if the car is stationary or stopping
            if S(i).Vehicle(j).AbsoluteSpeed == 0 || strcmp(S(i).Vehicle(j).Communication, 'driving behaviour') == 1
                %saving visibility of all VRUs in the video
                for q = 1:length(S(i).Participant)
                    Vis(index,1) = cellstr(S(i).Participant(q).Visibility);
                    index = index+1;
                end
                break %when a stationary car is found move to the next video so VRUs will not be saved twice
            end
        end
    end
end

%% storing visibility of all VRUs
indexx = 1;
for ii = 1:length(S); %all videos
    %storing the visibility of all VRUs in all videos
    for jj = 1:length(S(ii).Participant)
        VisTot(indexx,1) = cellstr(S(ii).Participant(jj).Visibility);
        indexx = indexx+1;
    end
end

%% counting visibility types
hist = zeros(2,3); %row 1 for all VRUs and row 2 for VRUs in scenarios with a second stationary car
V = {'clearly visible','difficult to notice','hardly visible or blocked view'};
for k = 1:length(Vis);
    y = find(ismember(V,Vis(k)));
    hist(2,y) = hist(2,y)+1;
end

for k = 1:length(VisTot);
    yy = find(ismember(V,VisTot(k)));
    hist(1,yy) = hist(1,yy)+1;
end

%calculate the percentages
hist(2,:) = hist(2,:)./35.*100;
hist(1,:) = hist(1,:)./107.*100;

%%
%plotting two stacked bars
figure
bar(hist,'stacked')
set(gca,'XTickLabel',{'all VRUs (107)','VRUs in two car scenario (35)'},'FontName', 'Times New Roman')
```

```
legend({'clearly visible','difficult to notice','blocked view'})
ylabel('visibility type frequency [%]')
axis([0 3 0 100])
grid on
```

```
%% Visibility of the VRU per severity of the conflict
%loading the neccesary data
load('MasterStruct.mat')
%% Obtaining conflict severity and visibility of the VRU
Vis = cell(107,1);
State = cell(107,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %all VRUs
        Vis{index,1} = char(S(i).Participant(j).Visibility); %obtaining visibility
        State{index,1} = char(S(i).PedestrianState); %obtaining conflict severity
        index = index+1;
    end
end

%% counting visibility type and sorting according to conflict severity
hist = zeros(4,3);
St = {'accident','agitation near miss','confusion and waiting','no delay or
    discomfort'};
Vi = {'clearly visible','difficult to notice','hardly visible or blocked view'};
for k = 1:107;
    x = find(ismember(St,State(k))); %checking which conflict severity
    y = find(ismember(Vi,Vis(k))); %checking which visibility type
    hist(x,y) = hist(x,y)+1; %counting one according to conflict and visibility
end

%calculating percentages
for ii = 1:4;
    perc(ii,:) = hist(ii,:)./sum(hist(ii,:)).*100;
end

%% drawing the bar graph
figure
bar(perc)
legend('clearly visible','difficult to notice','blocked view')
set(gca,'XTickLabel',{'accident (22)','near miss (31)','confusion (32)','no delay
    (22)'}, 'FontName', 'Times New Roman')
ylabel('visibility type frequency [%]')
axis([0 5 0 100])
grid on
```

```
%% Communication of the VRU plotted against the conflict severity
%loading the neccesary data
load('MasterStruct.mat')
%% obatining VRU communication and conflict severity
Com = cell(1,1);
State = cell(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %per video all VRUs
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per VRU
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            Com{index,1} = char(C{q,1}); %storing communication
            State{index,1} = char(S(i).PedestrianState); %storing conflict severity
            index = index+1;
        end
    end
end

%% Counting communication types and sorting according to conflict severity
hist = zeros(4,4);
%arrays for comparison
St = {'accident','agitation near miss','confusion and waiting','no delay or discomfort'};
Cc = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};
% counting in the according row and collumn
for k = 1:length(State);
    x = find(ismember(St,State(k)));
    y = find(ismember(Cc,Com(k)));
    hist(x,y) = hist(x,y)+1;
end
%calculating percentages
for ii = 1:4;
    perc(ii,:) = hist(ii,:)./sum(hist(ii,:)).*100;
end

%% drawing the bar graph
figure
bar(perc)
legend('no communication', 'eye contact', 'negative handgesture', 'positive handgesture')
set(gca,'XTickLabel',{'accident (23)','near miss (40)','confusion (44)','no delay (34)'}, 'FontName', 'Times New Roman')
ylabel('communication type frequency [%]')
axis([0 5 0 100])
grid on
```

```matlab
%% Severity of the conflict per traffic situation
%loading the neccesary data
load('MasterStruct.mat')

%% obtaining state and conflict severity
State = cell(101,1);
Traff = cell(101,1);
index = 1;
for i = 1:length(S); %all videos
        State{i,1} = char(S(i).PedestrianState);
        Traff{i,1} = char(S(i).TrafficSituation);
end
%% counting the conflict severities and sorting it according to traffic situation
perc = zeros(4,4);
% arrays for comparison
St = {'accident','agitation near miss','confusion and waiting','no delay or discomfort'};
Tr = {'crosswalk', 'intersection','Not Defined','roundabout'};
%comparing and counting in the according row and collumn
for n = 1:101;
    x = find(ismember(Tr,Traff(n))); %comparing traffic situation to the array above
    y = find(ismember(St,State(n))); %comparing conflict severity to the array above
    perc(x,y)= perc(x,y)+1;
end
%% plotting the charts
labels = {'accident', 'near miss', 'waiting', 'no delay'};
figure
subplot('Position', [0 0.6 0.4 0.4]) %plot the crosswalk pie chart
pie(perc(1,:))
title({'crosswalk, 38 conflicts'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
legend(labels,'Units','normalized','FontSize',11,'Position', [0.56 0.75 0.3 0.2],'Interpreter', 'latex')

subplot('Position', [0 0.1 0.4 0.4]) %plot the intersection pie chart
pie(perc(2,:))
title({'intersection, 40 conflicts'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')


subplot('Position', [0.5 0.1 0.4 0.4]) %plot the not defined pie chart
pie(perc(3,:))
title({'not defined, 22 conflicts'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
```

```
%% Fault margin on "no communication" vs velocity
load('Spd&Com.mat')
%% adjusting the speed vector to the errors
SSPMin = 0.9.*SortSpeedP;
SSPMax = 1.1.*SortSpeedP;
%% splitting the communication and velocity in intervals of 20 kph
%
low = 0;
up = 20;
%finding and storing the indices of communication at 0kph
%negative error
idxmin(:,1) = (SSPMin == 0);
countmin(1) = sum(idxmin(:,1));
%without error
idxp(:,1) = (SortSpeedP == 0);
countp(1) = sum(idxp(:,1));
%positive error
idxmax(:,1) = (SSPMax == 0);
countmax(1) = sum(idxmax(:,1));
%finding and storing the indices of communication at the intervals
for x = 1:5;
    %negative error
    idxmin(:,x+1) = (SSPMin>low & SSPMin<=up);
    countmin(x+1) = sum(idxmin(:,x+1));
    %without error
    idxp(:,x+1) = (SortSpeedP>low & SortSpeedP<=up);
    countp(x+1) = sum(idxp(:,x+1));
    %positive error
    idxmax(:,x+1) = (SSPMax>low & SSPMax<=up);
    countmax(x+1) = sum(idxmax(:,x+1));
    %increase the boundaries for the next interval
    low = low+20;
    up = up+20;
end

%% Counting the number of times 'no communication' occurs and calculating the
    percentages
hist = zeros(6,3);
nocommin = 0;
nocomP = 0;
nocommax = 0;
for y = 1:6; %all intervals
    %obtain VRU communication within the speed interval
    tempmin = SortComP(idxmin(:,y));
    tempP = SortComP(idxp(:,y));
    tempmax = SortComP(idxmax(:,y));
    % check whether there is 'no communication' for the VRU and count when
    % yes for the negative error
    for z = 1:countmin(y);
        if strcmp(tempmin{z},'no communication') == 1
            nocommin = nocommin+1;
        end
    end
    % check whether there is 'no communication' for the VRU and count when
    % yes for without error
    for z = 1:countp(y);
        if strcmp(tempP{z},'no communication') == 1
```

```
            nocomP = nocomP+1;
        end
    end
    % check whether there is 'no communication' for the VRU and count when
    % yes for the negative error
    for z = 1:countmax(y);
        if strcmp(tempmax{z},'no communication') == 1
            nocommax = nocommax+1;
        end
    end
    %calculating percentages
    hist(y,1) = nocommin/countmin(y)*100;
    hist(y,2) = nocomP/countp(y)*100;
    hist(y,3) = nocommax/countmax(y)*100;
    nocommin = 0;
    nocomP = 0;
    nocommax = 0;
end

%% plotting the graph
figure
b = bar(hist);
b(1).FaceColor = [1 1 0.7];
b(2).FaceColor = [1 1 0];
b(3).FaceColor = [0.8 0.8 0];
legend({'speed adjusted to 90%','speed without adjustment','speed adjusted to 110%'}, 'Location', 'northwest','FontSize',11,'Interpreter', 'latex')
ylabel('"no communication" frequency [%]','FontName', 'Times New Roman','FontSize', 12)
set(gca,'XTickLabel',{'0 kph', '1-20 kph', '21-40 kph', '41-60 kph', '61-80 kph', '81-100 kph'},'FontName', 'Times New Roman','FontSize', 12 )
```

```matlab
%% Communication per speed interval (0-30, 31-50, 51-80)
%loading the neccesary data
load MasterStruct
load('MaxSpeed.mat')
%% obtaining all the communication types of all the cars
ComV = cell(1,1);
SpeedV = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Vehicle); %per video all cars
        C = cellstr(S(i).Vehicle(j).Communication); %Obtain the communication per car
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComV{index,1} = char(C{q,1});
            SpeedV(index,1) = S(i).Vehicle(j).AbsoluteSpeed;
            index = index+1;
        end
    end
end

%% obtaining all the communication types of all the pedestrians
ComP = cell(1,1);
SpeedP = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %per video all VRUs
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per car
        %split multiple communication of a single car into multiple cells and add the according velocity to the speed vector
        for q = 1:length(C);
            ComP{index,1} = char(C{q,1});
            SpeedP(index,1) = MaxSpeed(i);
            index = index+1;
        end
    end
end
%% Sort the communication cell and speed vector according to the speed in an ascending order
[SortSpeedV, indexx] = sortrows(SpeedV);
SortComV = ComV(indexx,:);

[SortSpeedP, indexx] = sortrows(SpeedP);
SortComP = ComP(indexx,:);

%edit the speed vectors for the error plots
SSpMax = 1.1.*SortSpeedP;
SSpMin = 0.9.*SortSpeedP;

%% counting the frequency of the communication types and storing them so pie charts can be plotted
%defining the communication types so the communication cell can be compared
CcV = {'no communication', 'sound or horn', 'driving behaviour', 'lights signals'};
CcP = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};

```

```
%creating the pie chart matrix for the plot
pieP = zeros(3,4);
for k = 1:75; %low speed interval (0-30 kph)
    y = find(ismember(CcP,SortComP(k))); %check which communication type occured and storing it accordingly
    pieP(1,y) = pieP(1,y)+1; %counting
end
for k = 76:125; %medium speed interval (31-50 kph)
    y = find(ismember(CcP,SortComP(k)));
    pieP(2,y) = pieP(2,y)+1;
end
for k = 126:141; %high speed interval (51-80 kph)
    y = find(ismember(CcP,SortComP(k)));
    pieP(3,y) = pieP(3,y)+1;
end
%creating the pie chart matrix for the plot with the 110% error
piePMax = zeros(3,4);
for k = 1:62; %low speed interval (0-30 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMax(1,y) = piePMax(1,y)+1;
end
for k = 63:119; %medium speed interval (31-50 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMax(2,y) = piePMax(2,y)+1;
end
for k = 120:141; %high speed interval (51-80 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMax(3,y) = piePMax(3,y)+1;
end
%creating the pie chart matrix for the plot with the 90% error
piePMin = zeros(3,4);
for k = 1:81; %low speed interval (0-30 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMin(1,y) = piePMin(1,y)+1;
end
for k = 82:132; %medium speed interval (31-50 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMin(2,y) = piePMin(2,y)+1;
end
for k = 133:141; %high speed interval (51-80 kph)
    y = find(ismember(CcP,SortComP(k)));
    piePMin(3,y) = piePMin(3,y)+1;
end

%% plotting the pie chart without error
labels = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};
figure
subplot('Position', [0 0.6 0.4 0.4]) %plot with low speed interval
pie(pieP(1,:))
title({'0-30 kph, total of 75 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')

subplot('Position', [0 0.1 0.4 0.4]) %plot with medium speed interval
pie(pieP(2,:))
title({'31-50 kph, total of 50 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
```

```
subplot('Position', [0.5 0.1 0.4 0.4]) %plot with high speed interval
pie(pieP(3,:))
title({'51-80 kph, total of 16 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
legend(labels,'Units','normalized','FontSize',11,'Position', [0.55 0.65 0.3 0.3],'Interpreter', 'latex')


%%plotting the pie chart with 110% error
labels = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};
figure
subplot('Position', [0 0.6 0.4 0.4]) %plot with low speed interval
pie(piePMax(1,:))
title({'0-30 kph, total of 62 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')

subplot('Position', [0 0.1 0.4 0.4]) %plot with medium speed interval
pie(piePMax(2,:))
title({'31-50 kph, total of 57 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')


subplot('Position', [0.5 0.1 0.4 0.4]) %plot with high speed interval
pie(piePMax(3,:))
title({'51-80 kph, total of 22 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
legend(labels,'Units','normalized','FontSize',11,'Position', [0.55 0.65 0.3 0.3],'Interpreter', 'latex')

%% plotting the pie chart with 90% error
labels = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};
figure
subplot('Position', [0 0.6 0.4 0.4]) %plot with low speed interval
pie(piePMin(1,:))
title({'0-30 kph, total of 81 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')

subplot('Position', [0 0.1 0.4 0.4]) %plot with medium speed interval
pie(piePMin(2,:))
title({'31-50 kph, total of 51 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
legend(labels,'Units','normalized','FontSize',11,'Position', [0.55 0.65 0.3 0.3],'Interpreter', 'latex')

subplot('Position', [0.5 0.1 0.4 0.4]) %plot with high speed interval
pm = pie([8 1]);
title({'51-80 kph, total of 9 interactions'},'Units','normalized','Position', [0.5 -0.15 0],'Interpreter', 'latex')
colormap=[0.024 0.612 0.812];
pm(3).FaceColor = colormap;
```

```
%% Bargraph of frequency countries occured in the videos
%%loading the neccesary data
load('MasterStruct.mat')
%%
%Obtaining al the countries from the data
Country = cell(101,1);
for i = 1:101 %all videos
    Country(i,1) = cellstr(S(i).Country); %per video all the countries
end
%%
% counting the frequency of the occurance of the countries
[U,~,X] = unique(Country);
cnt = histc(X,1:numel(U));
%plotting the frequency of occuarnce against the countries
subplot(1,1,1,'Position', [0.15 0.2 0.75 0.7])
bar(cnt)
set(gca,'XTick',[],'FontName', 'Times New Roman')
cellfun(@(x,s)text(x,-1,s,'Rotation',270,'FontName', 'Times New Roman','FontSize',11)
    ,num2cell(1:numel(U)),U.')
ylabel('absolute frequency','FontName', 'Times New Roman','FontSize',12)
grid on
```

```
%Communication by the pedestrian per number of lanes
%%loading the neccesary data
load Masterstruct
%% Obtaining all the communication types of all the VRUs

Com = cell(1,1);
Lanes = zeros(1,1);
index = 1;
for i = 1:length(S); %all videos
    for j = 1:length(S(i).Participant); %all VRUs
        C = cellstr(S(i).Participant(j).Communication); %Obtain the communication per VRU
        %split multiple communication of a single car into multiple cells and add the according number of lanes to the speed vector
        for q = 1:length(C);
            Com{index,1} = char(C{q,1});
            Lanes(index,1) = (S(i).LanesToCross); %obtaining the number of lanes
            index = index+1;
        end
    end
end

%% Counting the number of times the certain number of lanes occurs
hist = zeros(6,4);
La = (1:6);
Cc = {'no communication', 'eye contact', 'negative handgesture', 'positive handgesture'};
for k = 1:length(Lanes);
    x = find(ismember(La,Lanes(k)));
    y = find(ismember(Cc,Com(k)));
    hist(x,y) = hist(x,y)+1;
end

%% Drawing the bargraphs
figure
bar(hist)
legend('no communication', 'eye contact', 'negative handgesture', 'positive handgesture')
set(gca,'XTickLabel',{'1','2','3','4','5','6'}, 'FontName','Times New Roman')
ylabel('frequency')
xlabel('number of lanes to cross')
axis([0 7 0 35])
grid on
```