A Method for Determining How Much Time People Need to Understand Traffic Situations

TU Delft
X.A. Coster

A Method for Determining How Much Time People Need to Understand Traffic Situations

By

X.A. Coster

MSc Thesis

Master of Science
in Mechanical Engineering

at the Delft University of Technology
to be defended publicly on November 3rd, 2015 at 14:00 h.

Supervisors:	Dr.ir. J.C.F. de Winter, TU Delft
	Z. Lu, MSc, TU Delft
Thesis Committee:	Dr. ir. D.A. Abbink, TU Delft
	Dr. ir. C. Borst, TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Acknowledgements

This thesis would not have been possible without the help and support of several people.

First and foremost, I would like to thank Dr. ir. Joost de Winter for supervising me during my graduation process. He provided me with feedback on a regular basis, and helped me get the thesis up to par. I could not have written this thesis without him.

Dr. ir. Riender Happee deserves my gratitude for introducing me to automated driving, which flamed my enthusiasm for the topic, for helping me get started on my graduation work, and for arranging that Joost became my supervisor.

It is my pleasure to thank MSc. Zhenji Lu for providing feedback on my methods, helping me setup the experiment, and for helping me find answers to some general questions.

Last but not least, very special thanks go to my wife, Charlotte, for supporting me all the way, and for correcting my English writing, which enhanced the readability of this thesis a great deal.

Index

Thesis Report

A Method for Determining How Much Time People Need to Understand Traffic Situations

Abstract

It is expected that in the coming decade, cars with an automated driving mode will become available. Most likely, this automated mode will only be used in specific situations and/or on specific roads, which means that the driver/automation needs to switch between the automated mode and manual control. Such switches are termed 'transitions of control' (ToCs). How much time humans need to assess a traffic situation (i.e., to gain situation awareness, SA) while preparing for ToCs is still largely unknown. This study has been designed to assess whether the method used by Gugerty (1997), where particpants were asked to reproduce a traffic scenario in a top-down view, is suited for finding this preparation time by assessing its capabilities for finding effects of preparation time, and traffic density on reproduction performance.

Participants were asked to reproduce the positions (i.e., distance & lane) and relative speeds of surrounding traffic in a top-down view after viewing each of 24 videos of simulated traffic scenarios on a three-lane highway, shown on a computer monitor. These scenarios had six different durations (1, 3, 7, 9, 12, and 20 seconds), which were participants' preparation times, as well as two different levels of traffic density (4 vs. 6 surrounding cars). The scenarios were shown to participants in a randomized order.

Results of the experiment showed that video length had a positive, and traffic density a negative effect on reproduction performance. The effects of video length on reproduction performance, when comparing scores of videos of 20 s to those of 1 s for combined traffic density, was strongest in the *Average absolute error distance of hits and misses*. Participants agreed more with having enough time to prepare for the task, and rated the reproduction task as 'less difficult' with longer videos, and with lower traffic density. The effect of traffic density on reproduction performance, averaged over all video lengths, was strongest in the *hit percentage* score. Although both video length and traffic density had clear effects on both self-reported ratings, the effect of video length was stronger in the self-reported time rating, and the effect of traffic density was stronger in the difficulty rating.

The effect of increasing video length seemed to diminish for the reproduction of positions and number of surrounding cars. The diminishing of the effect could indicate that there is a saturation level of SA. With a lower traffic density the diminishing effect was stronger, which could indicate that a saturation level of SA is reached earlier when less traffic needs to be assessed. The diminishing effect was absent in the reproduction scores for speeds, which indicates that accurately reproducing the (relative) speeds of these amounts of surrounding traffic might take more time than the maximum of 20 s of this experiment.

Although the used desktop-based method may lack the realism needed for direct application of the results, it seems suited for studying the preparation time if implemented in more immersive experimental designs.

Introduction

Over the past few decades, an increasing number of automated driving systems have become available, for both research and consumer purposes. Various manufacturers have brought cars onto the market that are equipped with a partially automated mode [1], which can take the form of a traffic jam assistant [2] where the driver still needs to monitor the system and his/her surroundings. By 2020, partially automated modes will probably have become available for a wider variety of traffic scenarios. Highly automated driving systems [1], during which the driver can opt to be 'out-of-the-loop', are expected by 2025 [3].

Even though some of the technical requirements for fully automated cars are already met (e.g., the Google Driverless Car [4]), highly automated driving is not yet commercially viable in terms of technology, costs, aesthetics, and legislation [4 5]. Regarding technical capabilities, vehicles with automated modes are not yet able to cope with all traffic scenarios. Thus, depending on the situation, control has to be handed over between the driver and the automation in so-called transitions of control (ToCs).

During an automation-to-human ToC, the human driver is requested to take over control of a moving vehicle, while (s)he may have been out-of-the-loop for an extended period of time [6 7]. While taking over, the driver has to go through three partially overlapping steps. Depending on what the 'driver' was doing before, (s)he needs to

1) prepare (e.g., put away a book, and/or put the seat in the right position).
2) assess the traffic situation and accordingly build up situation awareness (SA) [8]
3) physically take over control of the vehicle (i.e., grab the steering wheel, put feet on the appropriate pedals).

The amount of time needed for each of the three steps mentioned above is still largely unknown. Damböck (2013) [9] found basic values of times needed for steps 1) and 3), with 8 seconds for a complete ToC yielding more errors than a manual baseline condition, although the difference was not statistically significant. Following up on Damböck, Gold et al. (2013) [10] discovered that shorter available times lead to faster reactions of a lower quality. The effects of automation, compared to the manual condition, were still observable with an available time of 7 seconds. Petermann-Stock et al. (2013) [11] observed the time from an audible signal to actual take-over to be 8.8 seconds at most, with an average of 3.2 seconds. Beukel and van der Voort (2013) [12] noted a decrease in the number of accidents, as well as higher self-rated SA [13] scores, when more time was available in time-critical situations. Their scores, however, were not compared with a manual driving condition. Several more recent studies have confirmed that take-over time is an important design parameter for automated driving systems [14 15 16].

Situation awareness in car driving has previously been investigated by Gugerty (1997) [17]. In his research, participants were requested to watch animations of driving scenes lasting 18 to 35 seconds. After each scene, participants had to indicate the positions of surrounding cars in a top-down view. Gugerty determined participants' level of SA by comparing the reported positions of the cars with the 'true' (known) positions of the cars. While 'time' was not a factor taken into account in his study, Gugerty's method forms the basis of the present MSc thesis into the amount of time needed for the second step of a ToC, where the traffic situation is assessed and SA is built up.

This study aims to establish whether Gugerty's (1997) [17] method is suited for finding the preparation time needed for the second step of a ToC, as well as uses it to analyse the effect of six different preparation times (i.e., video lengths) on the level of SA that people attain, in situations of two levels of difficulty (i.e., having either 4 or 6 cars of surrounding traffic). The level of SA was determined by having participants reconstruct the end of 26 traffic scenarios, by indicating the positions and relative speeds of the surrounding traffic in a top-down view of the road, and assessing their performance on these reconstructions. Using 4 or 6 cars in the scenarios is in accordance with Pylyshyn and Storm (1988) [18], who found that people can track up to five moving objects in a perceptual task, and Gugerty (1997) [17], who used 3 to 8 cars in his research. Performance for the shorter videos is expected to be lower than for the longer videos, whereas performance between the longest two videos is expected to not differ

significantly, indicating a saturation level of SA. This saturation level is expected to take longer to attain with increased difficulty of the scenario.

The obtained levels of SA as a function of preparation time may serve as a basis for future studies (using more immersive conditions) on how SA builds up over time. Furthermore, values found for the time to regain SA may be used in the design of ToC protocols, and may form a basis for the time given to and/or imposed on drivers for the preparation of take-overs in highly automated driving.

Method

Ethics statement. This research was approved by the Human Research Ethics Committee (HREC) of the Delft University of Technology (TU Delft). All participants provided written informed consent (see Appendix A1-1).

Equipment. Both the videos and graphical user interface (GUI) were presented on a NEC MultiSync EA244Wmi 24 inch widescreen HD monitor, belonging to the SmartEye DR120 remote eye tracking system. The DR120 contained two integrated infrared cameras functioning at a sampling rate of 120 Hz [19]. The collected eye-tracking data will be used in a separate study. Traffic scenarios were reproduced using a standard Dell mouse. The test was conducted by Matlab R2013b (32bit) [20] on a HP EliteBook 8560w running on windows 7 (64bit). The setup can be seen in Figure 1.

The scenarios were programmed using PreScan 7.0 [21] and recorded as four separate videos with a resolution of 2080x1200 pixels for the first person view, 2160x600 pixels for the rear view mirror, and 121x600 pixels for the wing mirrors. Cameras for the mirrors were located and oriented to resemble reality (see Appendix A2-3: Placement of four cameras), and their feed was mirrored. The four videos per scenario (i.e., first person view and the three mirrors) were edited into one video of 1920x1080 pixels, with sound, with the use of VSDC Free Video Editor version 2.3.0.377 [22] (see Appendix A2-3). The Matlab program used Windows Media Player 12 [23] ActiveX control to play the videos.

Figure 1: Setup of experiment

Videos. Each video began with a five second target display for participants to focus on (see Figure 2a), after which the scenario started. At the end of the scenario, a black screen was displayed for 0.5 seconds. During the target display and the traffic scene, a sound of driving on a highway was played. This sound was unrelated to the specific traffic scenario.

The rear view mirror and left wing mirror were positioned in such a way as to resemble real positions. This was not possible for the right wing mirror, which was therefore placed at the right edge of the video at the same height as the left wing mirror. A black rectangle was placed behind each mirror, giving them an outline in order to make them more visible (see Figure 2b).

Participants. Thirty-four participants (5 female, 29 male), aged between 20 and 31 years (M = 24.6, SD = 2.6 years) volunteered to participate in this study. All participants had a valid driving license. All participants read and signed an informed consent form (see Appendix A1-1), explaining the purpose and procedures of the experiment. Participants received € 5 compensation for their time.

Participants were split into two groups (A and B), based on their participant number (i.e., group A if participant number was odd, B if even). These two groups viewed videos of the same traffic scenarios, but of different lengths. The videos had the same ending for both groups, but a different starting point within the scenario. Both groups consisted of 17 participants: 3 female, 14 male, aged between 21 and 31 (M = 25, SD = 2.8 years) for group A, where group B had 2 female, and 15 male participants, aged between 20 and 29 (M = 24.2, SD = 2.4 years).

Procedure. Prior to the test, the participants filled out a general questionnaire about their driving experience and habits (see Appendix A1-2).

Participants were asked to adjust their chair in order to directly face the screen mid-front, with the hind legs of the chair within a demarcation on the floor, approximately 65 cm away from the screen. The height of the screen was adjusted to the participant's height, thus ensuring comparable conditions for each participant.

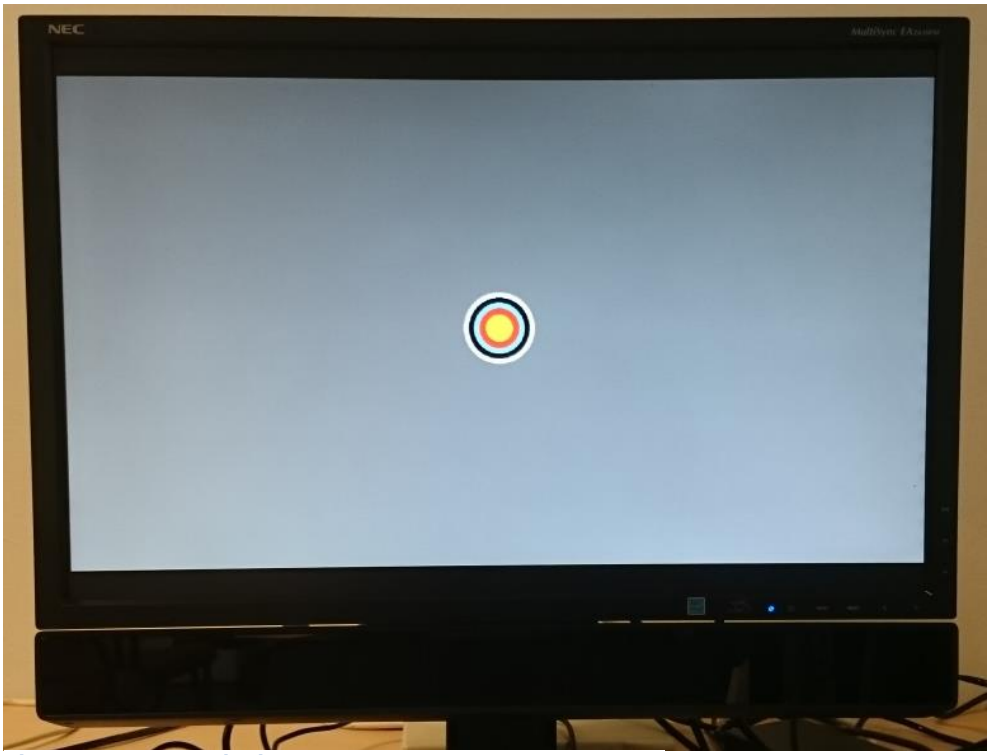
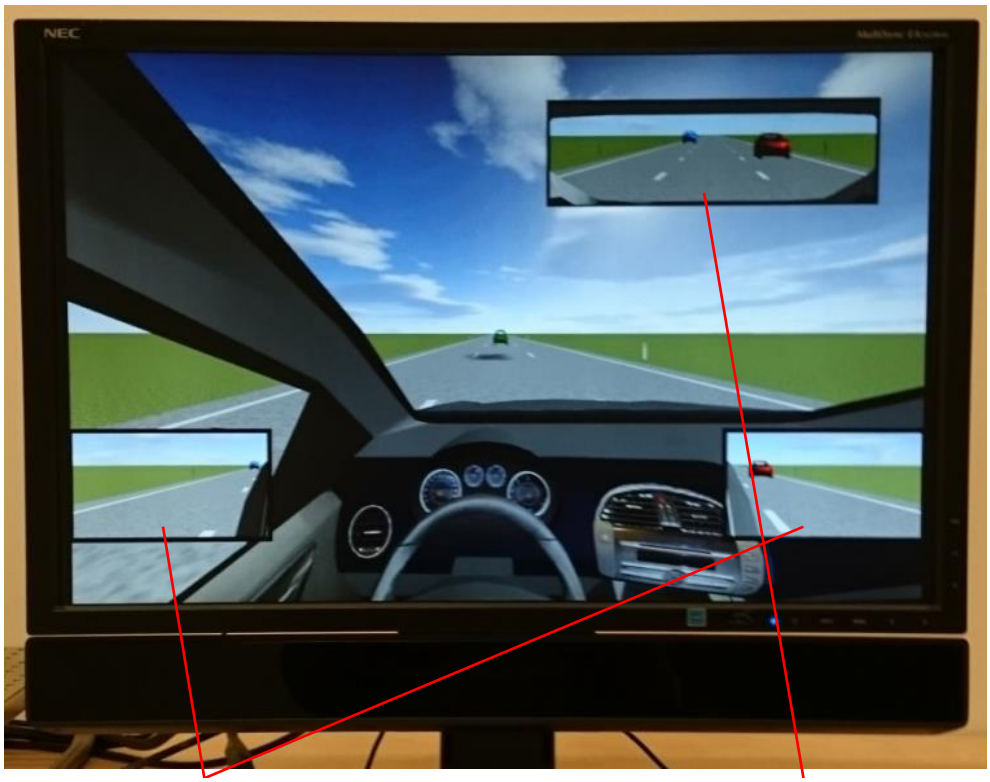


Figure 2a: Target Display



Wing Mirrors Rear View Mirror

Figure 2b: Scene and Mirror Display

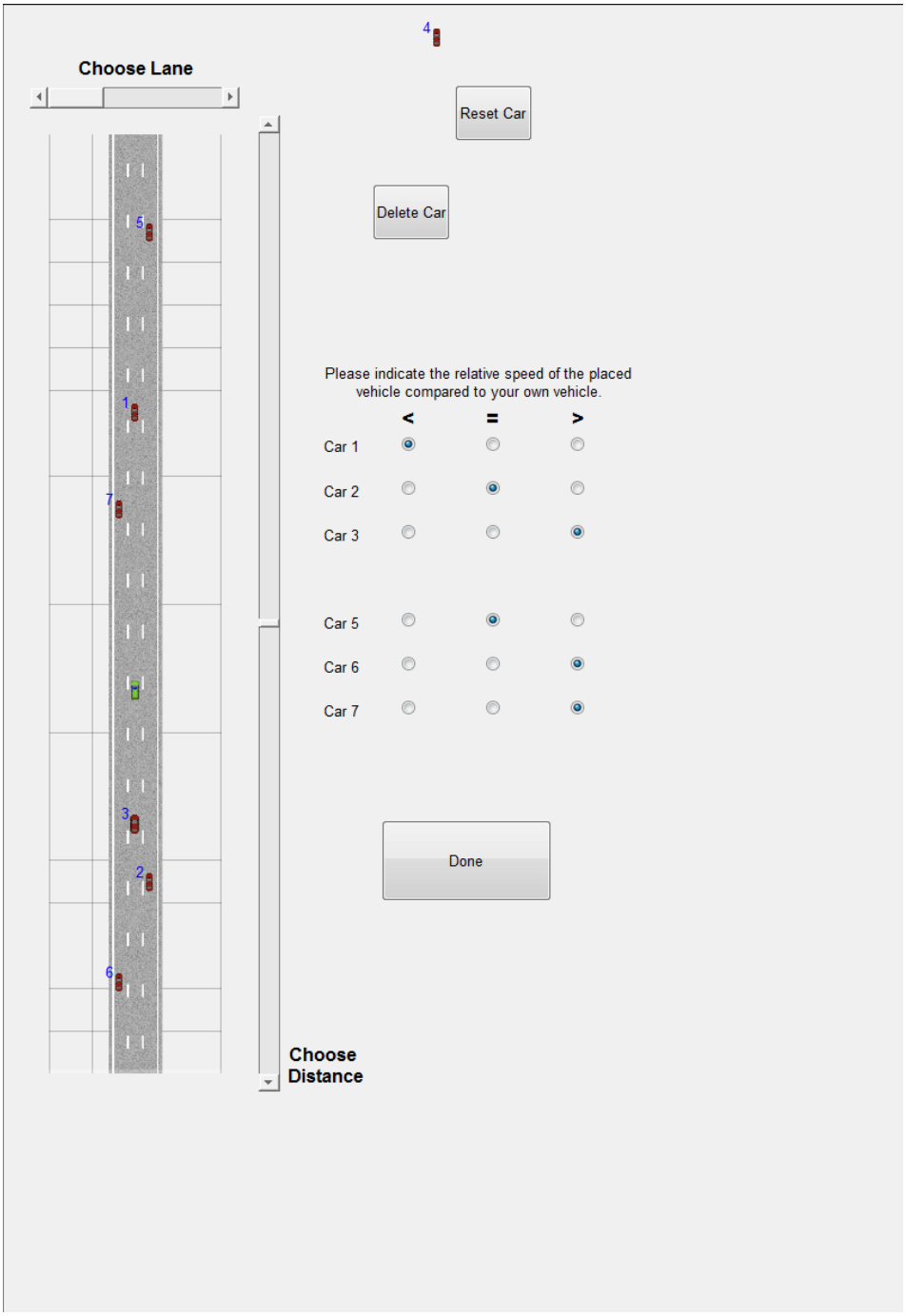


Figure 3: Reproduction of scenarios. Red cars represent the surrounding cars. The green car represents the ego-vehicle. The slider bars were used to set the positions of surrounding vehicles. The radio buttons were used to indicate the relative speed. Placed cars could be deleted by pressing the 'Delete Car' button, and deleted cars could be placed again by pressing the 'Reset Car' button. Additional cars could be placed by pressing the 'New Car' button (only visible when no cars were deleted). The 'Done' button was pressed when a participant had finished the reproduction.

Participants were given 2 training trials, followed by 24 test trials, viewing videos of traffic situations on a three-lane highway. After each video, participants were asked to reproduce the final positions of the surrounding cars, as well as their speeds, in relation to the ego-vehicle (see Figure 3). Placement of at least one vehicle was mandatory, and up to eight vehicles could be placed. The total number, distance and speed relative to the ego-vehicle, and lane of each placed car were collected for each trial.

After finishing a reproduction, a two item questionnaire assessed the subjective difficulty of the task and the preparation time (i.e., the length of the videos) on a scale from 0 (*completely disagree*) to 100 (*completely agree*) (Figure 4). Trials were finished by showing the actual situation in a second top-down view, providing feedback to the participant (Figure 5).

The duration of the experiment was approximately 60 minutes, which includes filling out the forms.

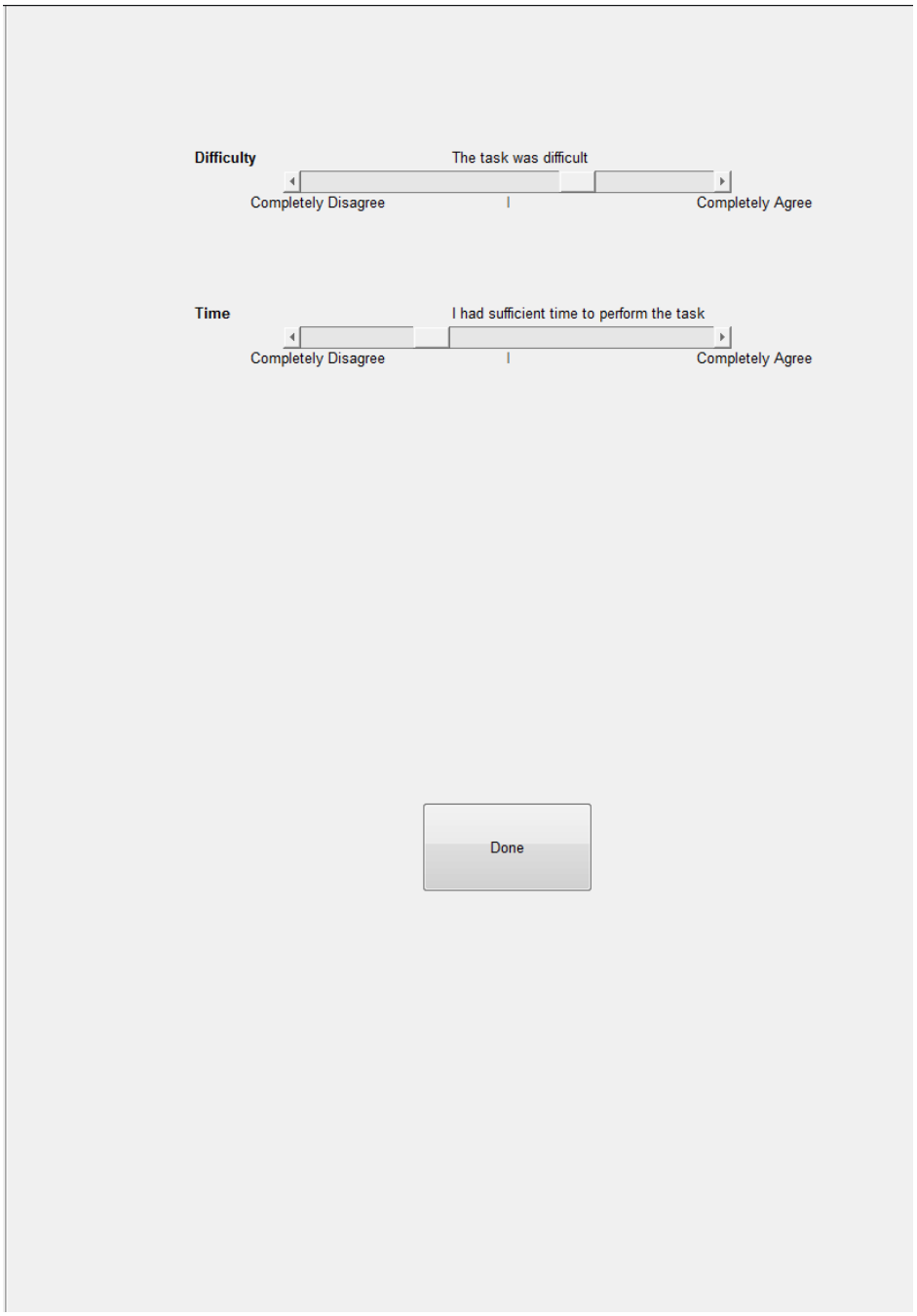


Figure 4: Difficulty and time questions presented to participants after finishing each reproduction

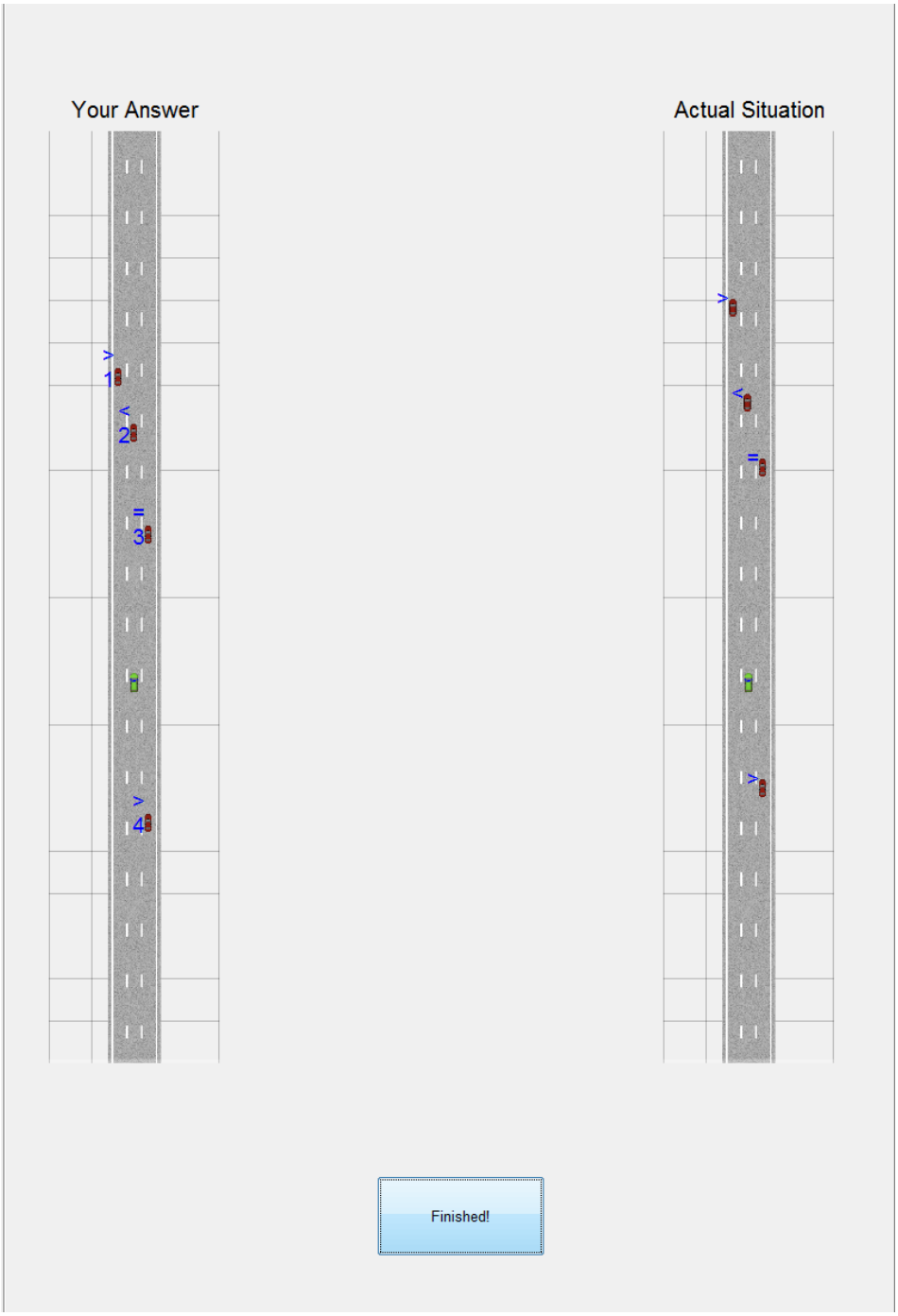


Figure 5: Feedback presented to participants after finishing the answering of the difficulty and time questions

Scenarios. Participants viewed videos of traffic scenarios situated on a three-lane highway, on which the ego-vehicle was driving in the middle lane with a constant speed of 28 m/s (100.8 km/h).

Training trials lasted 12 seconds and featured 3 cars of surrounding traffic, whereas test trials either incorporated 4 or 6 surrounding cars. Both traffic densities were used twelve times. Test trials lasted 1, 3, 7, 9, 12 or 20 seconds. Each duration was repeated twice per traffic density and four times in total (Table 1). Test trials were shown to each participant in randomized order (see Appendix A2-2). Car positions in scenarios 7 and 8 were mirrored (i.e., traffic on the left and right lanes was swapped) in scenarios 11 and 9, respectively, in order to create comparable, yet different, circumstances.

All traffic in each scenario met the following criteria. Each car was

- within a range of 120 m ahead of and 80 m behind the ego-vehicle during the full length of the scenario
- visible during the full length of the scenario, except when driving through the blind spot
- starting and ending, either partially or fully, outside of the blind spot
- driving at one of five constant speeds: 25.5, 26.75, 28, 29.25 or 30.5 m/s
- staying in its own lane during the full length of the scenario
- at least 5 m in front of or behind other cars at all times

The visibility criterion had the effect that there could be only one car directly in front of and/or behind the ego-vehicle on the middle lane. Additional criteria for traffic in four and six car scenarios were set. Specifically, there could be either 0 to 2 or 1 to 3 cars per lane, and speeds could either be used twice or thrice maximally in four or six-car scenarios, respectively.

The criteria mentioned above allowed for a limited number of distributions over the three lanes. Table 1 indicates the number of cars in the left, middle, and right lane, for each of the 24 scenarios.

The cars' speeds determined the possible range of starting positions for each car. This range of starting positions was divided into increments of 0.25 m, from which a random one was chosen as starting position. Each car's model and colour was randomly assigned out of 13 possible colours and 10 possible models. Video lengths were randomly distributed over the four and six car scenarios for group A. In order to be able to compare effects of different times for the same scenarios, the video lengths for group B were chosen one or two time-steps next to the times of the scenarios for group A. For a summary of scenario setups, see Table 1. For a complete overview of each scenario, see Appendix A2-1.

Instructions to participants. Participants received information about the procedures, and instructions on the informed consent form (Appendix A1-1). The instructions were stated as follows:

> "you will watch 26 videos of traffic scenarios on a three-lane highway (see figure), 2 of which are used as training, followed by 24 test trials. The scenarios vary in length from 1 to 20 seconds.
>
> After each video, you are asked to reproduce the traffic situation as accurately as possible. This means you have to position the cars and indicate their relative speeds in a top-down view. Furthermore, after each video, you are asked to fill out a questionnaire about your subjective difficulty and time pressure."

In addition to the information on the informed consent form, participants received oral instructions during the training trials about (1) how to work with the interface (e.g., how to use the slider bars and how to place cars), (2) how to interpret the 'time question' (i.e., "I had sufficient time to perform the task"), which should be interpreted as "the video was long enough for me to be able to perform the task", and (3) that surrounding traffic would not necessarily follow traffic rules (e.g., overtake on the right).

Table 1: Setup per scenario, indicating number of cars in the scenario, video lengths, and information about surrounding cars: Distribution over the lanes, end distance (relative to the ego-vehicle), and speed. Distances are ordered from low to high per lane, from left to right lane. <<, <, =, >, and > indicate speeds of 25.5, 26.75, 28, 29.15, and 30.5 m/s, respectively. Red and green colour bars indicate whether a car was behind or in front of the ego-vehicle. The size of the bar linearly corresponds to the relative distance.

Scenario Nr.	# Cars	Length Group A (s)	Length Group B (s)	Distri-bution	Relative End Distances (m) and Speeds								
					Left Lane			Middle Lane			Right Lane		
1	6	7	9	3-1-2	-32.25 >	-0.75 <<	107.50 =	-61.25 <			13.50 >	70.00 <<	
2	6	9	12	2-1-3	7.50 >>	43.25 =		22.50 <<			-42.25 <<	47.75 >	65.50 <
3	6	3	7	1-2-3	-54.00 =			-79.00 <<	22.00 <<		-77.75 =	-5.75 >>	4.00 >
4	6	3	9	3-2-1	-59.00 <<	-27.75 <	106.75 >>	-61.75 <	82.00 >>		84.00 =		
5	6	20	1	3-1-2	-71.50 =	-56.75 <<	33.75 <	48.00 <			-32.75 <<	87.75 >>	
6	6	9	20	2-2-2	-28.75 <	55.25 >		-66.25 <<	82.25 >		21.25 >	59.50 >>	
7	6	12	20	2-1-3	-9.50 <<	76.75 =		-49.00 <			-56.50 =	-45.75 <	25.75 >
8	6	1	3	3-2-1	-15.50 >>	15.75 =	49.25 >	-36.00 =	23.25 <		44.75 =		
9	6	20	3	1-2-3	44.75 =			-36.00 =	23.25 <		-15.50 >>	15.75 =	49.25 >
10	6	1	7	2-2-2	-12.00 <<	85.25 >		-33.25 =	40.50 >		-42.50 >	23.25 >>	
11	6	7	12	3-1-2	-56.50 =	-45.75 <	25.75 >	-49.00 <			-9.50 <<	76.75 =	
12	6	12	1	3-2-1	-62.00 <<	48.25 >	92.25 >>	-48.25 <	14.50 <<		36.50 =		
13	6	9	12	1-2-1	-52.50 <			-25.50 =	116.25 >>		42.75 >>		
14	4	12	20	1-1-2	89.00 >>			66.50 <<			-24.75 >>	51.25 =	
15	4	20	1	2-1-1	-74.50 <	68.50 >		47.75 >			73.25 <<		
16	4	7	9	2-0-2	-25.00 >>	62.75 <<					-28.25 <	36.75 >	
17	4	12	1	0-2-2				-15.75 >	42.25 >		-33.50 <<	71.00 =	
18	4	3	7	2-2-0	-59.25 <<	32.25 >		-28.75 >	23.50 <<				
19	4	7	12	1-2-1	47.50 >			-65.75 =	83.50 <		12.75 >>		
20	4	1	3	1-1-2	60.25 <<			-68.25 <<			17.25 >	33.75 =	
21	4	9	20	2-1-1	-34.00 <	50.75 =		42.00 <<			101.00 >>		
22	4	3	9	2-0-2	-49.50 =	44.00 <<					14.25 <	119.50 >>	
23	4	1	7	0-2-2				-59.75 =	69.75 >>		-54.75 <<	36.00 <	
24	4	20	3	2-2-0	-9.75 =	22.00 <		-44.50 >	66.00 >>				

Data Processing. Due to technical malfunctions, and unwanted participants' behaviour and/or distraction, parts of the data were excluded from the analyses (Table 2). In total, reconstruction data for 56 trials from 7 participants (out of a total of 34 participants * 24 trials = 816 trials) were left out of consideration. A total of 26 trials from 3 participants for the difficulty question, and 50 trials from 4 participants for the time question were removed.

Table 2: Excluded data. 'All data' includes the number of placed cars, reconstruction data, and answers to the self-report questions. 'Reconstruction data' encompasses positions, lanes, and speeds of placed cars.

Malfunction / behaviour / distraction	Number of participants	Total number of erroneous trials	Data not included for erroneous trials
Video malfunction (i.e., video did not show)	1	1	All data
Misunderstanding of GUI controls	1	1	
Using hands / fingers as memory support	2	48	Reconstruction data
Accidentally pressed 'Done' button	1	1	
Reconstructed beginning (instead of ending) of scene	2	5	
Answered 'Time question' wrongly	1	24	input for the 'time question'
Did not answer both questions	1	24	input for both the 'difficulty' and 'time' question

Several error scores were calculated from the remaining data. The scores can be divided into a basic and advanced scores, with the basic score providing a general indication of effects, while advanced scores give a more in-depth view, as well as being potentially better suited to detect differences between video lengths.

The basic score is the error percentage in the number of placed cars, defined as the absolute error in the number of placed cars, divided by the number of actual cars, times 100%, see Eq. 1:

$$\text{Error Percentage} = \frac{\left|\#Cars_{Placed} - \#Cars_{Actual}\right|}{\#Cars_{Actual}} \cdot 100 \quad (1)$$

Advanced scores were derived using a 'best match' approach, where placed cars were matched with actual cars as assumed to be intended by the participant [17]. The matching algorithm first determined the absolute longitudinal distance of each placed car to each actual car. When this distance was less than 38 m, the placed car was a possible match for the actual car. Total absolute error in distance was calculated for all possible combinations of possible matches, in which each actual car and each placed car could only be used once, with distance entailing both absolute longitudinal distance and absolute lane distance (i.e., lateral distance), with each next lane equalling 3.5 m. During matching, lane distance was multiplied by five (i.e., 17.5 or 35 meters for an error of one or two lanes, respectively) to par the maximum error in lanes with the maximal error in longitudinal distance, see Eq. 2:

$$\text{Total Absolute Error Distance} = \left|d_L Cars_{Placed} - d_L Cars_{Actual}\right| + 5 \cdot \left|d_l Cars_{Placed} - d_l Cars_{Actual}\right| \quad (2)$$

, in which d_L is longitudinal distance to the ego-vehicle, and d_l is lateral distance to the ego-vehicle.

The sequence of matches resulting in the lowest total absolute error in distance was assumed to be the intended placement. For an example of matching, see Appendix A3. The following advanced scores were calculated for each scenario:

- Hit percentage (# hits/#actual cars *100) (%).
- Average absolute error distance of hits (m) (Eq. 3).
- Average absolute error distance of hits and misses (m) (Eq. 4).
- Error percentage of indicated lanes (#falsely indicated lanes/#hits * 100) (%).
- Error percentage of indicated speeds (#falsely indicated speeds/#hits * 100) (%).
- Average size of errors in indicated speeds (i.e., sum of number of 'steps' from actual speed/#hits). Errors in indicated speeds were determined in two steps for each hit. First it was determined whether the indicated relative speed (i.e., slower than, same speed as, or faster than the ego-vehicle) was correct, by checking if the sign of actual and indicated relative speeds matched (i.e., <0, 0, or >0). When the indicated relative speed was incorrect, the number of errors in speed went up by one. The size of the error depended on the actual speed, as indicated in Table 3.

$$\text{Average Error Distance of Hits} = \frac{\left|d_L Cars_{Placed} - d_L Cars_{Actual}\right| + \cdot \left|d_l Cars_{Placed} - d_l Cars_{Actual}\right|}{\#Hits} \quad (3)$$

$$\text{Average Error Distance of Hits and Misses} = \frac{\left|d_L Cars_{Placed} - d_L Cars_{Actual}\right| + \cdot \left|d_l Cars_{Placed} - d_l Cars_{Actual}\right| + 38 \cdot \#Misses}{\#Hits + \#Misses} \quad (4)$$

, where 38 is the maximum longitudinal distance for matching.

Table 3: Size of errors in speeds for the different combinations of actual speed of a surrounding car and indicated relative speed. The ego-vehicle had a speed of 28 m/s in all scenarios.

Actual speed (m/s)	25.5			26.75			28			29.25			30.5		
Indicated relative speed	<	=	>	<	=	>	<	=	>	<	=	>	<	=	>
Size of error (# 'steps')	0	2	3	0	1	2	1	0	1	2	1	0	3	2	0

The matching algorithm was validated by comparing it to 12 matches made by the first three participants of this study, and with 17 matches made by six participants in a pilot study (see Appendix B: Material Regarding the Preliminary (Pilot) Study). These nine participants were asked to further elaborate on the placement of their cars by indicating the specific cars their placed cars were intended to match in a table on a special form (Appendix A1-3: Questionnaire for indicating intended matches), during the feedback stage. The algorithm matched the participants' intended cars in 23 of 29 cases, not taking into account misses caused by placement further than 38 m away (see Appendix A3-1 for an example of this type of miss). In the remaining 6 cases, cars were partially matched correctly. Mismatches yield minor errors for the total error distances, but can cause larger fluctuations in error percentages of lanes and speeds, and in size of average size of errors in indicated speeds (see Appendix A3-2 for an example of a mismatch).

Statistical Analyses. Differences in scores and ratings were analysed with respect to (1) pairs of video lengths and (2) traffic density. For scores expressed as error percentages, and the average size of errors in indicated speeds, Wilcoxon's signed rank test was used with a significance level of 0.01. For remaining scores (i.e., error distances of hits, and of hits and misses) and the self-report ratings, paired samples *t* tests were used with a significance level of 0.01. The conservative alpha value was selected to reduce the probability of false positives.

Results

The results from the different measures are shown and explained in the following sections. For each section extra results from statistical tests regarding the effects of video length within scenarios can be seen in Appendix A4 (Difficulty and time ratings, Basic score, and advanced scores in Appendices A4-1, A4-2, and A4-3, respectively).

Difficulty and time ratings

Difficulty Rating was higher for shorter videos of the same scenario, with the exception of scenario 2 (Figure 6). Figure 6 also shows that scenarios with higher traffic density (scenarios 1–12) had overall higher difficulty ratings than scenarios with lower traffic density (scenarios 13–24). Ratings between mirrored scenarios of same length did not show statistically significant differences (p = 0.380, t = 0.904, df = 15 for 7A vs. 11B, and p = 0.617, t = 0.511, df = 15 for 8B vs. 9B). The effects of video length were statistically significant for scenarios 4, 8, 10, and 24.

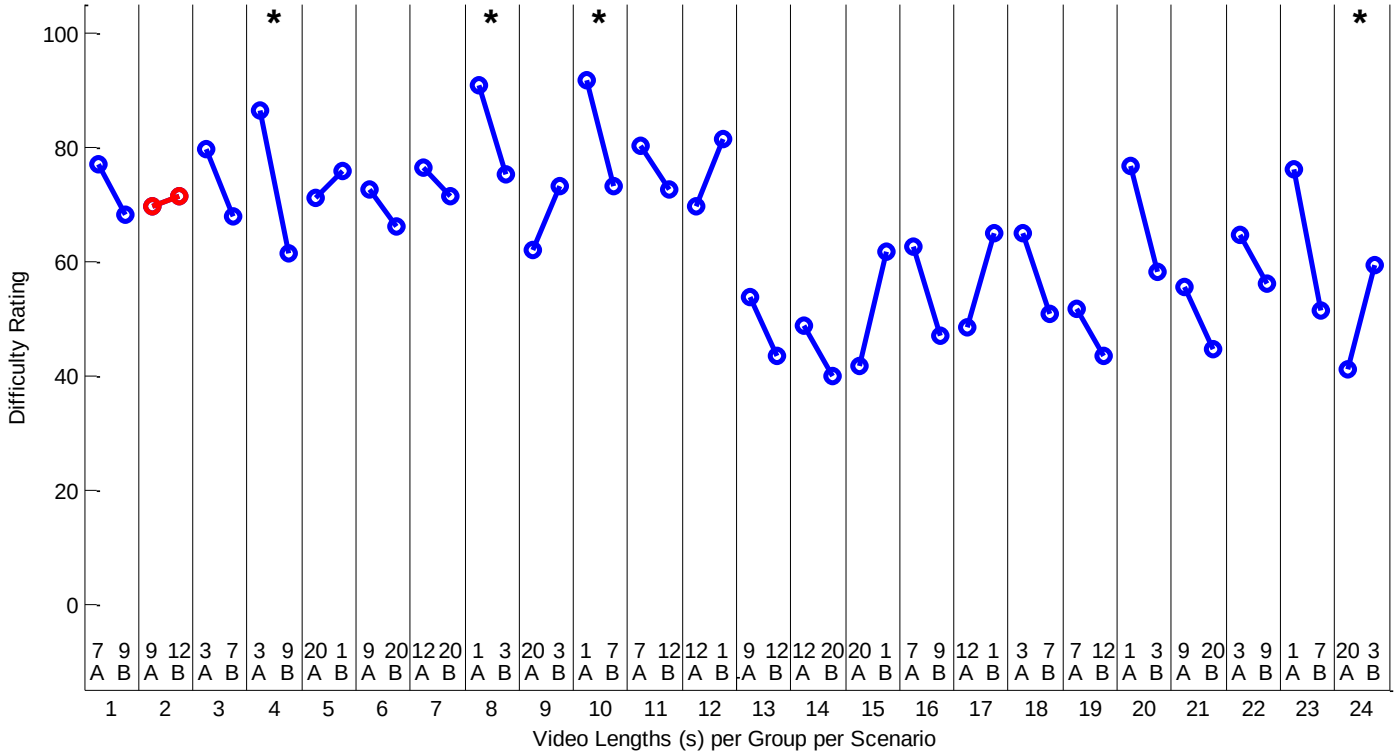


Figure 6: Self-reported difficulty rating as a function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean difficulty rating, where the colour of dots/lines indicates whether difficulty rating was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in difficulty rating, between video lengths within a scenario, are statistically significant (p < 0.01).

As can be seen in Figure 7 and Table 4, traffic density yielded a statistically significant (p < 0.001) negative effect on difficulty rating, both for each separate video length and when averaged over all video lengths. Difficulty ratings were lower for longer scenarios for both traffic densities. This effect was not always significant when comparing between two adjacent video lengths, but was statistically significant when ratings were compared between videos with a length of 20 seconds to those of 7 seconds or shorter.

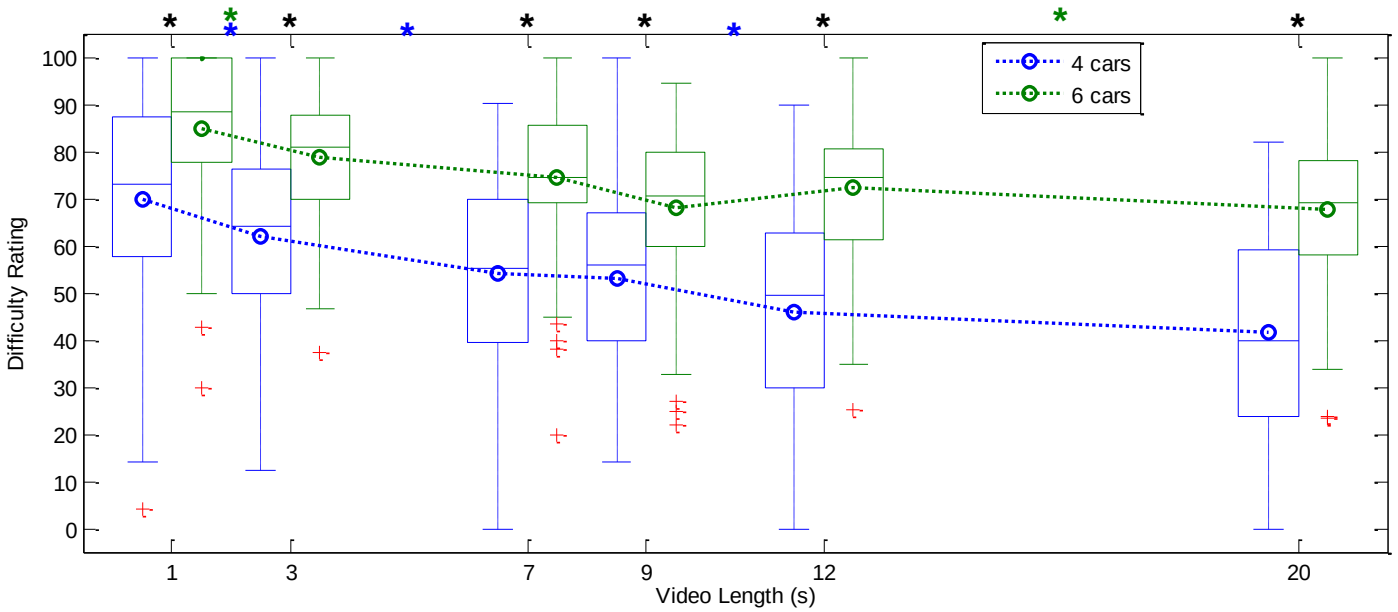


Figure 7: Self-reported difficulty rating as a function of video length and traffic density. Boxplots represent difficulty ratings of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in difficulty rating, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in difficulty rating between traffic densities for a specific video length.

Table 4: Results of paired samples *t* tests for differences in self-reported difficulty rating. The top part shows statistical significance of effects of video length, where difficulty ratings between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where difficulty ratings are compared between the two traffic densities for each specific video length (left), and for all video lengths combined (right). Statistically significant differences ($p < 0.01$) are indicated in boldface.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	**0.005**	**0.008**	0.728	**0.007**	0.129	**0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	t	**3.021**	**2.857**	0.351	**2.909**	-1.559	**-3.553**	**-3.896**	**-5.330**	**-6.880**
	df	**32**	**32**	32	**32**	32	**32**	**32**	**32**	**32**
6 cars	***p***	**0.007**	0.116	0.017	0.055	**0.008**	0.859	**0.005**	**< 0.001**	**< 0.001**
	t	**2.873**	1.618	2.528	-1.994	**-2.817**	-0.180	**-3.016**	**-4.264**	**-6.166**
	df	**32**	32	32	32	**32**	32	**32**	**32**	**32**
Combined	***p***	**< 0.001**	**0.007**	0.059	0.497	0.018	0.019	**< 0.001**	**< 0.001**	**< 0.001**
	t	**3.6857**	**2.868**	1.954	0.688	-2.493	-2.473	**-4.438**	**-5.838**	**-7.407**
	df	**32**	**32**	32	32	32	32	**32**	**32**	**32**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**		**< 0.001**	
	t	**-4.710**	**-6.919**	**-5.980**	**-5.967**	**-8.266**	**-8.438**		**-9.448**	
	df	**32**	**32**	**32**	**32**	**32**	**32**		**32**	

Time Rating was scored higher for longer videos of the same scenario in all cases (Figure 8*)*. Differences in scores per scenario were statistically significant ($p < 0.01$) in 14 out of 24 scenarios. Ratings between mirrored scenarios of same length did not show statistically significant differences ($p = 0.465$, $t = -0.751$, $df = 14$ for 7A vs. 11B, and $p = 0.805$, $t = -0.252$, $df = 15$ for 8B vs. 9B).

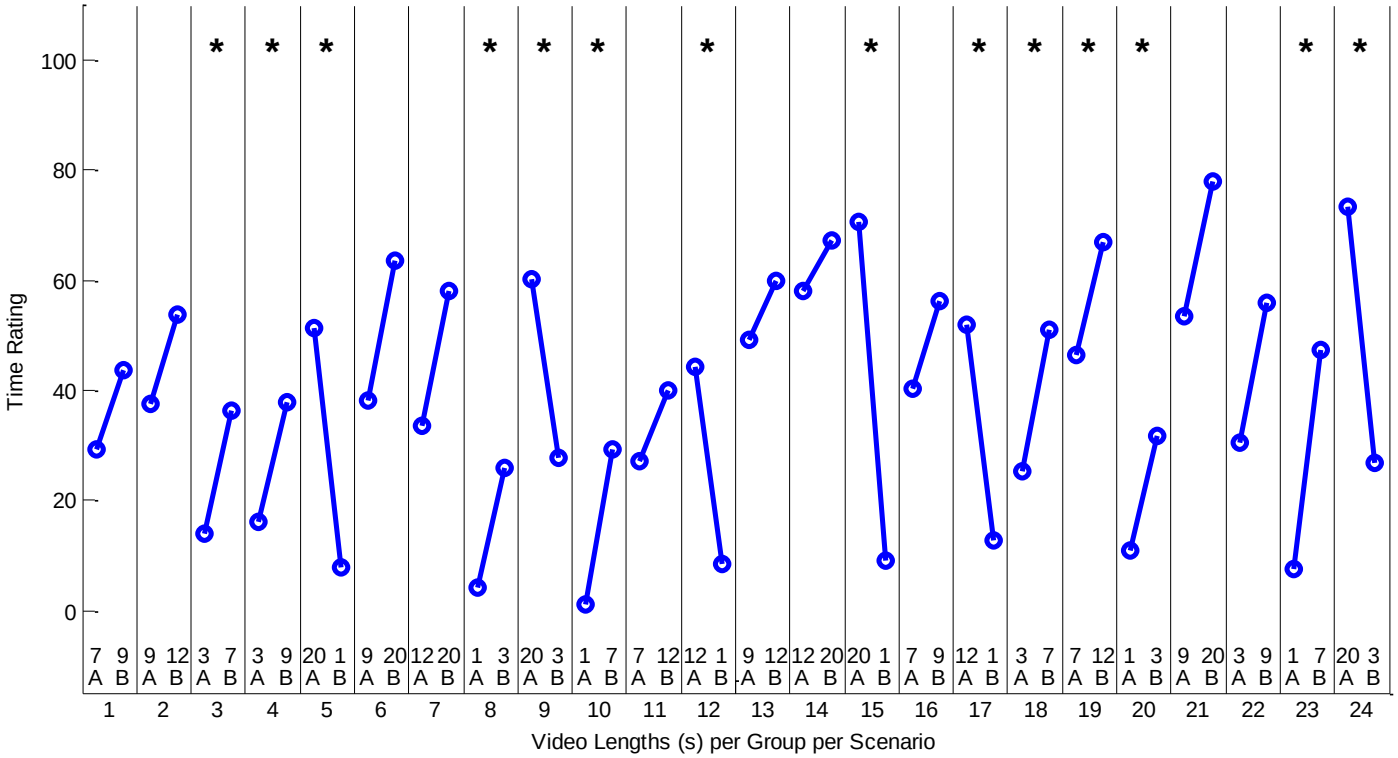


Figure 8: Self-reported time rating as a function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean time rating, where the colour of dots/lines indicates whether time rating was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in time rating, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Traffic density had a statistically significant ($p < 0.01$) negative effect on how participants perceived the amount of time as being sufficient for performing the task. Video length had a positive influence on how participants perceived the amount of time (Figure 9 and
Table 5). This effect, when compared between two adjacent video lengths, was statistically significant ($p < 0.01$) in all but one case (9 vs. 12 seconds length), with six surrounding cars. With four surrounding cars, the effect was not statistically significant in two cases (7 vs. 9, and 9 vs. 12 seconds length). The effect was stronger when ratings were compared between videos with a length of 20 seconds and those shorter than 12 seconds.

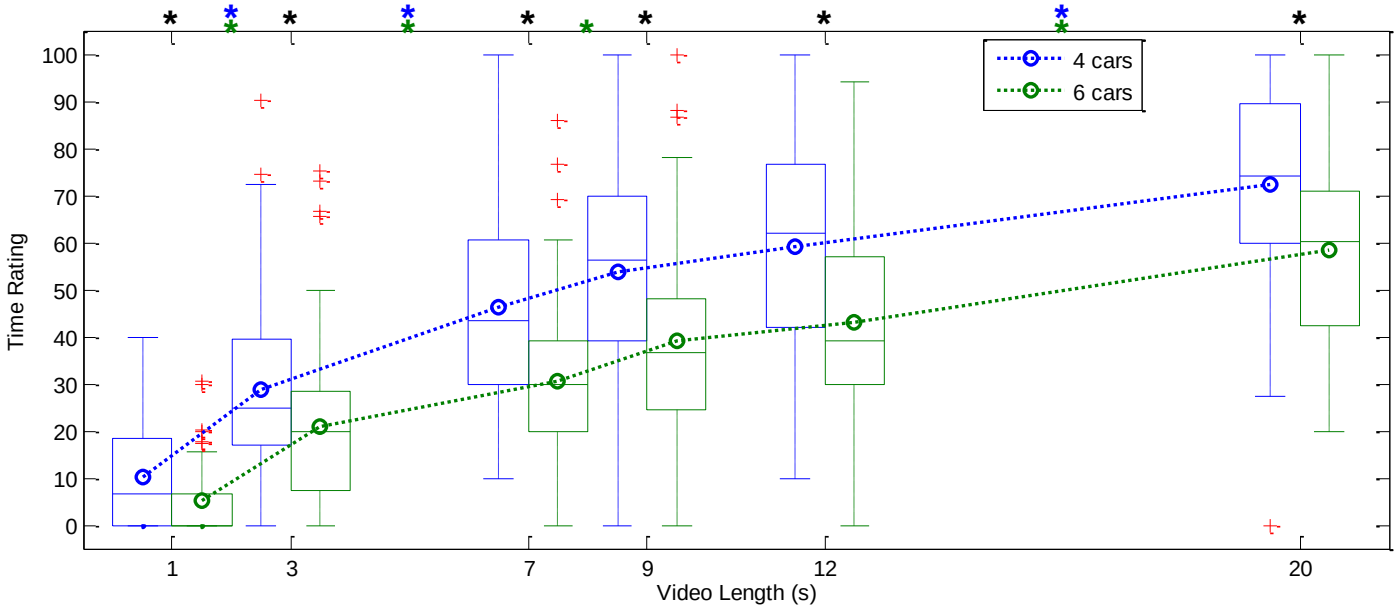


Figure 9: Self-reported time rating as a function of video length and traffic density. Boxplots represent time ratings of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in time rating, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in time rating between traffic densities for a specific video length.

Table 5: Results of paired samples *t* tests for differences in self-reported time rating. The top part shows statistical significance of effects of video length, where time ratings between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where time ratings are compared between the two traffic densities for each specific video length (left), and for all video lengths combined (right). Statistically significant differences ($p < 0.01$) are indicated in boldface.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	**< 0.001**	**< 0.001**	0.017	0.046	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	t	**-8.478**	**-5.609**	-2.531	-2.080	**3.916**	**4.852**	**10.044**	**11.521**	**20.162**
	df	**31**	**31**	31	31	**31**	**31**	**31**	**31**	**31**
6 cars	***p***	**< 0.001**	**< 0.001**	**0.004**	0.140	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	t	**-7.678**	**-4.194**	**-3.160**	-1.514	**6.746**	**8.059**	**9.715**	**12.298**	**16.695**
	df	**31**	**31**	**31**	31	**31**	**31**	**31**	**31**	**31**
Combined	***p***	**< 0.001**	**< 0.001**	**0.001**	0.023	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	t	**-10.459**	**-6.299**	**-3.576**	-2.388	**6.612**	**7.844**	**14.865**	**15.018**	**23.034**
	df	**31**	**31**	**31**	31	**31**	**31**	**31**	**31**	**31**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	**< 0.001**	**0.006**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**		**< 0.001**	
	t	**3.760**	**2.930**	**5.814**	**6.630**	**5.509**	**3.834**		**9.046**	
	df	**31**	**31**	**31**	**31**	**31**	**31**		**31**	

Basic Measure

Error percentage in number of placed cars was lower for longer videos, with the exception of scenarios 9, 14 and 21 (Figure 10). The effect of video length within scenarios was statistically significant ($p < 0.01$) in two scenarios (scenario 5 and 23). Error percentages between mirrored scenarios of same length did not show statistically significant differences ($p = 0.250$, $T = 6$ for 7A vs. 11B, and $p = 0.766$, $T = 10.5$ for 8B vs. 9B). Higher traffic density lead to overall higher error percentages (scenarios 1–12 vs. scenarios 13–24).

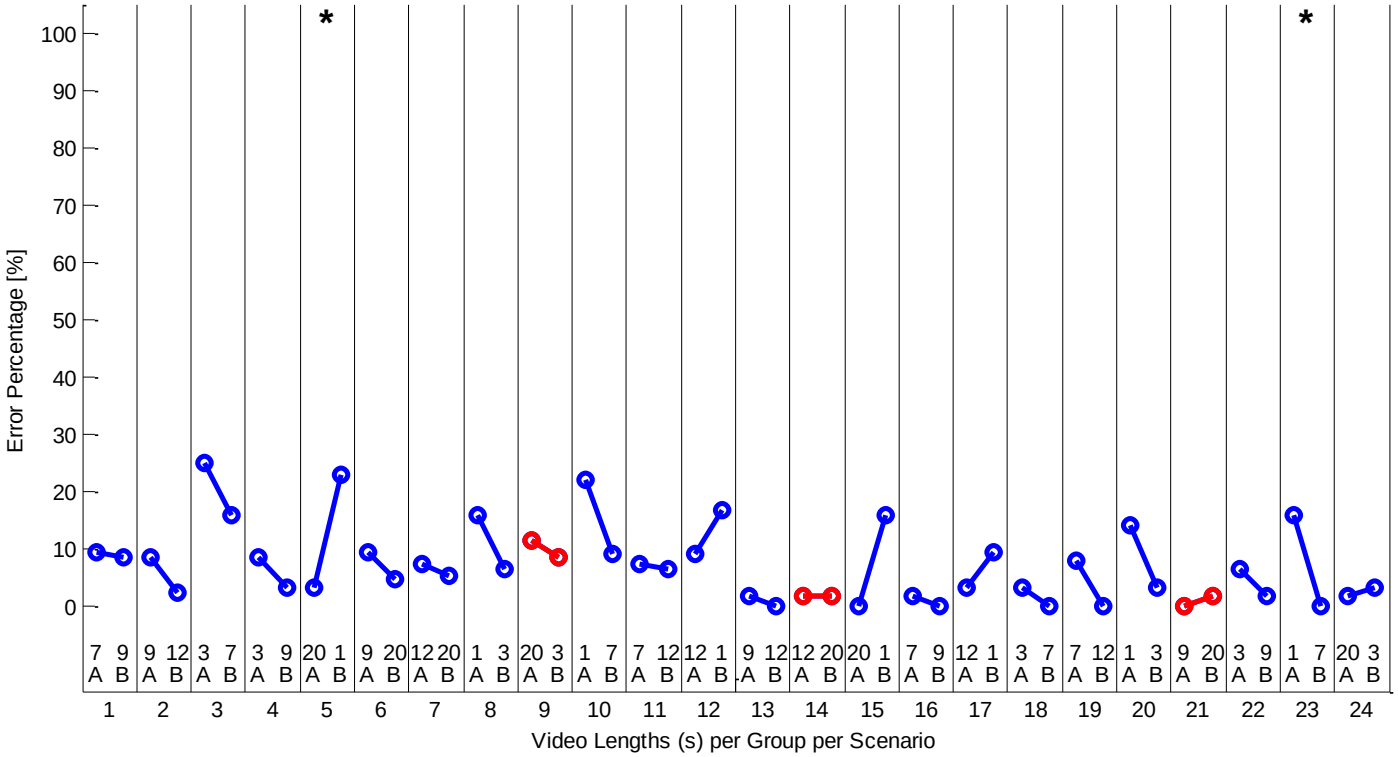


Figure 10: Error percentage in number of placed cars as a function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error percentage, where the colour of dots/lines indicates whether the error percentage was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error percentage, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Higher traffic density yielded a statistically significant increase of error percentage in total number of cars except for videos with a duration of 1 second (Figure 11 and
Table 6). When error percentages between adjacent video lengths were compared, the differences were statistically significant for the 1 vs. 3 second video length case. When error percentages for videos with a length of 20 seconds were compared to those for shorter videos, the difference was statistically significant for the 20 vs. 1 second case with four surrounding cars, and for the 20 vs. 1, 20 vs. 3, and 20 vs. 7 second cases with six surrounding cars. Error percentage in number of placed cars showed a total decrease from 14 to 1 percent and from 19 to 6 percent for four and six car traffic densities, respectively.

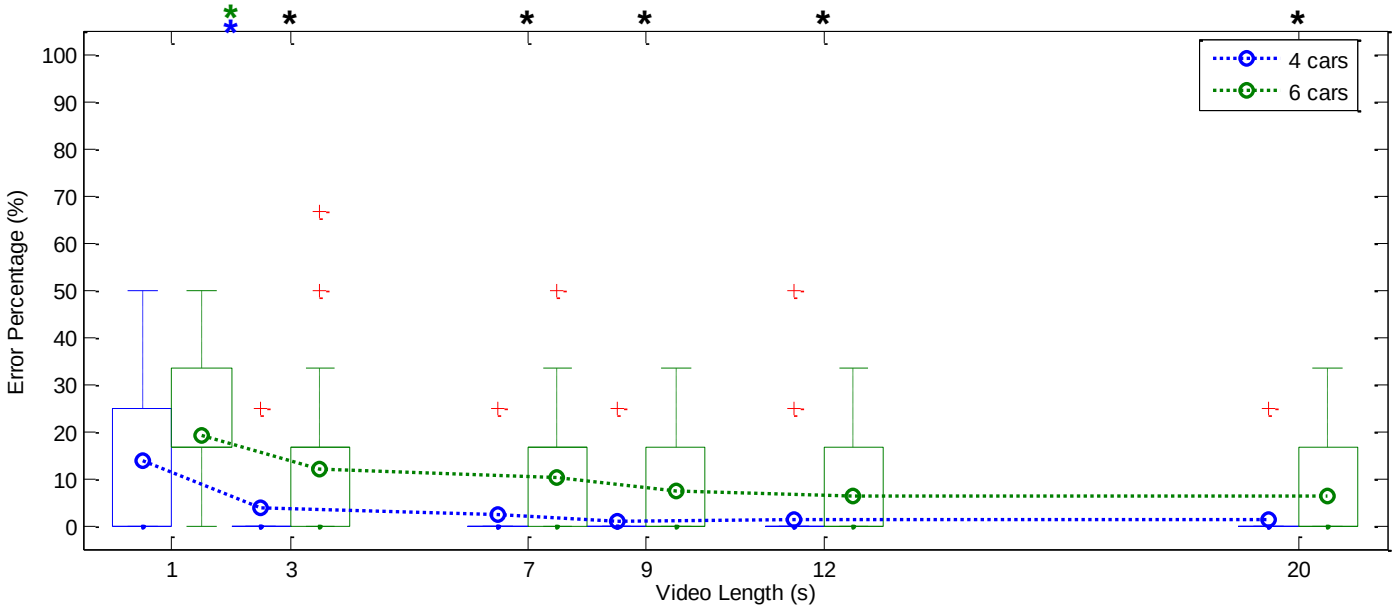


Figure 11: Error percentage in number of placed cars as a function of video length and traffic density. Boxplots represent error percentages of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in error percentage, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in error percentage between traffic densities for a specific video length.

Table 6: Results of Wilcoxon's signed rank tests for differences in error percentage in number of placed cars. The top part shows statistical significance of effects of video length, where error percentages between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where error percentages are compared between the two traffic densities for each specific video length (left), and for all video lengths combined (right). Statistically significant differences ($p < 0.01$) are indicated in boldface. Missing *z* values are caused by sample sizes being too small or lack of different values.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	**0.002**	0.367	0.359	1	1	1	0.453	0.094	**< 0.001**
	z	**3.044**	-	-	-	-	-	-	-	**-3.897**
	T	**154**	50.5	21	4	3	9	8	3	**6**
6 cars	***p***	**0.004**	0.277	0.184	0.537	0.927	0.590	0.024	**0.002**	**< 0.001**
	z	**2.888**	1.087	1.328	0.618	-0.092	-0.538	-2.264	**-3.143**	**-4.204**
	t	**307**	172	104	132.5	83.5	101	93	**28**	**12**
Combined	***p***	**< 0.001**	0.193	0.018	0.576	0.885	0.728	**0.010**	**< 0.001**	**< 0.001**
	z	**3.566**	1.302	2.366	0.559	-0.145	-0.347	**-2.594**	**-3.551**	**-4.614**
	T	**405**	210.5	139.5	169	91.5	150	**100**	**27**	**8.5**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	0.031	**< 0.001**	**< 0.001**	**< 0.001**	**0.004**	**< 0.001**		**< 0.001**	
	z	-2.153	**-3.474**	**-3.793**	**-3.661**	**-2.875**	**-3.191**		**-4.592**	
	T	128.5	**29.5**	**33.5**	**18**	**14**	**19**		**14**	

Advanced Measures

Hit percentage increased with the video length within a scenario, with the exception of six scenarios (Figure 12). The effect of video length within scenarios was statistically significant ($p < 0.01$) in scenario 5). Hit percentages overall were higher for those scenarios that had four cars surrounding the ego-vehicle (scenarios 13–24) than for scenarios with six cars (scenarios 1–12). Hit percentages between mirrored scenarios of same length did not show statistically significant differences ($p = 0.686$, $t = 7$ for 7A vs. 11B, and $p = 0.836$, $t = 25$ for 8B vs. 9B).

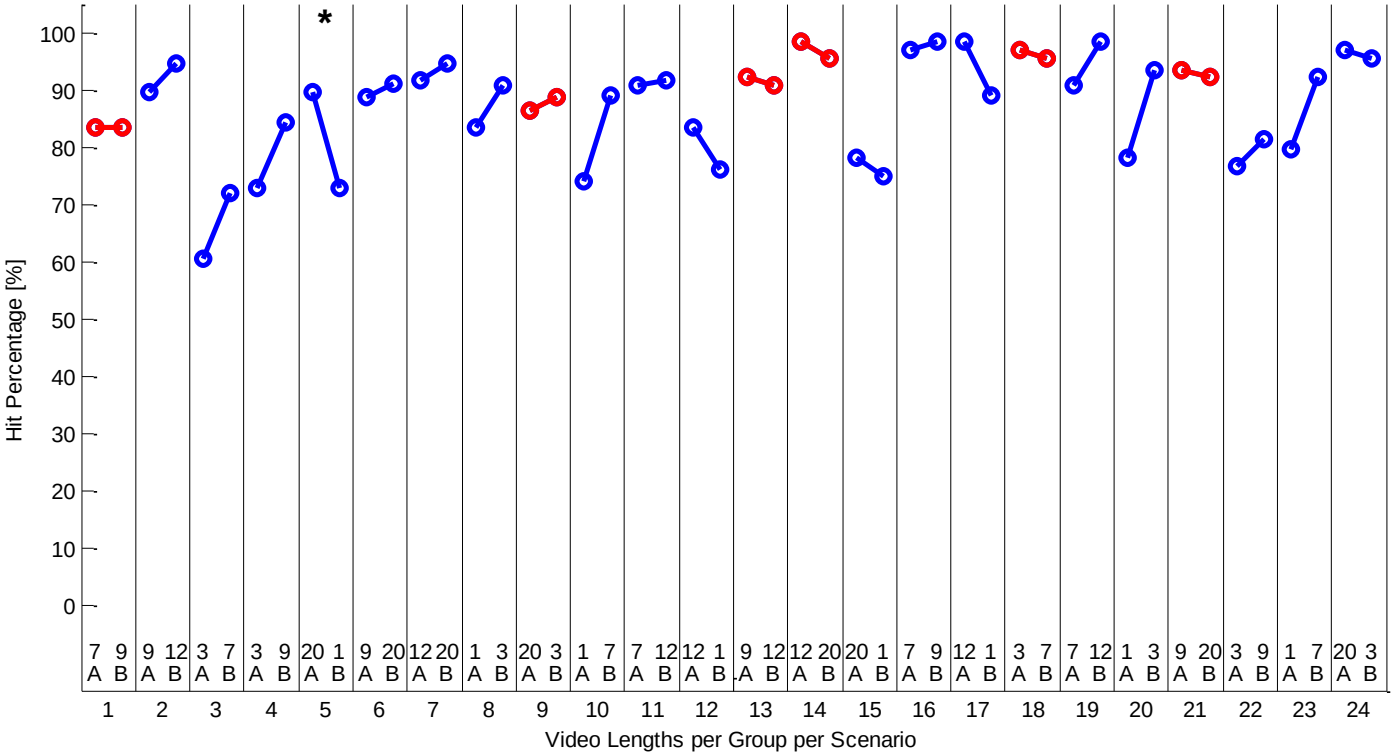


Figure 12: Hit percentage as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean hit percentage, where the colour of dots/lines indicates whether the hit percentage was lower for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in hit percentage, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Hit percentage rose as video length increased (Figure 13). It rose from 76 % (1 s videos) to 90 % (20 s videos) in scenarios containing six cars of surrounding traffic. In scenarios containing four cars, hit percentage showed an increase from 80 to 90 percent, with a maximum of 96 percent being reached around the 12 second mark. When compared between two adjacent video lengths, differences in hit percentage were statistically significant ($p < 0.01$) in the 1 vs. 3, 9 vs. 12, and the 12 vs. 20 seconds cases when there were four surrounding cars (
Table 7). When error percentages for videos with a length of 20 seconds were compared to those for shorter videos, the difference was statistically significant in the 20 vs. 1, and 20 vs. 12 seconds cases with four surrounding vehicles. With six surrounding vehicles, it was statistically significant in the 20 vs. 7, 20 vs. 3, and 20 vs. 1 second cases (
Table 7).
Higher traffic density had a negative effect on hit percentage, which was statistically significant ($p < 0.01$) for three out of six video lengths, and for combined video lengths (
Table 7).

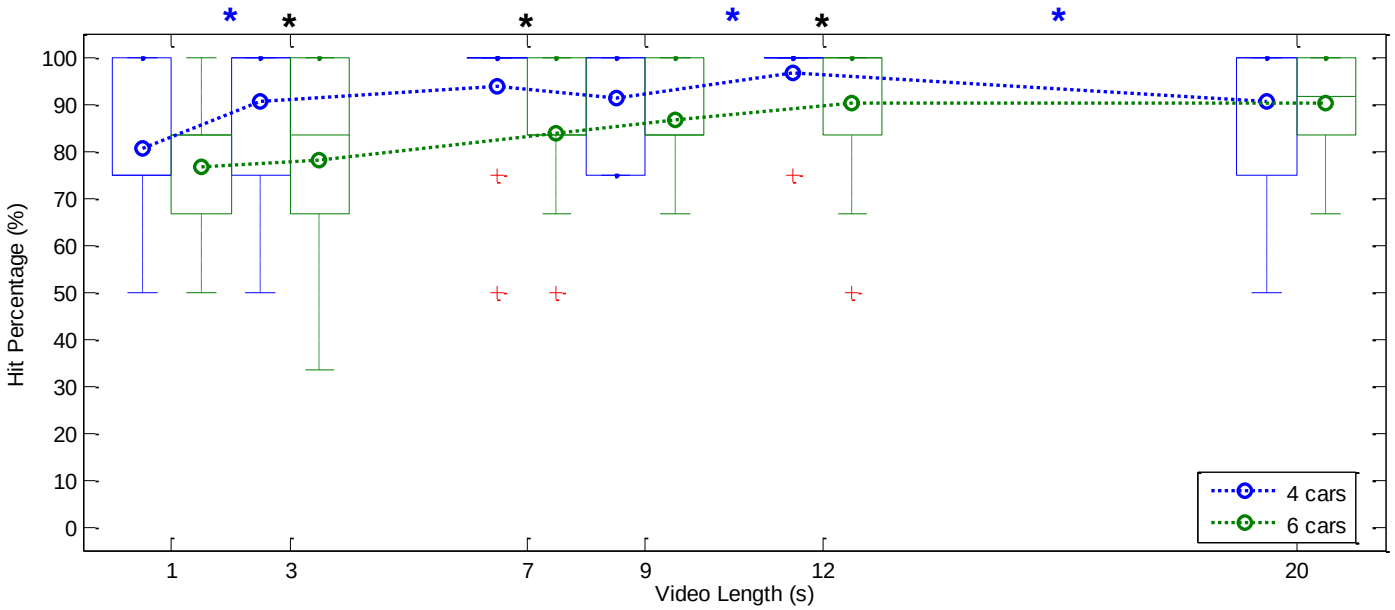


Figure 13: Hit percentage as function of video length and traffic density. Boxplots represent hit percentages of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in hit percentage, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in hit percentage between traffic densities for a specific video length.

Table 7: Results of Wilcoxon's signed rank tests for differences in hit percentage. The top part shows statistical significance of effects of video length, where hit percentages between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where hit percentages are compared between the two traffic densities for each specific time (left), and for all times combined (right). Significant differences ($p < 0.01$) are indicated in boldface. Missing z values are caused by sample sizes being too small or lack different values.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	**< 0.001**	0.120	0.201	**< 0.001**	**0.007**	0.682	0.151	1	**0.001**
	z	**-3.535**	-1.556	1.279	**-**	**-2.6958**	-0.410	-1.436	0	**3.287**
	t	**13**	67	161	**8**	**18**	95	47.5	76.5	**243.5**
6 cars	***p***	0.640	0.093	0.102	0.126	0.972	0.126	**0.003**	**< 0.001**	**< 0.001**
	z	-0.468	-1.682	-1.635	-1.529	0.035	1.531	**3.020**	**3.780**	**4.033**
	t	210	130	69.5	107.5	127.5	234	**253**	**323**	**378**
Combined	***p***	0.023	0.057	0.744	**< 0.001**	0.040	0.264	0.102	**< 0.001**	**< 0.001**
	z	-2.279	-1.905	-0.326	**-3.464**	-2.050	1.117	1.634	**3.308**	**4.581**
	t	132	140	175.5	**51.5**	113.5	269	311.5	**393**	**481.5**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	0.085	**< 0.001**	**< 0.001**	0.020	**0.003**	0.780		**< 0.001**	
	z	1.722	**4.112**	**3.715**	2.330	**3.021**	0.279		**4.743**	
	t	335	**407**	**436.5**	324.5	**255**	135		**517.5**	

Average absolute error distance of hits did not show a clear effect of video length within scenarios, as can be seen in Figure 14. The expected effect of greater error distance was seen in 15 out of 24 scenarios, being statistically significant in scenario 17. No statistically significant ($p < 0.01$) differences were found between mirrored scenarios ($p = 0.527$, $t = -0.649$, $df = 14$ for 7A vs. 11B, and $p = 0.825$, $t = 0.225$, $df = 15$ for 8B vs. 9B). No clear effect of traffic density can be seen (scenarios 1–12 vs. scenarios 12–24).

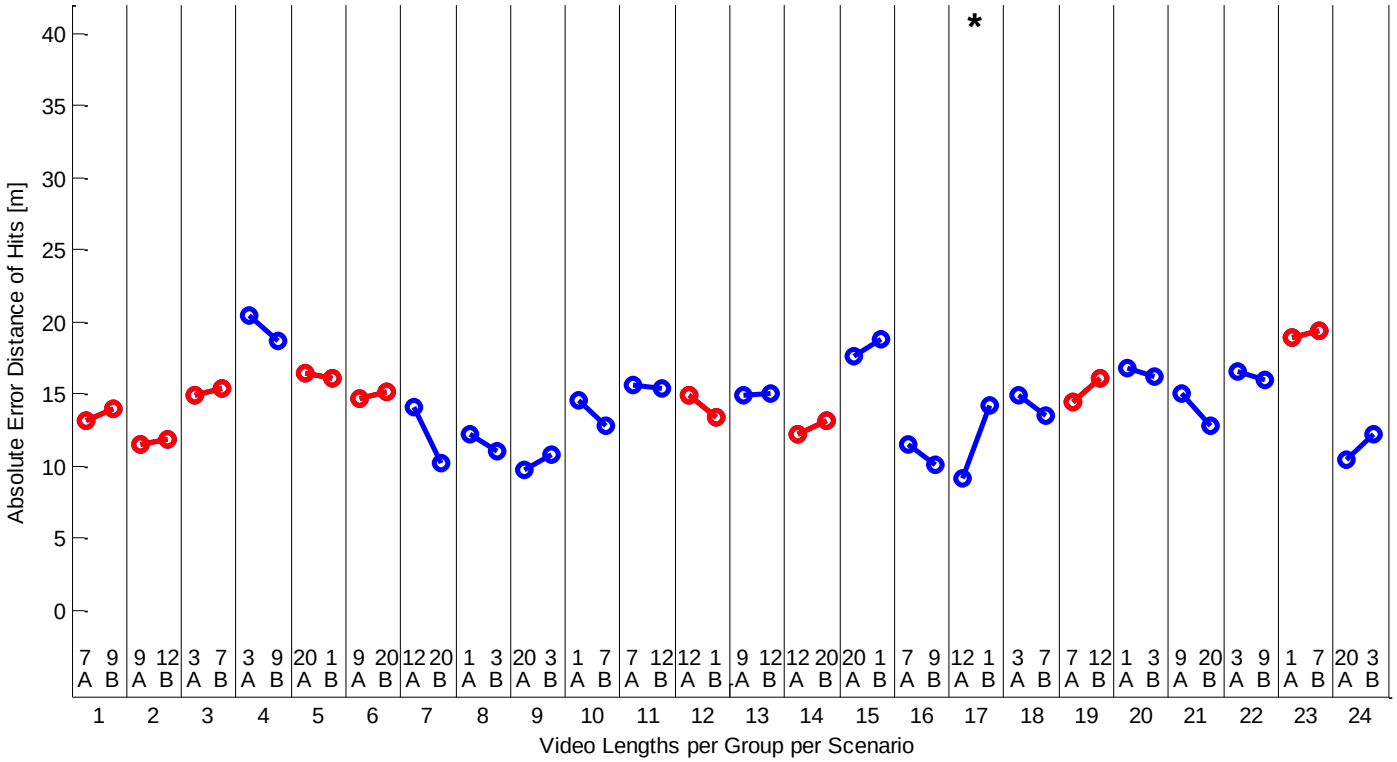


Figure 14: Average absolute error distance of hits as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error distance, where the colour of dots/lines indicates whether error distance was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error distance, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Absolute error distance of hits did not show statistically significant ($p < 0.01$) differences between adjacent video lengths, except between 1 vs. 3 seconds with a traffic density of four cars (Figure 15 and Table 8). When error distances for the videos of 20 s length were compared with those of shorter videos, the effect was statistically significant for the 20 vs. 1 second case with four surrounding vehicles. Traffic density did not have a clear effect on the average absolute error distance of hits. None of the effects of traffic density were statistically significant ($p < 0.01$).

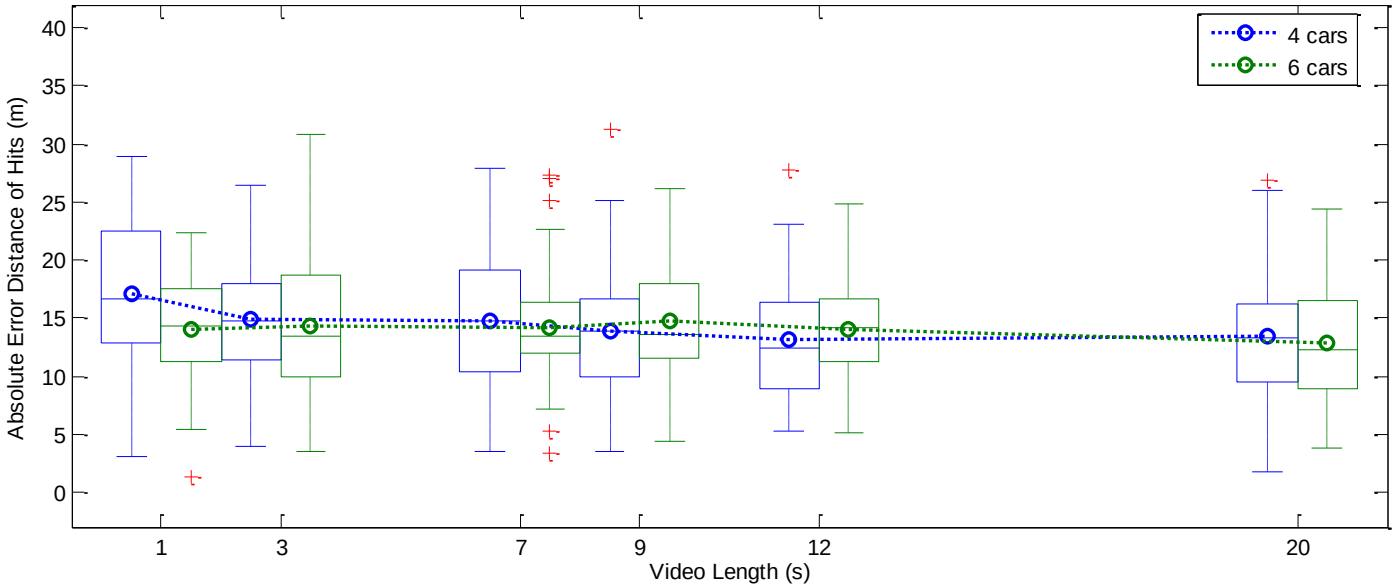


Figure 15: Average absolute error distance of hits as function of video length and traffic density. Boxplots represent error distances of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in error distance, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in error distance between traffic densities for a specific video length.

Table 8: Results of paired samples *t* tests for differences in average absolute error distance of hits. The top part shows statistical significance of effects of video length, where error distances between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where error distances are compared between the two traffic densities for each specific time (left), and for all times combined (right). Significant differences ($p < 0.01$) are indicated in boldface.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	0.018	0.853	0.549	0.341	0.744	0.479	0.185	0.089	**< 0.001**
	t	2.492	0.187	0.606	0.967	0.329	-0.716	-1.356	-1.756	**-4.173**
	df	31	31	31	31	31	31	31	31	**31**
6 cars	***p***	0.850	0.928	0.498	0.302	0.257	0.034	0.145	0.215	0.229
	t	-0.190	0.091	-0.656	1.051	-1.154	-2.216	-1.494	-1.266	-1.227
	df	31	31	31	31	31	31	31	31	31
Combined	***p***	0.161	0.877	0.855	0.174	0.603	0.052	0.062	0.050	**< 0.001**
	t	1.436	0.156	0.185	1.390	-0.526	-2.021	-1.938	-2.041	**-4.034**
	df	31	31	31	31	31	31	31	31	**31**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**			**Combined video lengths**
Traffic effects (4 – 6 cars)	***p***	0.010	0.489	0.553	0.435	0.402	0.502			0.139
	t	2.729	0.701	0.601	-0.792	-0.850	0.679			1.518
	df	31	31	31	31	31	31			31

Average absolute error distance of hits and misses decreased as video length increased (Figure 16), although this trend did not hold for scenarios 1, 9, 13, and 14. This decrease was statistically significant ($p < 0.01$) for scenarios 7, and 17. Error distance of hits and misses between mirrored scenarios of same length did not show statistically significant differences ($p = 0.972$, $t = 0.036$, $df = 14$ for 7A vs. 11B, and $p = 0.792$, $t = -0.268$, $df = 15$ for 8B vs. 9B).

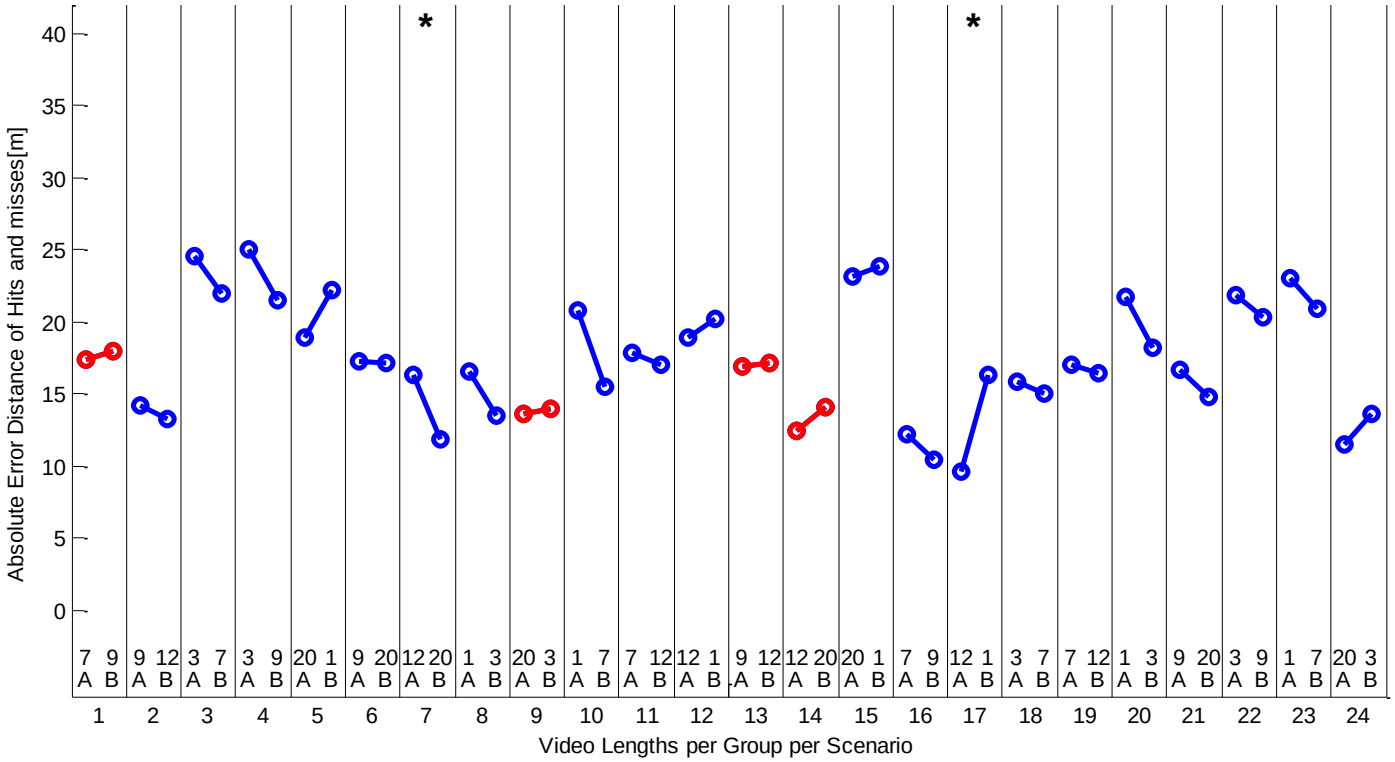


Figure 16: Average absolute error distance of hits and misses as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error distance, where the colour of dots/lines indicates whether error distance was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error distance, between video lengths within a scenario, are significant ($p < 0.01$).

Average absolute error distance of hits and misses decreased over time, except between 12 and 20 second video lengths for the four car traffic density (Figure 17). With a traffic density of six cars the total decline went from 19.9 to 14.4 meters. With a traffic density of four cars the total decline went from 21.2 to 15.9 meters.

When error distances between two adjacent video lengths were compared, a statistically significant ($p < 0.01$) decrease was only found in the 1 vs. 3 s case with a traffic density of four cars (
Table 9). When error distances for the longest video length were compared to those of shorter videos, statistical significance was found in the 20 vs. 1 second video length case with a traffic density of four cars. With six surrounding vehicles, it was found in the cases where the 20 s length was compared to the shortest three video lengths.

Traffic density had a no statistically significant effect for any of the six separate video lengths. When averaged over all video lengths, traffic density had a statistically significant ($p < 0.01$) negative effect (
Table 9).

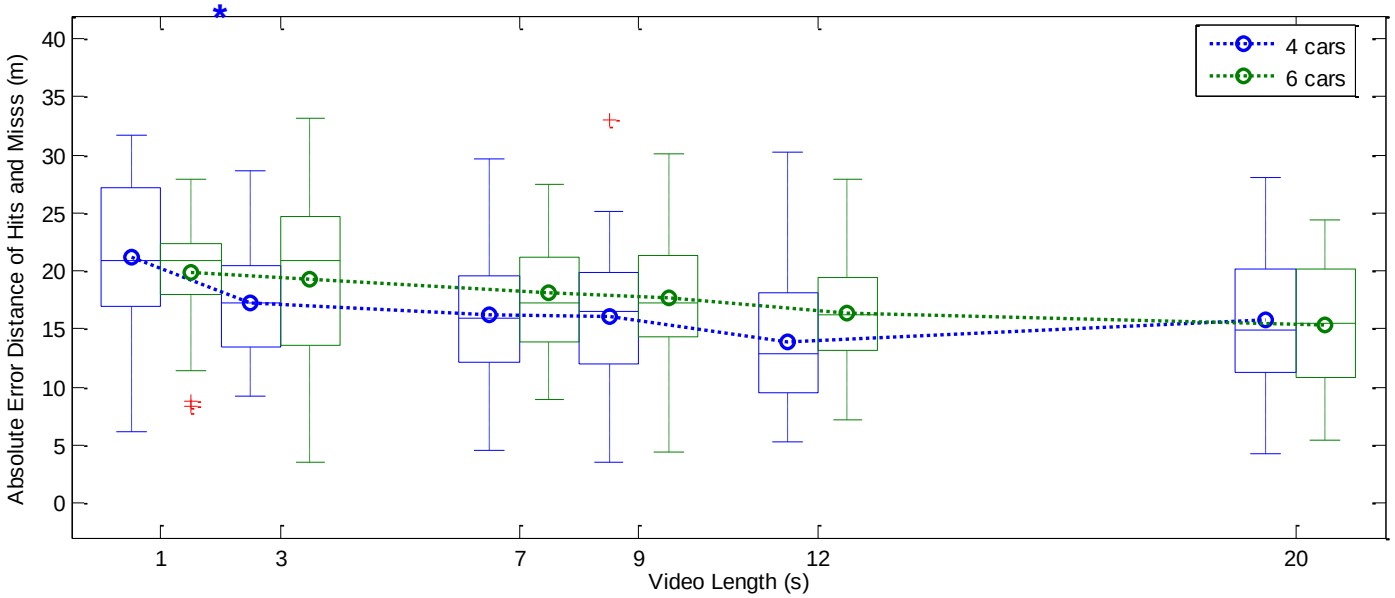


Figure 17: Average absolute error distance of hits and misses as function of video length and traffic density. Boxplots represent error distances of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in error distance, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in error distance between traffic densities for a specific video length.

Table 9: Results of paired samples *t* tests for differences in average absolute error distance of hits and misses. The top part shows statistical significance of effects of video length, where error distances between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where error distances are compared between the two traffic densities for each specific time (left), and for all times combined (right). Significant differences ($p < 0.01$) are indicated in boldface.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	**< 0.001**	0.311	0.884	0.029	0.097	0.718	0.602	0.078	**< 0.001**
	t	**5.722**	1.030	0.147	2.298	1.714	-0.364	-0.527	-1.821	**-6.821**
	df	**31**	31	31	31	31	31	31	31	**31**
6 cars	***p***	0.651	0.418	0.555	0.157	0.261	0.048	**0.004**	**0.004**	**< 0.001**
	t	0.457	0.821	0.597	1.449	-1.144	-2.056	**-3.074**	**-3.103**	**-4.378**
	df	31	31	31	31	31	31	**31**	**31**	**31**
Combined	***p***	**0.007**	0.324	0.498	**0.004**	0.462	0.060	0.017	**0.001**	**< 0.001**
	t	**2.873**	1.003	0.686	**3.131**	0.745	-1.953	-2.513	**-3.523**	**-7.905**
	df	**31**	31	31	**31**	31	31	31	**31**	**31**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**			**Combined video lengths**
Traffic effects (4 – 6 cars)	***p***	0.226	0.048	0.014	0.107	0.023	0.610			**0.003**
	t	1.236	-2.063	-2.618	-1.659	-2.396	0.515			**-3.245**
	df	31	31	31	31	31	31			**31**

Error percentage of indicated lanes was lower for scenarios with greater video length, although this trend did not hold for 7 out of 24 scenarios (Figure 18). The negative effect of video length within a scenario on the error percentage (i.e., better performance) was statistically significant ($p < 0.01$) for scenario 9. Error percentage between mirrored scenarios of same length did not show statistically significant ($p < 0.01$) differences ($p = 0.151$, $t = 24.5$ for scenario 7A vs. 11B, and $p = 1$, $t = 28$ for scenario 8B vs. 9B). The error percentage of indicated lanes was overall larger with more surrounding traffic (scenarios 1–12 vs. scenarios 13–24).

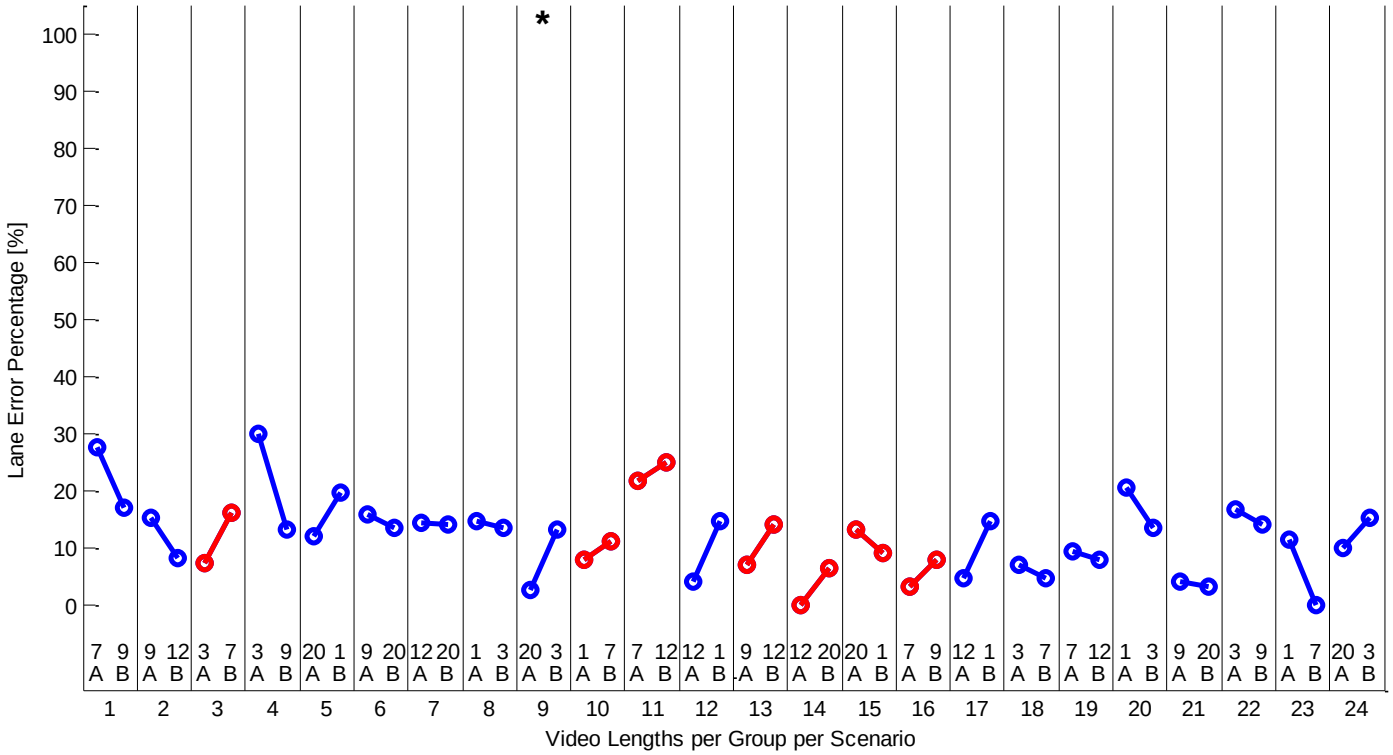


Figure 18: Error percentage of indicated lanes as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error percentage, where the colour of dots/lines indicates whether error percentage was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error percentage, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Different behaviour over video length in error percentage of indicated lanes was found for the two traffic densities (Figure 19). With the six car traffic density the error percentage shows a maximum at a video length of 7 seconds, while it shows a minimum with the four car traffic density at the same video length. When error percentages between two adjacent video lengths were compared, it was never statistically significant ($p < 0.01$). When error percentages were compared between the video length of 20 seconds and shorter videos, differences were also not statistically significant (Table 10), except in the 20 vs. 3 second case with combined traffic density. Traffic density had a statistically significant ($p < 0.01$) effect on error percentage (i.e., worse performance) for scenarios of 7 seconds long, and when all video lengths were combined (Table 10).

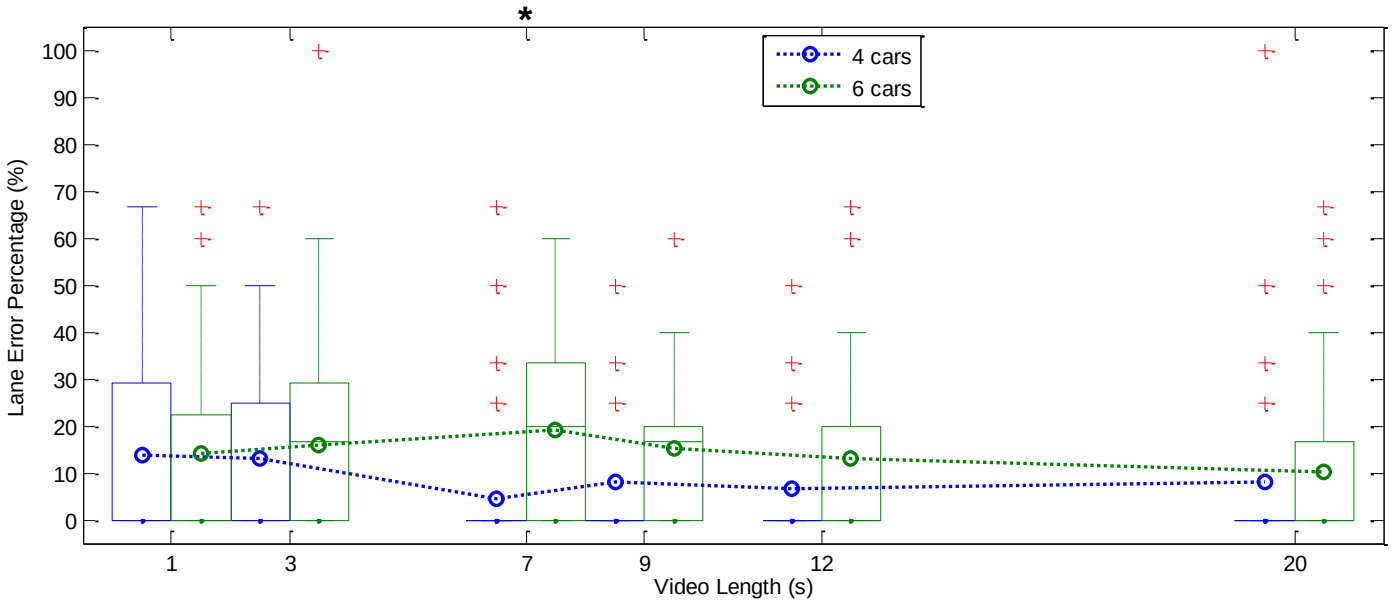


Figure 19: Error percentage of indicated lanes as function of video length and traffic density. Boxplots represent error percentages of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in error percentage, between two adjacent video lengths, are significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in error percentage between traffic densities for a specific video length.

Table 10: Results of Wilcoxon's signed rank tests for differences in error percentage of indicated lanes. The top part shows significance of effects of video length, where error percentages between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows significance of effects due to difference in traffic density, where error percentages are compared between the two traffic densities for each specific time (left), and for all times combined (right). Significant differences ($p < 0.01$) are indicated in boldface. Missing *z* values are caused by sample sizes being too small or lack different values.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**					
						20 – 12	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	0.696	0.014	0.190	0.668	0.815	0.935	0.241	0.127	0.052
	z	0.391	2.453	-1.312	-	0.235	-0.081	-	-1.526	-1.941
	t	138.5	218	43	68	72.5	93	81	64.5	60
6 cars	***p***	0.537	0.294	0.289	0.436	0.200	0.089	0.016	0.017	0.118
	z	-0.618	-1.049	1.061	0.780	-1.281	-1.703	-2.405	-2.386	-1.564
	t	216.5	169	266.5	253.5	147	108.5	89	74	114
Combined	***p***	0.801	0.102	0.969	0.411	0.726	0.256	0.252	**0.004**	0.025
	z	-0.253	1.637	0.039	0.822	-0.350	-1.135	-1.147	**-2.894**	-2.243
	t	250.5	331.5	250	255.5	215.5	165	164.5	**100.5**	104.5
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	0.840	0.458	**< 0.001**	0.014	0.082	0.377		**< 0.001**	
	z	-0.202	-0.742	**-3.925**	-2.446	-1.741	-0.883		**-3.871**	
	t	155	196.5	**9**	104.5	98	109		**57**	

Error percentage of indicated speeds was lower for longer videos of the same scenario, except for scenario 8 (Figure 20). The effect of video length within scenarios was statistically significant in 5 out of 24 scenarios. No clear effect could be seen when error percentages for the two different traffic densities were compared (scenarios 1–12 vs. scenarios 13–24). Error percentage between mirrored scenarios of same length did not show statistically significant ($p < 0.01$) differences ($p = 0.863$, t =25.5 for scenario 7A vs. 11B, and $p = 0.943$, $t = 61.5$ for scenario 8B vs. 9B).

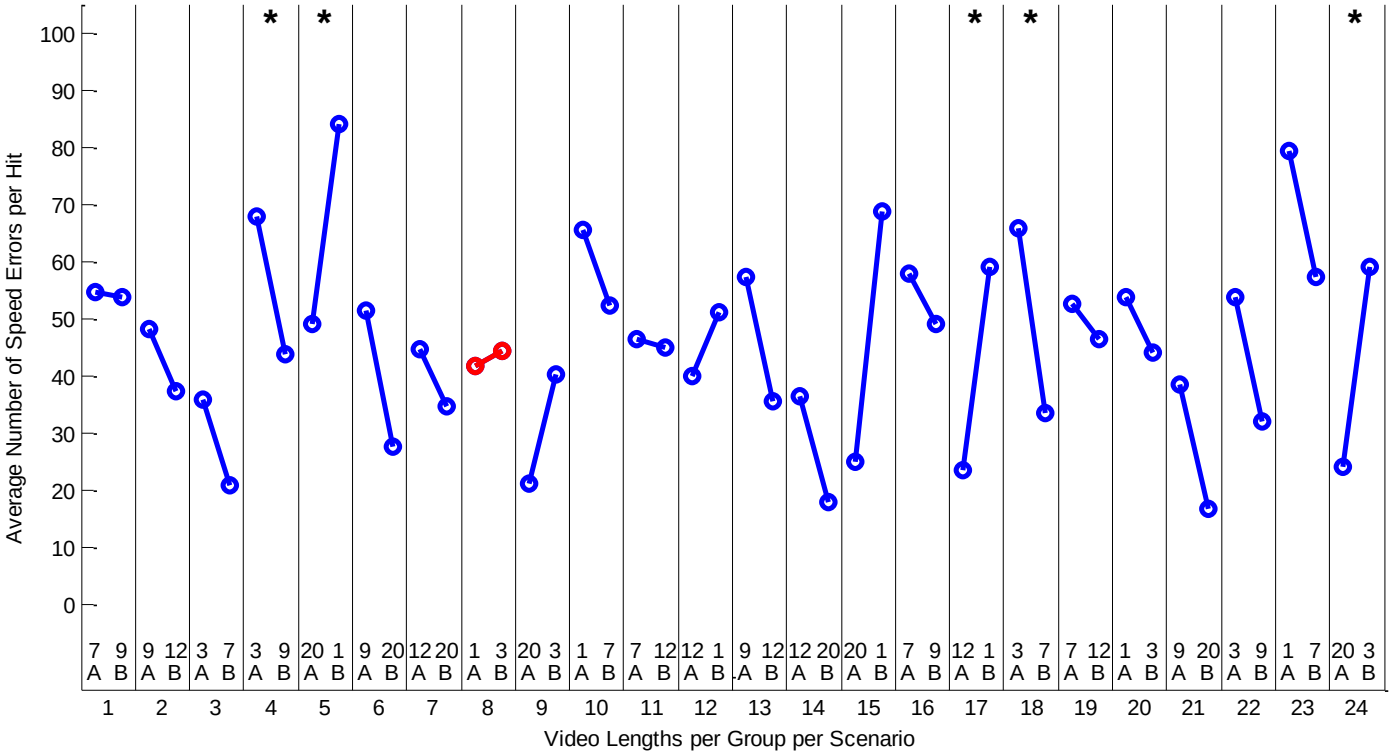


Figure 20: Error percentage of indicated speeds as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error percentage, where the colour of dots/lines indicates whether error percentage was higher for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error percentage, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Error percentage of indicated speeds decreased over video length for both traffic densities. This trend was approximately linear over video length with the traffic density of four cars, where it decreased from 65.1 to 20.8 percent. With six surrounding cars there was also a trend of decreasing over video length, but not monotonically. In the latter case it decreased from 60.5 % (1 s videos) to 33.1 % (20 s videos). When error percentages were compared between two adjacent video lengths, this was statistically significant ($p < 0.01$) for the 1 vs. 3 seconds video length with six surrounding cars, and the 12 vs. 20 seconds video length with four surrounding cars (
Table 11). When error percentages between the 20 second videos and shorter videos were compared, the effect was statistically significant ($p < 0.01$) in all cases with a traffic density of four cars. With a traffic density of six cars, this effect was statistically significant in the 20 vs. 9, 20 vs. 3, and the 20 vs. 1 second cases.

The effects of traffic density were not unilateral, where the error percentage of indicated speeds was higher for the lower traffic density for the shortest three video lengths, and vice versa for the longest three video lengths. The effect of traffic density was statistically significant ($p < 0.01$) only for videos of 20 seconds long. When averaged over all video lengths, the effect of traffic density was not statistically significant (
Table 11).

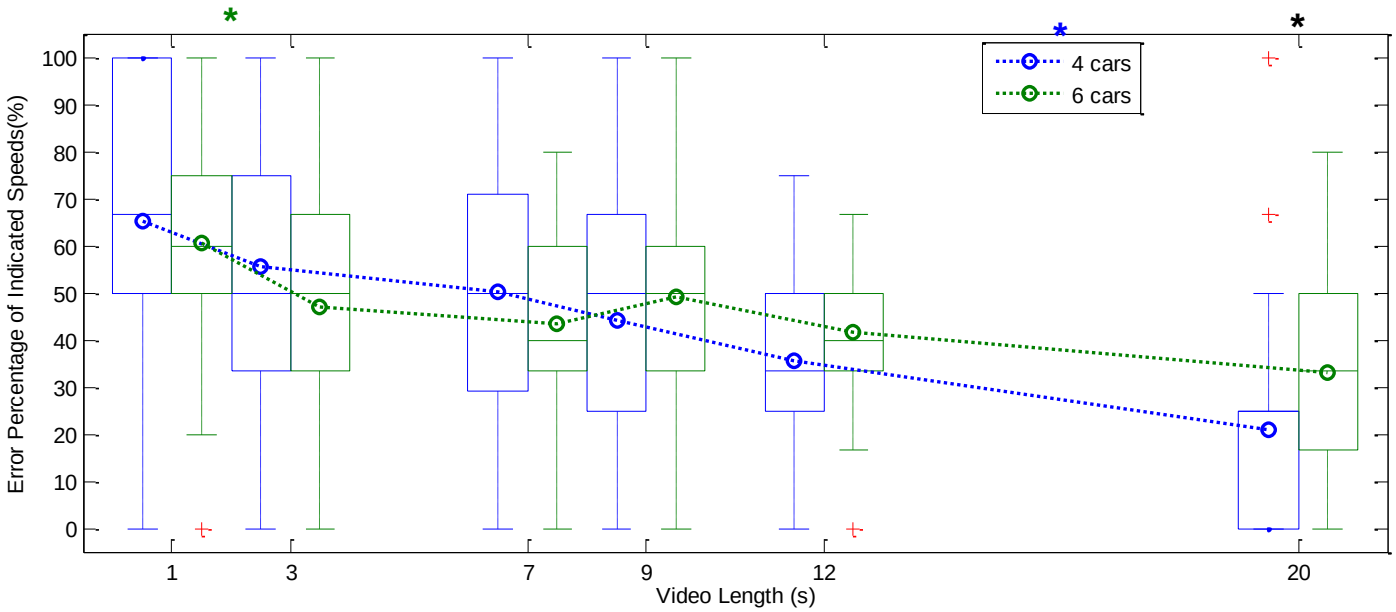


Figure 21: Error percentage of indicated speeds as function of video length and traffic density. Boxplots represent error percentages of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in error percentage, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in error percentage between traffic densities for a specific video length.

Table 11: Results of Wilcoxon's signed rank tests for differences in error percentage of indicated speeds. The top part shows statistical significance of effects of video length, where error percentages between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where error percentages are compared between the two traffic densities for each specific time (left), and for all times combined (right). Significant differences ($p < 0.01$) are indicated in boldface. Missing *z* values are caused by sample sizes being too small or lack different values.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	0.043	0.115	0.426	0.091	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	z	2.020	1.576	0.795	1.689	**-3.470**	**-3.663**	**-4.632**	**-4.728**	**-4.618**
	t	291.5	309	288.5	277	**58.5**	**61.5**	**8**	**7**	**12.5**
6 cars	***p***	**0.003**	0.339	0.070	0.067	0.015	**< 0.001**	0.013	**0.002**	**< 0.001**
	z	**2.927**	0.957	-1.814	1.832	-2.432	**-3.339**	-2.480	**-3.146**	**-4.274**
	t	**420.5**	279	155.5	321.5	134	**85.5**	112	**87.5**	**35.5**
Combined	***p***	**0.003**	0.079	0.845	0.032	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	z	**2.999**	1.759	-0.196	2.146	**-4.264**	**-4.311**	**-4.769**	**-4.841**	**-4.843**
	t	**401**	318	238	357.5	**36**	**28**	**9**	**1**	**5**
Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**	
Traffic effects (4 – 6 cars)	***p***	0.191	0.030	0.054	0.210	0.043	**0.005**		0.667	
	z	1.309	2.166	1.924	-1.255	-2.020	**-2.823**		-0.430	
	t	278	358.5	326	171.5	96	**104**		241	

Average size of errors in indicated speeds per hit was lower for longer videos of the same traffic scenario, except for scenarios 1 and 8 (Figure 22). This effect was statistically significant ($p < 0.01$) for 6 out of 24 scenarios. Average error size between mirrored scenarios of same length did not show statistically significant differences ($p = 0.898$, $t = 35$ for scenario 7A vs. 11B, and $p = 1$, $t = 45.5$ for scenario 8B vs. 9B). Traffic density did not show a clear trend.

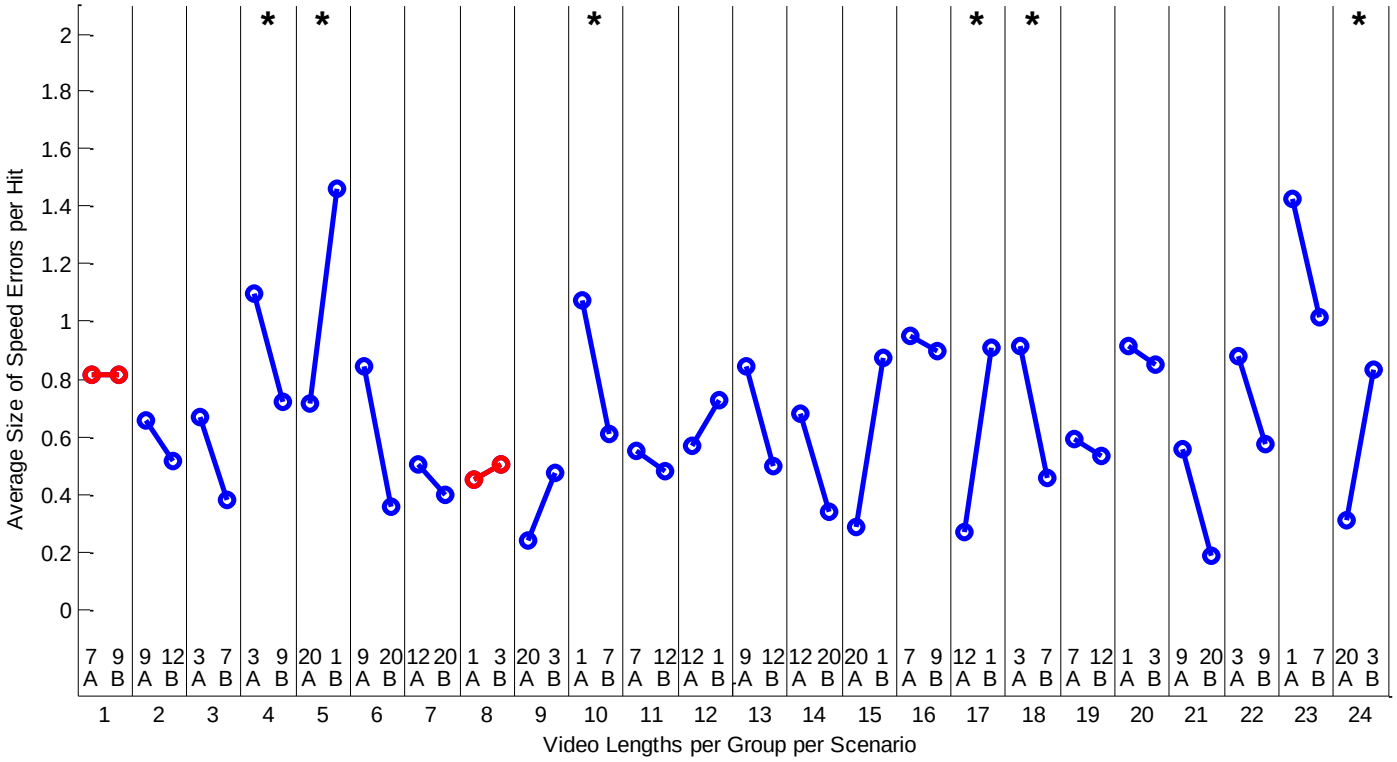


Figure 22: Average size of speed errors per hit as function of video length, group, and scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Dots indicate the mean error size per hit, where the colour of dots/lines indicates whether error size per hit was lower for the shorter video length (blue) or vice versa (red). Stars (*) indicate that differences in error size per hit, between video lengths within a scenario, are statistically significant ($p < 0.01$).

Average size of speed errors per hit showed a trend of decreasing over video length for both traffic densities (Figure 23). For scenarios containing four cars of surrounding traffic, this decrease was steady over video length, starting at 1.03, and ending at 0.28 steps per hit. Scenarios containing six cars, however did not have a steady decrease, showing a statistically significant increase between the 7 and 9 s marks. Overall it decreased from 0.93 (1s videos) to 0.43 steps (20 svideos). When average size of speed errors per hit was compared between videos of 20 seconds and shorter videos, it was found that video length had a statistically significant ($p < 0.01$) negative effect in all but one case (20 vs. 12 seconds with 6 surrounding cars), see

Table 12. Traffic density did not show a clear trend: Average size of speed errors per hit was higher with the lower traffic density for the shortest three video lengths, and lower for the longest three video lengths.The effect of traffic density was statistically significant ($p < 0.01$) for scenarios with a length of 7 seconds, but not when video lengths were combined (

Table 12).

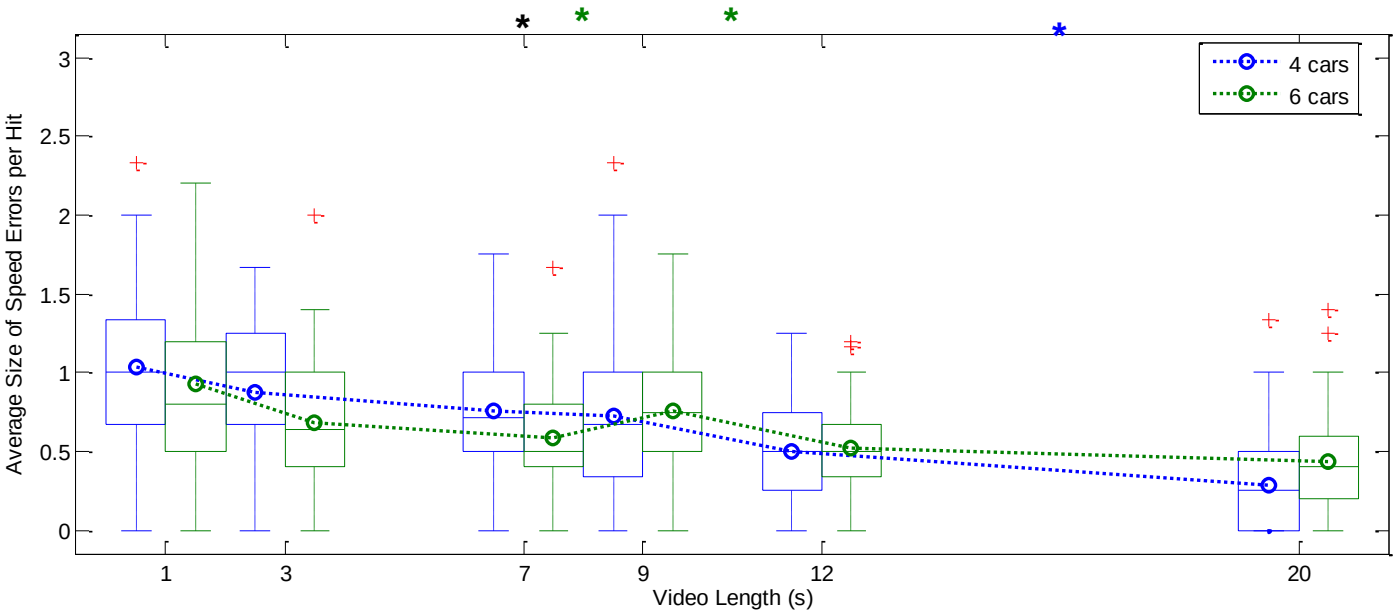


Figure 23: Average size of speed errors per hit as function of video length and traffic density. Boxplots represent average error sizes of scenarios with certain video lengths, where bars indicate medians, and boxes indicate the 25th and 75th percentiles. Blue and green colours indicate traffic densities of four and six cars, respectively. Dots indicate means and dotted lines show trends over video length. Blue and green stars (*) indicate that differences in average error size, between two adjacent video lengths, are statistically significant ($p < 0.01$) for traffic densities of four and six cars, respectively. Black stars indicate a statistically significant difference ($p < 0.01$) in average error size between traffic densities for a specific video length.

Table 12: Results of Wilcoxon's signed rank tests for differences in average size of speed errors per hit. The top part shows statistical significance of effects of video length, where average error sizes between two specific video lengths are compared, both for separate traffic densities as well as for combined traffic densities. The bottom part shows statistical significance of effects due to difference in traffic density, where average error sizes are compared between the two traffic densities for each specific time (left), and for all times combined (right). Statistically significant differences ($p < 0.01$) are indicated in boldface. Missing *z* values are caused by sample sizes being too small or lack different values.

Specific video lengths (s) comparisons		**1 – 3**	**3 – 7**	**7 – 9**	**9 – 12**	**20 – 12**	**20 – 9**	**20 – 7**	**20 – 3**	**20 – 1**
4 cars	***p***	0.035	0.100	0.993	0.024	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	z	2.111	1.645	0.009	2.265	**-3.773**	**-4.132**	**-4.562**	**-4.800**	**-4.847**
	t	335	293.5	264.5	385	**43.5**	**32**	**11**	**7.5**	**5**
6 cars	***p***	0.013	0.090	**0.005**	**< 0.001**	0.098	**< 0.001**	**0.005**	**< 0.001**	**< 0.001**
	z	2.487	1.695	**-2.815**	**3.773**	-1.657	**-4.030**	**-2.796**	**-3.430**	**-4.432**
	t	397	334.5	**113.5**	**440.5**	163.5	**48.5**	**114.5**	**73**	**27**
Combined	***p***	**0.001**	0.052	0.336	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
	z	**3.207**	1.945	-0.963	**3.404**	**-3.955**	**-4.741**	**-4.731**	**-4.881**	**-4.937**
	t	**435.5**	368	212.5	**446**	**52.5**	**10.5**	**11**	**3**	**0**

Video length (s)		**1**	**3**	**7**	**9**	**12**	**20**		**Combined video lengths**
Traffic effects (4 – 6 cars)	***p***	0.196	0.018	**0.008**	0.550	0.715	0.012		0.161
	z	1.294	2.376	**2.654**	-0.598	-0.365	-2.506		1.403
	t	314	391	**361.5**	217.5	244.5	130		339

Discussion/Conclusion

The aim of this research was to find out whether the method used by Gugerty (1997) [17] is suited for measuring how much time people need to understand traffic situations. For this purpose, a similar method was used, where participants watched videos of simulated traffic scenarios, whereafter they were asked to reproduce the traffic scenario in a top-down view, and to rate their agreement with two claims ("The task was difficult" and "I had sufficient time to perform the task"). Thus, the effects of six different video lengths (i.e., preparation times from 1 to 20 s) and two traffic densities (4 vs. 6 cars) were assessed. The main results are discussed below.

Subjective Measures

Difficulty rating was lower for longer videos in all but one scenario, and significantly lower in 4 out of 24 scenarios (Figure 6). Furthermore, for both traffic densities, difficulty rating showed a similar diminishing trend of decreasing over video length (Figure 7), which indicates that the task did not get easier when more time was given, after a certain time, and that traffic density had no clear effect on this time. Traffic density had a clear, and statistically significant (Table 4), effect on self-reported difficulty rating, which. This shows that performing the task of reproducing a traffic scenario in a top-down view was deemed less difficult when there was more time to assess the traffic situation, and when the traffic situation had less surrounding traffic.

Time rating **was higher for longer videos in all scenarios, and significantly so in 14 out of 24 scenarios (Figure 8). The self-reported preparation time rating showed a strong increase over video length at first, which seemed to diminish for the longer video lengths (Figure 9), which might indicate that a certain amount of time is deemed 'enough' time to prepare for the reconstruction of a scene. The diminishing effect was similar for both traffic densities. Comparisons between two adjacent video lengths were significant in most cases (**
Table 5), which makes this measure more sensitive to effects of time than the difficulty rating. Traffic density had a statistically significant ($p < 0.01$) negative effect on time rating in all cases. These findings show that people agreed more with the notion of having enough time to perform the task of reproducing a traffic scenario in a top-down view when more time was available to assess that situation, and when the situation contained less traffic.

Basic score

Error percentage in number of placed cars **was lower in the longer version of a scenario in most of the 24 scenarios (Figure 10). This decrease over video length within scenarios was, however, not statistically significant ($p < 0.01$) in most scenarios. When compared between adjacent video lengths (Figure 11), it was found that there is a clear decreasing trend over video length with both traffic densities, but the measure (mostly) lacks the sensitivity to show statistically significant ($p < 0.01$) differences between adjacent video lengths. This is especially the case when error percentages were compared between the longest three videos, with the lower traffic density, which suggests that a saturation level of SA might exist, and that it is reached earlier with the lower traffic density. Error percentage in number of placed cars was significantly higher (**
Table 6) with more surrounding vehicles for all but one video length (1 s), and when averaged over all video lengths, which supports the hypothesis that performance worsens with higher traffic density.

Advanced scores

Hit percentage **was higher when participants got more time to look at the same scenario (Figure 12), but not in all cases. The difference was statistically significant ($p < 0.01$) in only one scenario. Hit percentage showed an increasing trend over video length for both traffic densities (Figure 13). Differences due to traffic density were statistically significant in three out of six video lengths, and when averaged over all video lengths (**
Table 7). The way of increasing differed between the two traffic densities: For six cars of surrounding traffic hit percentage showed a monotonical, diminishing increase over video length. For four cars, however, there was a peak at the 12 second mark, after which the hit percentage deceased significantly. In the latter case, hit percentage at the 20 s mark was comparable to that at the 9 s mark. These differences in behaviour between the two traffic densities can have several explanations, which are further elaborated upon below. Averaged over all video lengths, hit percentage increased from 84 to 91 percent between six and four car traffic densities, which is similar to that found by Gugerty [17], who found an increase from approximately 85 to 94 percent.

Average absolute error distance of hits **did not show a clear effect of video length or traffic density (Figure 14 and Figure 15), where most effects were not statistically significant (**
Table 8). The data suggests that people did not tend to position cars better or worse when they had more time to assess a situation, or when the situation had more traffic. This could be caused by the fact that this measure only takes 'hits' into account: If people choose to reproduce only part of the traffic scene, or only

have seen or remember a portion of the traffic, it could be they reproduce the positions of this portion of cars at a similar level of performance. Not taking misses into account (which are more numerous for shorter videos and higher traffic density) may thus give an incomplete picture of a person's performance on reconstruction of a traffic scene.

***Average absolute error distance of hits and misses* did show better performance on longer scenarios in most cases (Figure 16). Differences between traffic densities was significant when averaged over all video lengths (**
Table 9), and a decreasing trend was seen for the error distance of hits and misses for both traffic densities (Figure 17). As with the hit percentage, the behaviour of this trend differs between both traffic densities. This difference can be partially explained by the measure's dependency on the hit percentage or, specifically, the 'miss percentage' (i.e., the reciprocal of the hit percentage), because for each miss, a constant value was added to the error distance, which means that when the average absolute error distance of hits does not show great variety, the average absolute error distance of hits and misses shows behaviour in accordance with the hit percentage. Both these measures could possibly be improved upon by using weighted errors, as is discussed below.

***Error percentage of indicated lanes* was only slightly in favour of the hypothesis that performance increases for longer videos (Figure 18), but the effect was absent in almost a third of the scenarios, and the effect was only statistically significant in one scenario. Traffic density did have a negative effect on the error percentage of indicated lanes, which was significant for one separate video length, and when averaged over all video lengths (**
Table 10). No clear trend was discernible when error percentages were compared between video lengths (Figure 19). The apparent lack of sensitivity to differences in video length, and the overall low level of the error percentage, might indicate that people do not make many mistakes in reproducing the lane in which they have last seen a car. Furthermore, the way the matching algorithm works, which favours matches within the same lane to those in other lanes, could explain part of the lack of sensitivity of this measure.

Error percentage of indicated speeds was lower for longer versions of the same scenario in all but one scenario, and significantly so in 5 out of 24 scenarios (Figure 20), which supports the hypothesis that performance is higher when more time is available. This trend of decreasing over time was also seen when comparing error percentages for videos of 20 s length with shorter video lengths, for both traffic densities (Table 11). Interesting to see is that the error percentage of indicated speeds was quite high for all video lengths, which suggests that it was difficult to determine relative speeds. Another interesting feature is the difference in how error percentage in indicated speeds decreased between the two traffic densities: The error percentage kept decreasing over video length in the cases where there were four surrounding cars, although this trend was not visible for the six car cases, where an in-between peak can be seen. Furthermore, the error percentage was higher for the three shortest video lengths, and lower for the longest three video lengths in the four car cases, which made this measure inconclusive about effects of traffic density. Part of this unexpected behaviour might be caused by the measure's sensitivity to mismatches (Appendix A3-2), and possibly also to the hit percentage (i.e., only errors of hits have been taken into account). Combining it with a standard error for misses, and averaging over hits and misses might give a clearer picture of the total error made in indicated speeds. With both traffic densities, the development of the error percentage over video length did not suggest that a saturation level of SA had been reached. This could be partly explained by the fact that it takes time to perceive the (relative) speed of an object, because it can only be deduced from changes in the scene (i.e., changing of the relative distance), which requires tracking of a vehicle [24]. This tracking can only occur after the vehicles are identified, which means that people first need time to scan the environment, and then need time for integrating the speed of the surrounding vehicles. When the changes are smaller (i.e., smaller speed difference and/or greater distance), they are more difficult to perceive, arguably increasing the time needed to accurately perceive these changes. Assessing the speed of all cars in the scenarios used in this study might thus take even more time than 20 s. In a real-life situation, however, it is not always necessary to know the exact speeds of all surrounding cars, where cars with a low relative speed are less directly important

Average size of errors in indicated speeds per hit was lower for longer scenarios in all but two scenarios, and significantly lower in six out of 24 scenarios (Figure 22). This measure showed the same general trends as the error percentage of indicated speeds regarding comparisons between specific video lengths and both traffic densities (Figure 23 and

Table 12). The possible causes are also the same (i.e., sensitivity to mismatches (Appendix A3-2) and hit percentage). This measure therefore does not suggest that a saturation level of SA was reached, and is

inconclusive about effects of traffic density. In order to make this measure more insightful it might be better to average it over the number of errors in indicated speeds instead of over the number of hits, making it less sensitive to the hit percentage.

Various advanced scores, and the basic score, point out a difference in SA development as traffic becomes denser (e.g., hit percentage, Figure 13, showing a monotonical, diminishing increase over video length with six cars, and an in-between peak with four cars). This difference in behaviour can have several explanations:

1) The scenarios with four cars are too 'easy', making people
    - Lose interest after a certain time
    - Filter out/forget 'less urgent' traffic (e.g., non-threatening / far away)
    - Complacent, settling for a lower level of SA (Endsley and Kiris (1995) [25])
2) A saturation level exists, but only when a scenario is 'difficult enough'
3) A saturation level exists, but is not maintained, which was not visible for the higher traffic density.
4) With four surrounding cars a saturation level is reached earlier than twelve seconds, but SA has a peak which was not seen with six surrounding cars.
5) Scanning behaviour of people in assessing traffic situations differs for the two traffic densities.
6) The saturation level of SA, with a traffic density of four cars, is reached earlier (e.g., between seven and nine seconds) and the apparent drop in SA does not exist.
7) The way of processing the data influences scores: in scenarios with six cars of surrounding traffic, there is a higher chance that a randomly placed car is matched than when there are four cars of surrounding traffic.

Overall, it can be concluded that main trends within the different scores support the hypothesis that performance improved when more time is available. For some of the scores (i.e., basic score, hit percentage, and average absolute error distance of hits and misses), however, more time does not necessarily mean better performance, which could indicate that there is a saturation level of SA people can attain, but there might also be floor or ceiling effects within the different measures (e.g., the basic score approaching 0, and the limited number of possible outcomes for the different 'percentage' scores). Furthermore, depending on the measure, traffic density either has a clear negative effect on performance, or the measure is inconclusive about the effects of traffic density, which suggests that traffic density does have an effect on performance, but that some measures are less suited to find these effects (i.e. average absolute error distance of hits, and both speed measures).

Implications and further research

Gugerty's method [17], where participants are asked to reproduce a traffic scenario in a top-down view, has been found to be an adequate method for determining differences in SA for different video lengths (reaction times) and traffic scenarios. Although the method used in this study seems suited for its purpose, it was not yet able to find the exact amount of time needed to assess traffic situations, as this seems to be at least partially dependent on the number of surrounding cars. Therefore, further research is needed to establish the effects of different traffic densities in more detail.

Although the results of this study may not be directly suited to real-life use, as used traffic scenarios were not extensive enough, more elaborate traffic scenes, or a driving simulator could be used to mimic real-life situations more closely. Furthermore, the participants were only asked to assess the traffic situation and not to perform any driving related decisions or actions, which would normally be performed (partially) simultaneously with the assessment of the traffic situation (e.g., step 1) and 3) of a ToC). This might have consequences for the time needed to assess the situation, as well as influence behaviour in assessing the situation (e.g., certain cars that would be entitled as "not directly important" are not focussed upon in real life, but are focussed upon in this task). It would be possible to ask participants to do ToCs and to use the method applied in this study when using a driving simulator. In this way, driving performance after a ToC can be compared to SA scores derived from this method. This suggested method could be suitable for assessing any ToC-protocol (i.e., with any preparation time, any order of steps to take, and between any two automation modes). When comparing the results from a driving simulator study with those using videos of traffic scenarios, it is important to have a clear starting point of when the assessment of the traffic situation begins and ends. In case of videos it is the start and end of the scenarios, when using a driving simulator it could be the first gaze through any window and the moment a participant has finished the ToC,

as was used in [9 10 11], where it is important to distract a participant to such an extent that it can be assumed (s)he was completely out-of-the-loop before the ToC was initiated.

Taking the differences in SA scores between the two traffic densities into account, a minimal time given to or imposed on the driver for assessing a traffic situation becomes more important when a traffic situation contains more surrounding cars.

Further research should be performed to establish whether a saturation level for SA exists or not. If it exists, people need to get sufficient time to be able to reach this level. If it does not exist, some other estimation of when it is safe(st) to return control to a human driver should be established. If this saturation level exists, and is indeed reached faster when there is less surrounding traffic, the time given to/imposed on the driver to assess traffic situations could be made dependent on how many cars are surrounding the ego-vehicle. Doing so could also prevent complications with the driver (e.g. annoyance at the system being too slow), which in turn might lead to a drop in SA. If the level of SA diminishes after the saturation level is reached, the time at which this peaks in level of SA are reached could arguably be the safest times to hand over control to the driver.

Although both basic and advanced scores, with the exception of error distance of hits, seem to be adequate for their purpose of showing effects of video length and/or traffic density, some other effects that could be of influence on these scores have not been taken into consideration:

- *Differences between scenarios.* In all performance scores, and in the self-rated scores, differences can be found between scenarios, even if they have the same video length and the same traffic density. Although it was attempted to keep the number of variables small (e.g., 5 constant speeds, limited range, visibility criterion), characteristics and behaviour of surrounding traffic might still influence visibility and the driver's perception of surrounding traffic. For example, noticing a car that is further away, much less interpreting its behaviour, becomes increasingly harder as distance increases. Gugerty (1997) [17] corrected his calculated absolute error distances by comparing it to an 'expected' absolute error distance, derived from a linear regression on all the absolute error distances in relation to the distances of actual cars, for he found that the absolute error distance increased with the distance of actual cars. Although a correction of error distance scores was not part of this study, a regression analysis was performed on the differences in (non-absolute) error in distance of placed cars as effect of distance of the actual car they were matched with (see Appendix A5: Regression analysis). It was found that the distance of cars that were farther away was underestimated (i.e., cars were placed too close to the ego-vehicle), as well as the distance of closer cars being overestimated. Incorporating this regression analysis into the error distance scores (e.g., by checking how far-off the measured error distance of a match is from an 'expected error distance'), comparisons between the different scenarios could become more compatible. This type of scoring system could also be implemented, for example by scoring missed vehicles directly in front of the ego-vehicle as a 'bigger' error than missed cars further away. A similar strategy could be used on (size of) errors in indicated speeds, for it is more difficult to see the changes in distance, caused by a relative speed difference, when a car is far away, and when the relative speed difference is small.

- *Errors made by the matching algorithm.* The matching algorithm has a threshold distance of 38 meters for considering placed cars a match. In some cases, this threshold was too tight, resulting in either a false miss, or a false match. In case of a false miss, the error made by the matching algorithm can be kept small by incorporating errors for misses in the scores (e.g., average error distance of hits and misses). In case of a false match, the error made by the matching algorithm might have some effects on scores that are calculated for each hit and/or are divided by the number of hits (Appendix A3: Matching algorithm examples). These effects can be minimized by making the algorithm more reliable, or can be averaged out by working with larger numbers of trials and/or participants. The algorithm could be made more reliable by increasing or not using the threshold distance, but doing so will have effect on the calculation of a 'best match' regarding the error of lanes. Adding indicated speed as extra matching variable could increase reliability as well.

- *Differences between the two groups of participants*. Even though all participants had a valid driving license, and the two groups were similar regarding age and sex, other differences might have existed. It might be that these other differences had some influence on scores compared within scenarios. Using larger groups or a different research design could minimize these effects.

Next to expanding on the used method, it can be improved upon as well. During reproduction of scenarios, time spent on placing cars and indicating their speeds could become so much, participants might have started to forget or confuse some of the features they saw (e.g., their speed). One improvement would be to replace the slider bars with a system where participants can click or press where they want to place cars and drag them directly with the use of a mouse or by touch. Furthermore, if a greater range of speeds is used for the surrounding traffic, going to a more continuous way to indicate relative speeds (e.g., using a slider bar for indicating the magnitude of the relative speed) could provide more information on perception of speed.

In conclusion, the used desktop-based method, which was based upon the method Gugerty (1997) [17] used, may lack the realism needed for direct application of the results. The method, however, does seem suited for studying the preparation time if implemented in more immersive experimental designs.

References

1. Gasser, T. M., & Westhoff, D. (2012). BASt-study: Definitions of automation and legal issues in Germany. *Paper presented at the 2012 Road Vehicle Automation Workshop*, Irvine, CA. Retrieved from http://onlinepubs.trb.org/onlinepubs/conferences/2012/Automation/presentations/Gasser.pdf
2. www.**bmw**.com/com/en/newvehicles/x/x5/.../**traffic_jam_assistant**.html
3. http://singularityhub.com/2013/03/21/bmw-cars-will-be-highly-automated-by-2020-driverless -by-2025-behind-the-curve/
4. http://www.youtube.com/watch?v=HUU7GgBMmY
5. Benz, T. et al. (2003). Traffic Effects of Driver Assistance Systems – The Approach within INVENT.
6. N.A. Stanton, & P. Marsden (1996). From Fly-By-Wire to Drive-By-Wire: Safety Implications of Automation in Vehicles.
7. Kaber, D.B., & Endsley, M.R. (1997). Out-of-the-Loop Performance Problems and the Use of Intermediate Levels of Automation for Improved Control System Functioning and Safety.
8. Endsley, M.R. (1995). Toward a theory of situation awareness in dynamic-systems. *Human Factors 37 (1)*, pp. 32-64.
9. Damböck, D. (2013). Automationseffekte im Fahrzeuf – von der Reaktion zur Übernahme.
10. Gold, C., Damböck, D., Lorenz, L., Bengler, K. (2013). ”Take over!” How long does it take to get the driver back into the loop?. *Proceedings of the Human Factors and Ergonomic Society 57th Annual Meeting*, 2013, pp. 1938-1942
11. Petermann-Stock, I., Hackenberg, L., Muhr, T. Mergl, Ch. (2013). Wie lange braucht der Fahrer? – Eine Analyse zu Übernahmezeiten aus verschiedenen Nebentätigkeiten während einer hochautomatisierten Staufahrt.
12. Van den Beukel, A.P., and Van der Voort, M.C. (2013). The Influence of Time-criticality on Situation Awareness when Retrieving Human Control after Automated Driving.
13. Taylor, R.M. (1990). Situational Awareness Rating Technique (SART): the development of a tool for aircrew systems design (AGARD-CP-478) pp3/1 -3/17. *Situational Awareness in Aerospace Operations. NATO-AGARD*, Neuilly Sur Seine, France.
14. Naujoks, F., Mai, C., & Neukum, A. (2014). The effect of urgency of take-over requests during highly automated driving under distraction conditions. In N. Stanton, S. Landry, G. Di Bucchianico, & A. Vallicelli (Eds.), *Advances in Human Aspects of Transportation: Part I*. AHFE Conference. pp. 431–438
15. Merat, N., Jamson, A. H., Lai, F. C., Daly, M., & Carsten, O. M. (2014). Transition to manual: Driver behaviour when resuming control from a highly automated vehicle. *Transportation Research Part F: Traffic Psychology and Behaviour*, 27, 274–282.
16. Mok, B., Johns, M., Lee, K. J., Miller, D., Sirkin, D., Ive, P., & Ju, W (2015). Emergency, automation off: Unstructured transition timing for distracted drivers of automated vehicles. *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation*.
17. Gugerty, L.J. (1997). Situation Awareness During Driving: Explicit and Implicit Knowledge in Dynamic Spatial Memory. *Journal of Experimental Psychology: Applied*, 1997, Vol. 3, No. 1, pp. 42-66.

18. Pylyshyn, Z., & Storm, R. (1988). Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, 3, pp. 179-197.
19. http://smarteye.se/products/dr120/
20. www.mathworks.com/products/**matlab/**
21. https://www.tassinternational.com/prescan
22. http://www.videosoftdev.com/free-video-editor
23. http://windows.microsoft.com/en-us/windows7/windows-media-player-12
24. Kwon, O., Tadin, D., & Knill, D.C. (2015). Unifying account of visual motion and position perception. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 26, pp. 8142-8147.
25. Endsley, M.R., and Kiris, E.O. (1995). The out-of-the-loop performance problem and level of control in automation. *Human Factors*, 37, pp 381-394.

Appendices

Appendix A: Material regarding the thesis report

Appendix A1: Forms

Appendix A1-1: Informed Consent Form for Participants

This appendix contains the form which allowed participants to provide written informed consent. It describes, inter alia, the purpose and procedures of the experiment.

Research title: "How much time do people need to understand traffic situations?: A desktop simulation study"

Researchers:
Xander Coster – MSc student
Email: x.a.coster@student.tudelft.nl

Dr.ir. Joost C.F. de Winter – Supervisor
Email: j.c.f.dewinter@tudelft.nl

Zhenji Lu, MSc – Supervisor
Email: Z.Lu@tudelft.nl

Location of the experiment:
Driving simulator lab; room 34 G-0-210
Faculty of Mechanical, Maritime and Materials Engineering
Delft University of Technology
Mekelweg 2, 2628 CD Delft

Introduction: Please read this consent document carefully before you decide to participate. This document describes the purpose, procedures, risks, and discomforts of this study. Your signature is required for participation. You have the right to withdraw at any time. If you desire a copy of this consent form, you may request one and we will provide it.

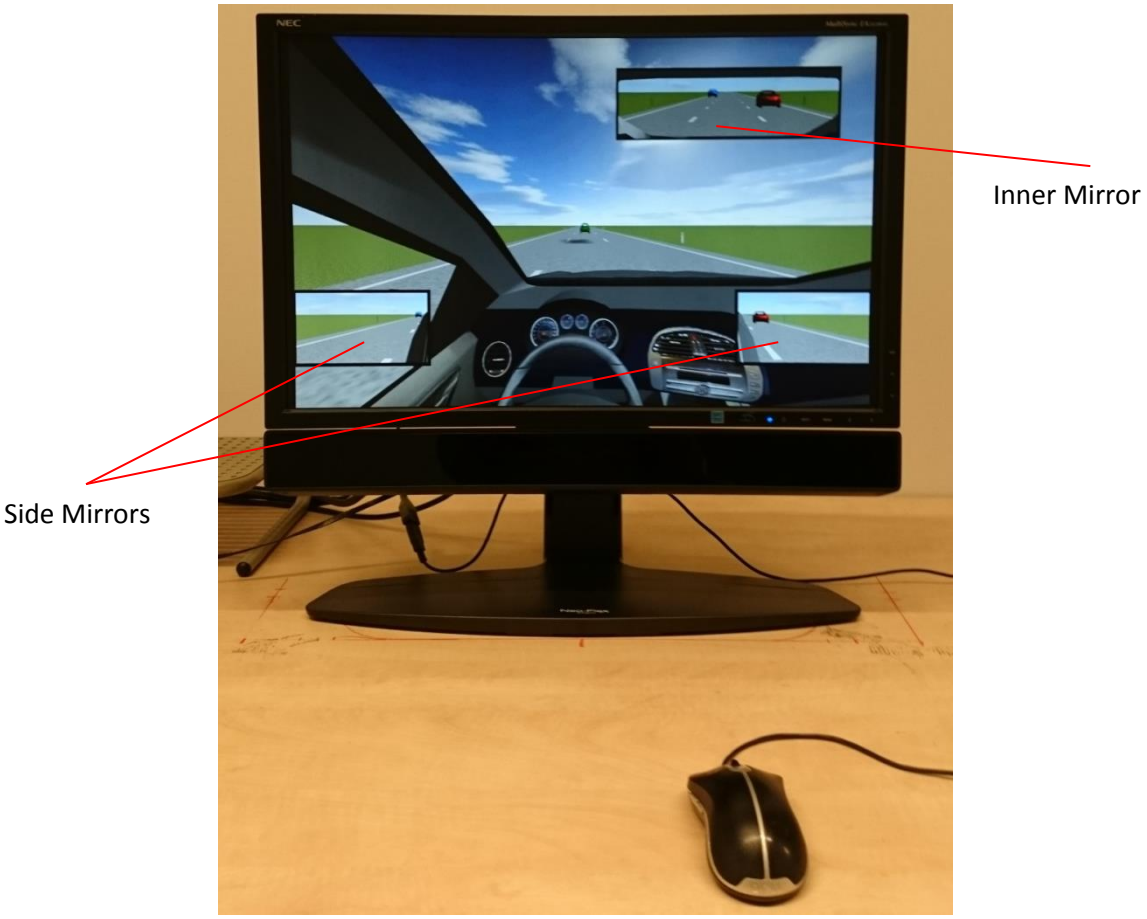
Purpose of the study: The purpose of this study is to investigate how quickly drivers can gain knowledge of a traffic situation. Furthermore, using an eye-tracker, we will measure how you move your eyes while observing a traffic situation. The results of this experiment will be published in a Master's thesis and research paper.

Duration: Your participation in this study will last approximately 60 minutes. You will be compensated for your time with €5 and have the right to withdraw from the experiment at any time, without losing the compensation.

Procedures: Before the experiment starts, you will be asked to fill out a questionnaire about your past driving experiences. Next, you will position yourself at approximately 65 cm from the monitor, after which the eye-tracker will be calibrated.

Next, you will watch 26 videos of traffic scenarios on a three-lane highway (see figure), 2 of which are used as training, followed by 24 test trials. The scenarios vary in length from 1 to 20 seconds.

After each video, you are asked to reproduce the traffic situation as accurately as possible. This means you have to position the cars and indicate their relative speeds in a top-down view. Furthermore, after each video, you are asked to fill out a questionnaire about your subjective difficulty and time pressure.



Risks and discomforts: There are no known risks for you in this study. You may feel uncomfortable focussing your eyes on a fixed point on the screen for a period of time.

Confidentiality: All data collected in this study will be kept confidential. You will not be identifiable in any documentation obtained from this study.

Right to refuse or withdraw: Your participation in this study is voluntary. You have the right to refuse or withdraw from this experiment at any time, without negative consequences.

Questions: Contact Xander Coster (contact details are included at the top) in case you have any questions or concerns about this study or your rights as a research participant.

I have read and understood the information provided above.
I give permission to process the data for the purposes described above.
I voluntarily agree to participate in this study.

Name:

.....

Signature of participant: Date:

..../ ...../ .....

Appendix A1-2: Pre-test Questionnaire

This appendix contains the pre-test questionnaire, providing information about participants' driving experience and habits.

Participant Number:	Date:	Time:

Questionnaire:

1) What is your age?

   …….

2) What is your gender?

   Male □ Female □

3) At which age did you obtain your first driver's license?

   ……

4) On average, how often did you drive a car or motorcycle during the last 12 months?

   Every day □ 4-6 days a week □ 1-3 days a week □

   Once a month □ to once a week Less than once a month □ Never □

5) About how many kilometres did you drive during the last 12 months?

   0 □ 1 – 1,000 □ 1,001 – 5,000 □

   5,001 – 10,000 □ 10,001 – 15,000 □ 15,001 – 20,000 □

   20,001 – 25,000 □ 25,001 – 35,000 □ 35,001 – 50,000 □

   50,001 – 100,000 □ More than 100,000 □

6) Have you ever heard of the Google Driverless Car?

   Yes □ No □

Appendix A1-3: Questionnaire for indicating intended matches

This questionnaire was handed to the first three participants during four random trials at the step where they received feedback. In the table on the questionnaire, participants could write numbers in the left and right columns corresponding to the numbers of the cars they had placed and the numbers of the actual cars they intended to match with, respectively. This same form was used during the pilot-study.

Participant Number:	Date:	Time:	Trial:

Please fill in the numbers of the cars you placed on the left, and the numbers of the cars from the actual situation (that you intended to indicate with the particular placed car) on the right.

Appendix A2: Information about scenarios and videos

Appendix A2-1: Complete overview of set up of scenarios

This Appendix contains all information about the setup of cars for each scenario. This entails distribution of cars over lanes (i.e., the three digits after the scenario number), type and colour of cars, (relative) end distance, start distance (both for used video lengths and for the test case of 20 seconds), and their speed.

Training1	Type	Colour	Start20	Start12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	28.00	860.00	0.00
L1	Ford Focus Wagon	Navy	340.00	564.00	28.00	900.00	40.00
M1	Fiat Bravo	White	240.00	464.00	28.00	800.00	-60.00
R1	Audi A8	Lime Green	400.00	624.00	28.00	960.00	100.00

Training2	Type	Colour	Start20	Start12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	28.00	860.00	0.00
L1	BMW X5	Dodger Blue	220.00	454.00	29.25	805.00	-55.00
M1	Toyota Yaris	Dark Green	330.00	574.00	30.50	940.00	80.00
R1	Mazda RX8	Dark Red	270.00	484.00	26.75	805.00	-55.00

6 Cars

1_312_1	Type	Colour	Start20	Start 7	Start 9	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	664.00	608.00	28.00	860.00	0.00
L1	Audi A8	Red	242.75	623.00	564.50	29.25	827.75	-32.25
L2	Toyota Yaris	Dim Gray	407.50	771.50	715.50	28.00	967.50	107.50
L3	Fiat Bravo	White Smoke	349.25	680.75	629.75	25.50	859.25	-0.75
M1	Citroen C3	Silver	263.75	611.50	558.00	26.75	798.75	-61.25
M2				0.00	0.00		0.00	-860.00
R1	Mazda RX8	Lime Green	288.50	668.75	610.25	29.25	873.50	13.50
R2	BMW Z3	Dark Green	420.00	751.50	700.50	25.50	930.00	70.00
R3								

2_213_1	Type	Colour	Start20	Start 9	Start 12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	608.00	524.00	28.00	860.00	0.00
L1	Ford Focus Wagon	Navy	257.50	593.00	501.50	30.50	867.50	7.50
L2	Toyota Yaris	Black	343.25	651.25	567.25	28.00	903.25	43.25
L3				0.00	0.00		0.00	-860.00
M1	Ford Fiesta	Dark Red	372.50	653.00	576.50	25.50	882.50	22.50
M2				0.00	0.00		0.00	-860.00
R1	BMW X5	Blue	307.75	588.25	511.75	25.50	817.75	-42.25
R2	Audi A8	White Smoke	390.50	684.75	604.50	26.75	925.50	65.50
R3	Citroen C3	Dark Green	322.75	644.50	556.75	29.25	907.75	47.75

3_123_1	Type	Colour	Start20	Start 3	Start 7	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	776.00	664.00	28.00	860.00	0.00
L1	Audi A8	Blue	246.00	722.00	610.00	28.00	806.00	-54.00
L2				0.00	0.00		0.00	-860.00
L3				0.00	0.00		0.00	-860.00
M1	Fiat Bravo	Light Grey	271.00	704.50	602.50	25.50	781.00	-79.00
M2	Toyota Previa	Black	372.00	805.50	703.50	25.50	882.00	22.00
R1	Ford Fiesta	White Smoke	222.25	698.25	586.25	28.00	782.25	-77.75
R2	Ford Focus Wagon	Navy	279.00	776.25	659.25	29.25	864.00	4.00
R3	Citroen C3	Dodger Blue	244.25	762.75	640.75	30.50	854.25	-5.75

4_321_1	Type	Colour	Start20	Start 3	Start 9	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	776.00	608.00	28.00	860.00	0.00
L1	Citroen C3	Black	356.75	875.25	692.25	30.50	966.75	106.75
L2	Toyota Previa	Dark Green	291.00	724.50	571.50	25.50	801.00	-59.00
L3	Fiat Bravo	Light Grey	297.25	752.00	591.50	26.75	832.25	-27.75
M1	BMW X5	Silver	263.25	718.00	557.50	26.75	798.25	-61.75
M2	Audi A8	Red	332.00	850.50	667.50	30.50	942.00	82.00
R1	Ford Fiesta	Lime Green	384.00	860.00	692.00	28.00	944.00	84.00
R2				0.00	0.00		0.00	-860.00
R3				0.00	0.00		0.00	-860.00

5_312_2	Type	Colour	Start20	Start 20	Start 1	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	300.00	832.00	28.00	860.00	0.00
L1	Toyota Yaris	Dodger Blue	228.50	228.50	760.50	28.00	788.50	-71.50
L2	Toyota Previa	Dark Green	293.25	293.25	777.75	25.50	803.25	-56.75
L3	Citroen C3	Dark Red	358.75	358.75	867.00	26.75	893.75	33.75
M1	Mazda RX8	Silver	373.00	373.00	881.25	26.75	908.00	48.00
M2				0.00	0.00		0.00	-860.00
R1	BMW X5	Black	337.75	337.75	917.25	30.50	947.75	87.75
R2	Fiat Bravo	Blue	317.25	317.25	801.75	25.50	827.25	-32.75
R3				0.00	0.00		0.00	-860.00

6_222_1	Type	Colour	Start20	Start 9	Start 20	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	608.00	300.00	28.00	860.00	0.00
L1	Toyota Previa	Dark Red	296.25	590.50	296.25	26.75	831.25	-28.75
L2	Fiat Bravo	Blue	330.25	652.00	330.25	29.25	915.25	55.25
L3			0.00	0.00	0.00	0.00		-860.00
M1	BMW X5	White Smoke	357.25	679.00	357.25	29.25	942.25	82.25
M2	Ford Focus Wagon	Dark Green	283.75	564.25	283.75	25.50	793.75	-66.25
R1	Audi A8	Navy	309.50	645.00	309.50	30.50	919.50	59.50
R2	BMW Z3	Silver	296.25	618.00	296.25	29.25	881.25	21.25
R3				0.00	0.00		0.00	-860.00

7_213_2	Type	Colour	Start20	Start 12	Start 20	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	300.00	28.00	860.00	0.00
L1	Audi A8	Dark Green	376.75	600.75	376.75	28.00	936.75	76.75
L2	Mazda RX8	White Smoke	340.50	544.50	340.50	25.50	850.50	-9.50
L3				0.00	0.00		0.00	-860.00
M1	BMW Z3	Red	276.00	490.00	276.00	26.75	811.00	-49.00
M2				0.00	0.00		0.00	-860.00
R1	Toyota Previa	Blue	243.50	467.50	243.50	28.00	803.50	-56.50
R2	BMW X5	Dim Gray	300.75	534.75	300.75	29.25	885.75	25.75
R3	Ford Fiesta	Dark Red	279.25	493.25	279.25	26.75	814.25	-45.75

8_321_2	Type	Colour	Start20	Start 1	Start 3	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	832.00	776.00	28.00	860.00	0.00
L1	Citroen C3	Red	234.50	814.00	753.00	30.50	844.50	-15.50
L2	BMW X5	Light Grey	324.25	880.00	821.50	29.25	909.25	49.25
L3	Fiat Bravo	Black	315.75	847.75	791.75	28.00	875.75	15.75
M1	BMW Z3	Dark Red	264.00	796.00	740.00	28.00	824.00	-36.00
M2	Ford Focus Wagon	Dim Gray	348.25	856.50	803.00	26.75	883.25	23.25
R1	Toyota Previa	Silver	344.75	876.75	820.75	28.00	904.75	44.75
R2				0.00	0.00		0.00	-860.00
R3				0.00	0.00		0.00	-860.00

MIRROR of 8

9_123_3	Type	Colour	Start20	Start 20	Start 3	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	300.00	776.00	28.00	860.00	0.00
L1	Toyota Previa	Silver	344.75	344.75	820.75	28.00	904.75	44.75
L2				0.00	0.00		0.00	-860.00
L3				0.00	0.00		0.00	-860.00
M1	BMW Z3	Dark Red	264.00	264.00	740.00	28.00	824.00	-36.00
M2	Ford Focus Wagon	Dim Gray	348.25	348.25	803.00	26.75	883.25	23.25
R1	Citroen C3	Red	234.50	234.50	753.00	30.50	844.50	-15.50
R2	BMW X5	Light Grey	324.25	324.25	821.50	29.25	909.25	49.25
R3	Fiat Bravo	Black	315.75	315.75	791.75	28.00	875.75	15.75

10_222_2	Type	Colour	Start20	Start 1	Start 7	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	832.00	664.00	28.00	860.00	0.00
L1	Mazda RX8	Wite	360.25	916.00	740.50	29.25	945.25	85.25
L2	Fiat Bravo	White Smoke	338.00	822.50	669.50	25.50	848.00	-12.00
L3				0.00	0.00		0.00	-860.00
M1	Toyota Previa	Red	315.50	871.25	695.75	29.25	900.50	40.50
M2	BMW Z3	Light Grey	266.75	798.75	630.75	28.00	826.75	-33.25
R1	Toyota Yaris	Dark Red	273.25	852.75	669.75	30.50	883.25	23.25
R2	Audi A8	Navy	232.50	788.25	612.75	29.25	817.50	-42.50
R3				0.00	0.00		0.00	-860.00

Mirror of 7

11_312_3	Type	Colour	Start20	Start 7	Start 12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	664.00	524.00	28.00	860.00	0.00
L1	Toyota Previa	Blue	243.50	607.50	467.50	28.00	803.50	-56.50
L2	BMW X5	Dim Gray	300.75	681.00	534.75	29.25	885.75	25.75
L3	Ford Fiesta	Dark Red	279.25	627.00	493.25	26.75	814.25	-45.75
M1	BMW Z3	Red	276.00	623.75	490.00	26.75	811.00	-49.00
M2				0.00	0.00		0.00	-860.00
R1	Audi A8	Dark Green	376.75	740.75	600.75	28.00	936.75	76.75
R2	Mazda RX8	White Smoke	340.50	672.00	544.50	25.50	850.50	-9.50
R3				0.00	0.00		0.00	-860.00

12_321_2	Type	Colour	Start20	Start 12	Start 1	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	832.00	28.00	860.00	0.00
L1	BMW Z3	Black	288.00	492.00	772.50	25.50	798.00	-62.00
L2	BMW X5	Lime Green	342.25	586.25	921.75	30.50	952.25	92.25
L3	Ford Focus Wagon	White	323.25	557.25	879.00	29.25	908.25	48.25
M1	Toyota Previa	Silver	276.75	490.75	785.00	26.75	811.75	-48.25
M2	Citroen C3	Navy	364.50	568.50	849.00	25.50	874.50	14.50
R1	Audi A8	Light Grey	336.50	560.50	868.50	28.00	896.50	36.50
R2				0.00	0.00		0.00	-860.00
R3				0.00	0.00		0.00	-860.00

4 Cars

13_121_1	Type	Colour	Start20	Start 9	Start 12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	608.00	524.00	28.00	860.00	0.00
L1	Ford Fiesta	White Smoke	272.50	566.75	486.50	26.75	807.50	-52.50
L2				0.00	0.00		0.00	-860.00
M1	Citroen C3	Red	274.50	582.50	498.50	28.00	834.50	-25.50
M2	Mazda RX8	White	366.25	701.75	610.25	30.50	976.25	116.25
R2	Toyota Previa	Black	292.75	628.25	536.75	30.50	902.75	42.75
R3							0.00	-860.00

14_112_1	Type	Colour	Start20	Start 12	Start 20	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	300.00	28.00	860.00	0.00
L1	Citroen C3	White Smoke	339.00	583.00	339.00	30.50	949.00	89.00
L2				0.00	0.00		0.00	-860.00
M1	Toyota Yaris	Light Grey	416.50	620.50	416.50	25.50	926.50	66.50
M2				0.00	0.00		0.00	-860.00
R2	Mazda RX8	Lime Green	351.25	575.25	351.25	28.00	911.25	51.25
R3	Ford Fiesta	Blue	225.25	469.25	225.25	30.50	835.25	-24.75

15_211_1	Type	Colour	Start20	Start 20	Start 1	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	300.00	832.00	28.00	860.00	0.00
L1	Audi A8	Navy	343.50	343.50	899.25	29.25	928.50	68.50
L2	Fiat Bravo	Light Grey	250.50	250.50	758.75	26.75	785.50	-74.50
M1	Ford Fiesta	Dark Red	322.75	322.75	878.50	29.25	907.75	47.75
M2				0.00	0.00		0.00	-860.00
R2	Mazda RX8	Black	276.75	276.75	761.25	25.50	786.75	-73.25
R3				0.00	0.00		0.00	-860.00

16_202_1	Type	Colour	Start20	Start 7	Start 9	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	664.00	608.00	28.00	860.00	0.00
L1	BMW X5	Silver	412.75	744.25	693.25	25.50	922.75	62.75
L2	BMW Z3	Blue	225.00	621.50	560.50	30.50	835.00	-25.00
M1				0.00	0.00		0.00	-860.00
M2				0.00	0.00		0.00	-860.00
R2	Audi A8	Light Grey	311.75	692.00	633.50	29.25	896.75	36.75
R3	Ford Focus Wagon	Dark Green	296.75	644.50	591.00	26.75	831.75	-28.25

17_022_1	Type	Colour	Start20	Start 12	Start 1	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	524.00	832.00	28.00	860.00	0.00
L1				0.00	0.00		0.00	-860.00
L2				0.00	0.00		0.00	-860.00
M1	BMW Z3	White	317.25	551.25	873.00	29.25	902.25	42.25
M2	Ford Focus Wagon	Dark Green	259.25	493.25	815.00	29.25	844.25	-15.75
R2	Audi A8	Black	316.50	520.50	801.00	25.50	826.50	-33.50
R3	Toyota Yaris	White Smoke	371.00	595.00	903.00	28.00	931.00	71.00

18_220_1	Type	Colour	Start20	Start 3	Start 7	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	776.00	664.00	28.00	860.00	0.00
L1	BMW Z3	Lime Green	265.75	720.50	613.50	26.75	800.75	-59.25
L2	Ford Focus Wagon	White	307.25	804.50	687.50	29.25	892.25	32.25
M1	Toyota Yaris	Silver	246.25	743.50	626.50	29.25	831.25	-28.75
M2	Audi A8	Light Grey	373.50	807.00	705.00	25.50	883.50	23.50
R2				0.00	0.00		0.00	-860.00
R3				0.00	0.00		0.00	-860.00

19_121_2	Type	Colour	Start20	Start 7	Start 12	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	664.00	524.00	28.00	860.00	0.00
L1	Fiat Bravo	Red	322.50	702.75	556.50	29.25	907.50	47.50
L2				0.00	0.00		0.00	-860.00
M1	Citroen C3	Dark Red	234.25	598.25	458.25	28.00	794.25	-65.75
M2	Toyota Yaris	Black	408.50	756.25	622.50	26.75	943.50	83.50
R2	BMW Z3	Navy	262.75	659.25	506.75	30.50	872.75	12.75
R3				0.00	0.00		0.00	-860.00

20_112_2	Type	Colour	Start20	Start 1	Start 3	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	832.00	776.00	28.00	860.00	0.00
L1	Fiat Bravo	Dim Gray	410.25	894.75	843.75	25.50	920.25	60.25
L2				0.00	0.00		0.00	-860.00
M1	Citroen C3	White	281.75	766.25	715.25	25.50	791.75	-68.25
M2				0.00	0.00		0.00	-860.00
R2	BMW X5	Blue	333.75	865.75	809.75	28.00	893.75	33.75
R3	Ford Fiesta	Dark Green	292.25	848.00	789.50	29.25	877.25	17.25

21_211_2	Type	Colour	Start20	Start 9	Start 20	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	608.00	300.00	28.00	860.00	0.00
L1	Ford Fiesta	Light Grey	291.00	585.25	291.00	26.75	826.00	-34.00
L2	Citroen C3	White Smoke	350.75	658.75	350.75	28.00	910.75	50.75
M1	Ford Focus Wagon	Dim Gray	392.00	672.50	392.00	25.50	902.00	42.00
M2				0.00	0.00		0.00	-860.00
R2	BMW X5	Red	351.00	686.50	351.00	30.50	961.00	101.00
R3				0.00	0.00		0.00	-860.00

22_202_2	Type	Colour	Start20	Start 3	Start 9	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	776.00	608.00	28.00	860.00	0.00
L1	Mazda RX8	White	250.50	726.50	558.50	28.00	810.50	-49.50
L2	Ford Fiesta	Light Grey	394.00	827.50	674.50	25.50	904.00	44.00
M1				0.00	0.00		0.00	-860.00
M2				0.00	0.00		0.00	-860.00
R2	Ford Focus Wagon	Dim Gray	339.25	794.00	633.50	26.75	874.25	14.25
R3	BMW X5	White Smoke	369.50	888.00	705.00	30.50	979.50	119.50

23_022_2	Type	Colour	Start20	Start 1	Start 7	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	832.00	664.00	28.00	860.00	0.00
L1				0.00	0.00		0.00	-860.00
L2				0.00	0.00		0.00	-860.00
M1	BMW X5	Red	240.25	772.25	604.25	28.00	800.25	-59.75
M2	Fiat Bravo	Dodger Blue	319.75	899.25	716.25	30.50	929.75	69.75
R2	Ford Fiesta	White Smoke	295.25	779.75	626.75	25.50	805.25	-54.75
R3	Mazda RX8	Lime Green	361.00	869.25	708.75	26.75	896.00	36.00

24_220_2	Type	Colour	Start20	Start 20	Start 3	Speed (m/s)	End:	Dist:
SELF	Fiat Bravo	Grey	300.00	300.00	776.00	28.00	860.00	0.00
L1	BMW Z3	Dodger Blue	290.25	290.25	766.25	28.00	850.25	-9.75
L2	Audi A8	Silver	347.00	347.00	801.75	26.75	882.00	22.00
M1	Mazda RX8	Dark Green	316.00	316.00	834.50	30.50	926.00	66.00
M2	Toyota Previa	Lime Green	230.50	230.50	727.75	29.25	815.50	-44.50
R2				0.00	0.00		0.00	-860.00
R3				0.00	0.00		0.00	-860.00

Appendix A2-2: Order of appearance of scenarios

Order of appearance of videos for the experimental trials was randomized for each participant. Sequences of videos for each participant are shown in Table A2-1, the order of appearance of the different video lengths for each participant is shown in Table A2-2.

Table A2-1: Order of appearance of videos for experimental trials for each participant, where the numbers indicate the video number. Numbers 1 and 2 were used for the two training trials/videos, which were given in the same order to each participant prior to the experimental trials.

Trial Nr. / PN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	5	4	20	11	19	21	18	10	14	25	7	8	22	26	3	12	23	17	6	16	13	24	15	9
2	4	15	11	14	22	26	13	8	5	18	24	25	17	3	10	16	6	7	19	9	12	21	23	20
3	18	19	5	3	16	23	4	9	10	20	13	26	24	15	14	22	21	6	7	8	11	12	17	25
4	24	7	4	25	5	13	9	23	3	18	19	17	20	10	16	15	8	14	21	22	26	11	6	12
5	7	13	21	4	10	26	24	8	20	9	6	25	14	17	12	22	3	16	18	19	11	5	15	23
6	8	6	21	11	19	23	10	3	24	12	5	7	13	20	16	18	17	4	26	9	22	14	15	25
7	22	26	10	11	5	7	6	23	14	19	25	3	20	18	4	12	17	16	15	13	24	9	8	21
8	9	22	3	4	20	18	14	25	7	11	5	10	13	17	19	24	8	21	16	12	23	15	26	6
9	14	21	15	23	25	24	26	19	17	13	5	3	9	11	20	6	22	8	12	7	18	10	4	16
10	12	9	15	5	17	25	19	13	6	22	20	21	10	7	14	26	24	4	16	11	23	8	18	3
11	14	5	3	7	22	10	8	17	4	26	23	13	18	20	16	21	12	24	15	19	25	9	6	11
12	16	15	12	18	25	3	17	22	14	21	11	20	4	13	19	6	8	10	23	7	24	9	26	5
13	11	9	8	7	21	22	18	3	25	5	12	6	20	14	17	10	15	26	16	4	13	19	23	24
14	14	8	13	10	24	3	21	7	6	5	18	15	12	22	25	19	23	9	26	17	16	4	20	11
15	12	25	15	7	24	16	20	5	13	19	9	22	10	21	8	26	11	14	23	3	4	18	6	17
16	5	4	20	11	19	21	18	10	14	25	7	8	22	26	3	12	23	17	6	16	13	24	15	9
17	23	24	13	14	10	25	17	11	9	15	16	21	3	26	18	22	12	8	4	7	19	6	20	5
18	16	3	10	4	13	6	20	14	11	12	22	19	17	26	7	21	5	9	15	23	24	25	8	18
19	23	7	13	21	26	19	15	4	20	5	24	12	14	9	18	6	22	11	3	8	10	17	16	25
20	26	19	12	18	9	14	15	3	21	16	25	10	17	5	6	7	13	24	8	23	20	11	22	4
21	20	16	19	12	23	14	11	17	8	10	18	4	21	5	3	13	25	7	22	24	9	15	6	26
22	8	10	6	18	16	22	13	11	7	15	25	5	19	14	20	12	3	23	24	17	26	9	4	21
23	16	21	20	25	4	12	26	10	18	22	24	8	9	23	11	17	14	5	7	13	19	6	3	15
24	6	14	17	21	15	25	19	4	10	22	7	12	20	5	18	9	3	26	11	8	23	24	16	13
25	12	13	8	25	18	22	3	10	17	24	23	7	4	21	20	15	16	9	6	19	14	11	26	5
26	12	6	16	20	22	24	9	14	21	18	15	19	25	4	10	3	17	5	23	7	8	11	13	26
27	26	13	25	8	4	12	23	19	6	9	20	16	15	5	22	21	3	10	17	7	14	24	11	18
28	11	25	24	8	7	9	20	23	6	14	10	22	26	18	13	16	15	5	19	12	21	17	3	4
29	12	10	4	5	13	14	26	25	9	16	21	20	24	6	15	11	19	8	17	3	18	22	23	7
30	9	4	18	3	13	22	16	19	7	14	17	21	20	24	25	12	8	23	11	10	5	15	6	26
31	5	21	22	19	20	6	3	11	7	4	26	15	23	24	16	14	18	12	8	25	13	17	9	10
32	3	22	26	23	17	9	19	24	20	25	15	12	8	16	21	7	5	4	6	18	14	13	10	11
33	20	24	6	15	10	17	25	18	8	21	19	22	12	14	13	5	16	9	3	26	23	11	4	7
34	13	25	3	10	8	9	19	26	24	16	6	4	15	21	7	17	20	5	14	18	23	11	22	12

Table A2-2: Order of appearance of video lengths for experimental trials for each participant, where the numbers indicate the video lengths. The table does not contain video lengths for the two training trials/videos, which were given to each participant prior to the experimental trials. Both training trials lasted twelve seconds.

Trial Nr. / PN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	3	9	3	20	12	7	7	1	12	1	20	9	1	20	7	1	9	20	3	12	7	3	9	12
2	12	12	3	1	3	3	12	20	7	9	9	7	1	9	3	20	9	1	1	20	7	12	20	7
3	7	12	3	7	12	9	9	12	1	3	7	20	3	9	12	1	7	3	20	9	20	1	20	1
4	9	1	12	7	7	12	20	20	9	9	1	1	7	3	20	12	20	1	12	3	3	3	9	7
5	20	7	7	9	1	20	3	9	3	12	3	1	12	20	1	1	7	12	7	12	20	3	9	9
6	20	9	12	3	1	20	3	9	9	7	7	1	12	7	20	9	1	12	3	20	3	1	12	7
7	1	20	1	20	3	20	3	9	12	12	1	7	3	7	9	1	20	12	9	7	3	12	9	7
8	20	3	9	12	7	9	1	7	1	3	7	3	12	1	1	9	20	12	20	7	20	12	3	9
9	12	7	9	9	1	3	20	12	20	7	3	7	12	20	3	3	1	9	1	20	7	1	9	12
10	7	20	12	7	1	7	1	12	9	3	7	12	3	1	1	3	9	12	20	3	20	20	9	9
11	12	3	7	20	1	1	9	20	9	20	9	7	7	3	12	7	1	3	9	12	1	12	3	20
12	20	12	7	9	7	9	1	3	1	12	3	7	12	12	1	9	20	3	20	1	9	20	3	7
13	20	12	9	20	7	1	7	7	1	3	1	3	3	12	20	1	9	20	12	9	7	12	9	3
14	1	20	12	3	9	9	12	1	9	7	9	12	7	3	7	1	20	20	3	1	20	12	7	3
15	1	1	9	20	3	12	3	3	7	12	12	1	1	7	9	20	20	12	9	7	9	7	3	20
16	7	12	7	3	1	12	9	3	1	7	1	20	3	3	9	7	20	1	9	20	12	9	12	20
17	9	3	7	12	1	1	20	20	12	9	12	7	7	20	7	1	1	9	9	20	12	3	3	3
18	20	9	3	12	12	9	7	1	3	7	3	1	1	3	1	12	7	20	12	20	9	7	20	9
19	9	20	7	7	20	12	9	9	3	3	3	1	12	12	7	3	1	20	7	9	1	20	12	1
20	3	1	7	9	20	1	12	9	12	20	7	3	1	7	9	1	12	9	20	20	7	3	3	12
21	3	12	12	1	9	12	20	20	9	1	7	9	7	3	7	7	1	20	1	3	12	9	3	20
22	20	3	9	9	20	3	12	3	1	12	7	7	1	1	7	7	9	20	9	1	3	20	12	12
23	12	7	3	1	9	1	20	1	7	1	3	9	12	9	20	20	12	3	20	7	12	3	7	9
24	9	1	1	12	12	7	1	12	3	3	1	7	7	7	9	20	9	3	3	20	20	9	20	12
25	1	7	9	1	7	1	7	1	20	3	9	20	9	7	3	9	12	12	3	12	12	20	20	3
26	7	9	20	7	3	9	20	1	12	9	12	1	7	12	3	9	1	7	20	1	20	3	12	3
27	20	7	1	9	9	1	9	12	3	12	3	12	9	3	1	7	7	1	20	20	12	3	20	7
28	3	7	9	20	1	20	7	20	9	1	3	3	3	9	12	20	12	7	1	7	12	1	9	12
29	1	1	9	3	7	12	20	1	12	12	7	3	3	3	9	20	12	9	20	7	7	1	9	20
30	20	12	9	9	12	3	20	1	1	1	1	12	7	9	7	7	20	20	3	3	7	12	9	3
31	3	7	1	12	3	3	7	20	20	9	20	9	9	3	12	12	7	1	9	1	7	20	12	1
32	9	3	3	20	1	20	1	9	7	7	12	7	20	20	12	1	7	12	9	9	1	12	3	3
33	3	3	3	9	1	20	1	7	9	7	12	1	1	12	7	3	12	12	7	20	9	20	9	20
34	12	7	9	3	20	20	1	3	9	20	9	12	12	12	1	1	7	7	1	9	20	3	3	7

Appendix A2-3: Placement of four cameras

In PreScan [21], four cameras were placed on the ego-vehicle in order to get a first-person view with mirrors for each scenario. The settings of these four cameras are presented in Figures A2-1 to A2-4

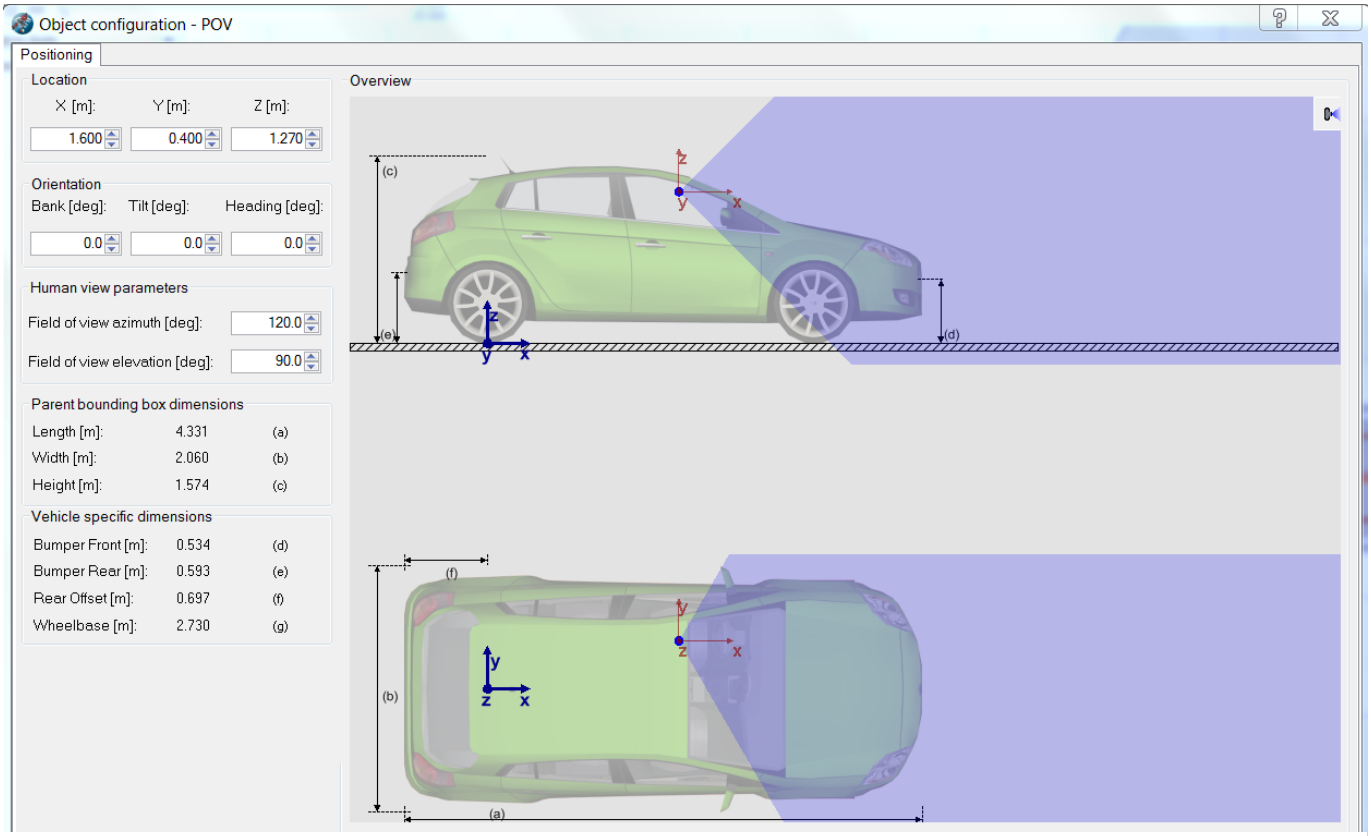


Figure A2-1: First-person view camera. Placed to resemble the position of the driver's eyes. The location was altered from PreScan's standard POV in order to have the left wing mirror completely in view.

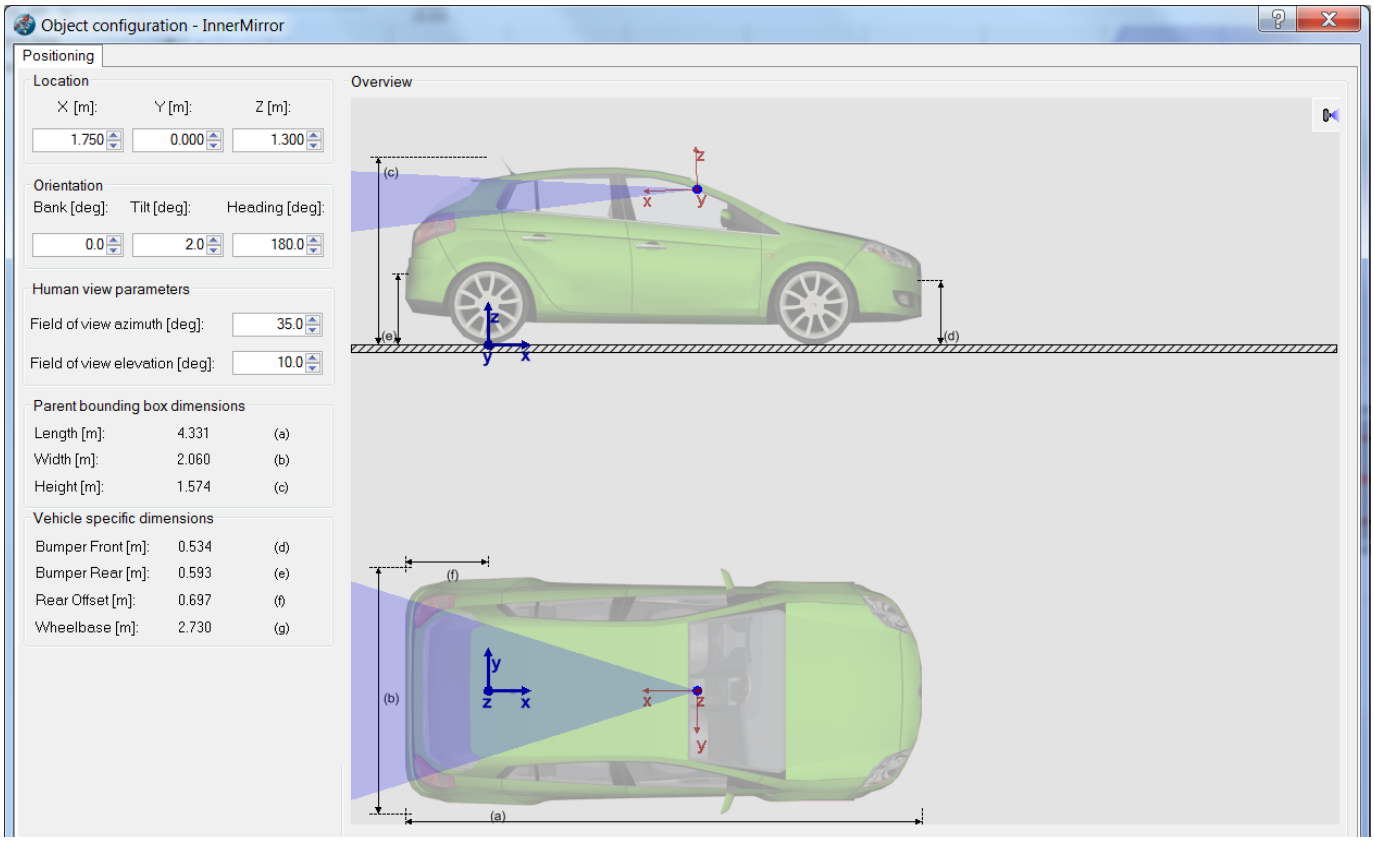


Figure A2-2: Inner mirror camera. Placed at the approximate position of the inner mirror, angled in such a way that the view resembles that of a real inner mirror. The camera's feed was recorded with a mirror effect, making it resemble the view of an actual mirror.

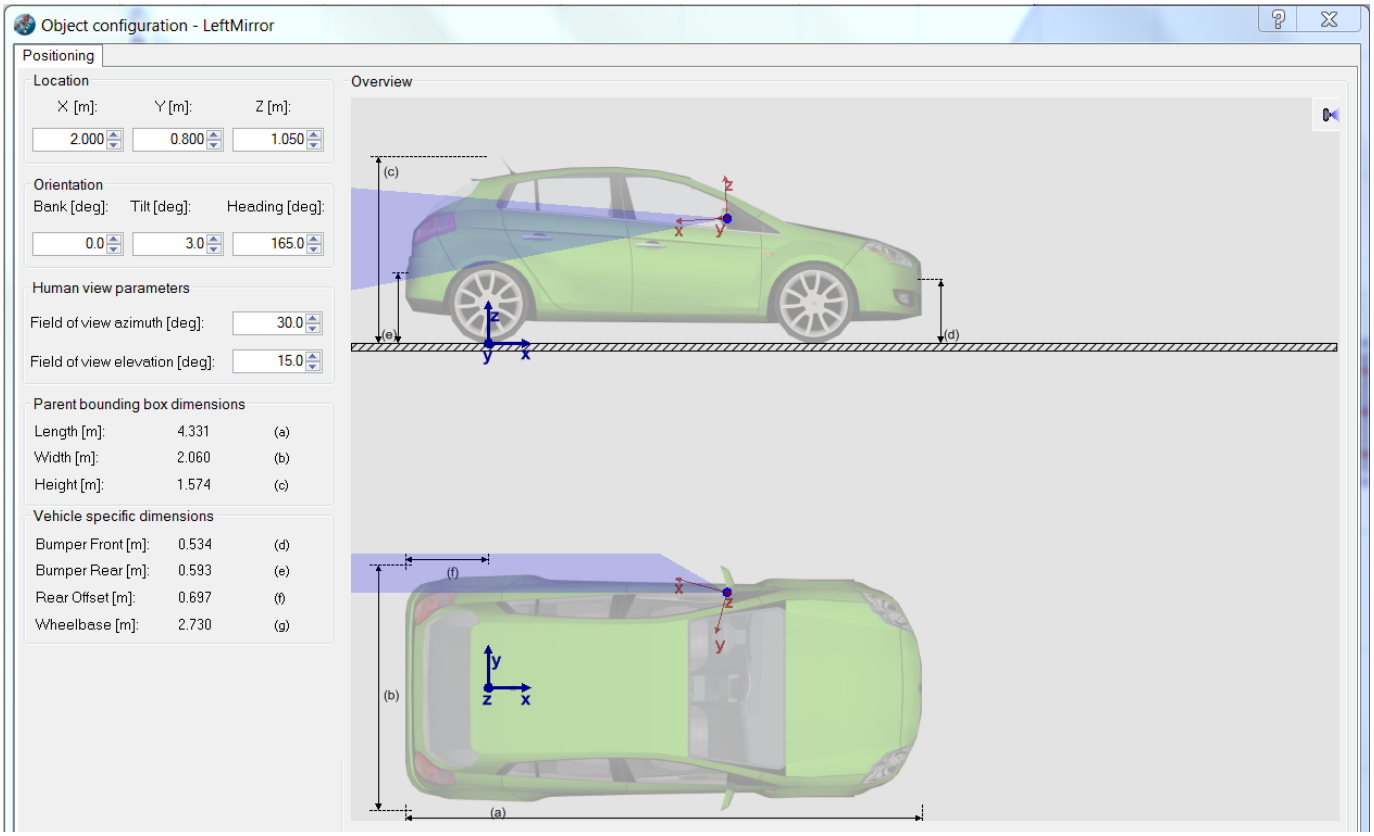


Figure A2-3: Left wing-mirror camera. Placed at the most inner point of the actual left wing-mirror, in order to resemble placement of a real left wing-mirror, but also in such a way as to force participants to use all mirrors (i.e., cars directly behind the ego-vehicle could not be seen in the wing-mirrors).

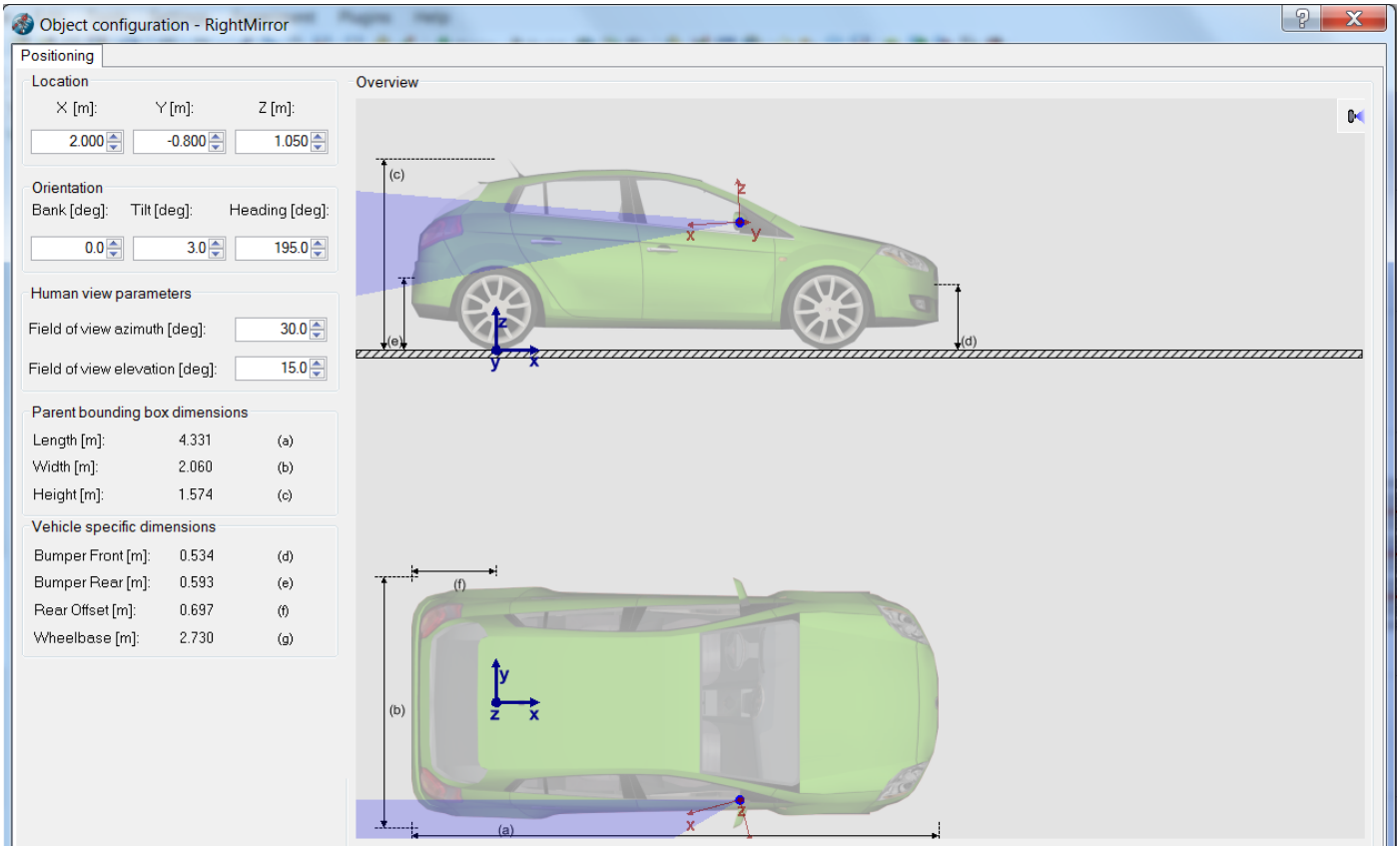


Figure A2-4: Right wing-mirror camera. Placed at the most inner point of the actual left wing-mirror, in order to resemble placement of a real left wing-mirror, but also in such a way as to force participants to use all mirrors (i.e., cars directly behind the ego-vehicle could not be seen in the wing-mirrors).

Appendix A2-4: Editing four videos into one using VSDC Free Video Editor

The four videos, recorded using PreScan, were edited into one using VSDC Free Video Editor. The different steps and layers involved in this process are explained here.

Each final video had a resolution of 1920x1080 pixels, which meant that the four separate videos needed to be fitted to that frame. During fitting, the aspect ratio of the separate videos was kept the same in order to prevent stretched images. In doing so, VSDC automatically centres video feed, and 'fills' remaining space with transparent filler (i.e., any layer underneath shows through the filler).

Each video started with a target being displayed for five seconds. The target was a 1920x1080 PNG image, see FIGURE. During target display, a beep was played, which eventually turned out to be unnecessary for the study. After the target, the traffic scenario was shown, which was built up out of the four separate videos.

The first person view (POV) video was fitted to the 1920x1080 pixels frame, which caused a small black band (i.e., the standard black background seen through the filler) on both sides of the scene. Behind each video of a mirror, a black rectangle was placed to increase visibility of the mirrors. The black rectangles were placed on top of the POV video, and were shown for the same amount of time as the POV and mirror videos. The three mirrors (i.e., two wing-mirrors and the inner mirror), were placed on top of the black rectangles. After the intended duration of the video, a black end screen was shown for 0.5 seconds, covering the complete 1920x1080 pixel frame. During display of the target and the traffic scenario, sound of a car driving on a highway was played. This sound stopped at the same time the traffic scene stopped.

For all elements of the final videos the resolutions, positions, and duration can be seen in Table A2-3.

Table A2-3: Information on all elements in a final video, containing resolutions (original and final), placement, and start and end times.

Name	Original Resolution (pixels)	Final Resolution (pixels)	Placement in frame ([left, top], pixels)	Start Time (s)	End Time (s)
Target	1920x1080	1920x1080	[0, 0]	0	5
POV View	2080x1200	1920x1080	[0, 0]	5	5+Length
Inner Mirror	2160x600	705x225	[1076.25, 76.875]	5	5+Length
Left Wing Mirror	121x600	431.25x208.125	[22.5, 740.625]	5	5+Length
Right Wing Mirror	121x600	431.25x208.125	[1466.25, 740.625]	5	5+Length
'Inner' Rectangle	dna	720x206.25	[1068.75, 84.375]	5	5+Length
'Left' Rectangle	dna	436.875x226.875	[22.5, 731.625]	5	5+Length
'Right' Rectangle	dna	436.875x226.875	[1460.625, 731.625]	5	5+Length
End Screen	dna	1920x1080	[0, 0]	5+Length	5+Length+0.5
Interior sound	dna	dna	dna	0	5+Length
Beep	dna	dna	dna	2	3.52

Appendix A3: Matching algorithm examples

Two examples are worked out for explaining the matching algorithm. In the first example, the workings of the algorithm are explained by working out one scenario where the matching algorithm made the matches as the participant intended (Participant 1, trial 16 (video 12)), but also missed one match. Example two shows in what way the matching algorithm can make mismatches (Participant 1, trial 12 (video 8)), and the consequences of them. Placed cars will be indicated with a P and actual cars with an A (e.g., P2 and A4 for placed car number two and actual car number for, respectively)

Appendix A3-1: Example 1

In Figure A3-1 boundary boxes indicate the distance within which placed cars were marked as a 'possible match' for each actual car. Table A3-1 and

Table A3-2 show all possible matches for each actual car, together with their longitudinal distances and absolute error distances, respectively. The absolute error distance is the distance calculated with Eq. (2).

Table A3-1: Possible matches for each actual car, and their (absolute) longitudinal distances, for example 1.

Actual Car:	**Possible Matches Longitudinal distances (m)**					
A1	X > 38	X > 38	X > 38	X > 38	X > 38	X X
A2	P1 3.78	P2 18.51	X > 38	P4 30.54	X > 38	X X
A3	X > 38	X > 38	P3 10.26	P4 21.96	P5 13.45	X X
A4	P1 17.47	P2 22.26	X > 38	X > 38	X > 38	X X
A5	X > 38	P2 34.24	P3 6.99	P4 4.71	P5 3.80	X X
A6	P1 26.72	P2 31.51	X > 38	X > 38	X > 38	X X

Table A3-2: Possible matches for each actual car, and their error distances, for example 1. Error distances are the absolute longitudinal distance, plus five times the lane distance between A and P cars. Absolute error distances that differ from the longitudinal distance (i.e., contain a lane error) are indicated in boldface.

Actual Car:	**Possible Matches Absolute error distances (m)**					
A1	X 38	X 38	X 38	X 38	X 38	X X
A2	P1 3.78	P2 18.51	X 38	P4 **48.04**	X 38	X X
A3	X 38	X 38	P3 **27.76**	P4 21.96	P5 **30.95**	X X
A4	P1 **34.97**	P2 22.26	X 38	X 38	X 38	X X
A5	X 38	P2 **51.74**	P3 **41.99**	P4 **22.21**	P5 3.80	X X
A6	P1 **61.72**	P2 **49.01**	X 38	X 38	X 38	X X

When all combinations of matches between actual cars and possible matches are made, 227 unique combinations of matches are found. The combination of matches used for defining further scores is the one with the lowest total absolute error distance. The total absolute error distance is found by adding absolute error distances for each match within the unique combinations. In this example, the best combination of matches (see Figure A3-1) had a total absolute error distance of 197.80 m.

The match made by the algorithm agrees with the intended match of the participant, with the exception that there is no match between P3 and A1. That P3 made no match with A1 was caused by it have been being placed too far away. This slightly influences the 'absolute error distance of hits and misses' by P3 getting an error distance of 38 instead of the actual error distance when it would have been matched with A1

(approximately 57 meters). It also counts as one less ‘hit’, which influences other scores. In this example, however, the error of P3 is so big, that counting it as a ‘miss’ seems more logical and fair than giving it the score of the actual absolute error distance.

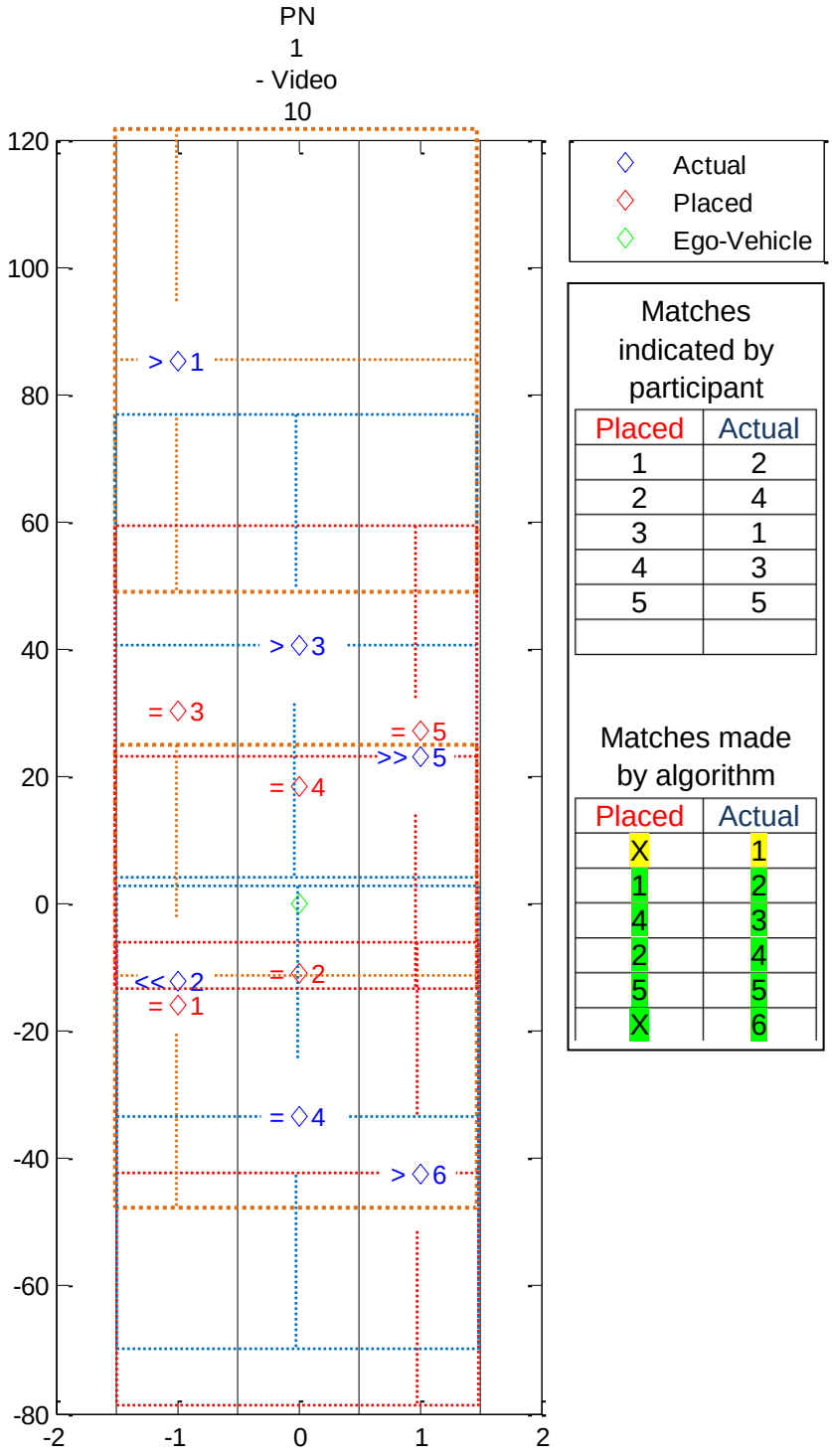


Figure A3-1: Example 1. Boundary boxes indicate the distance around actual cars within which placed cars are counted as a possible match. On the right, tables indicate the participant’s intended matches and the matches made by the matching algorithm. Green highlighting indicates the match, made by the algorithm, agrees with the one indicated by the participant. Yellow highlighting indicates a miss that was not indicated by the participant.

Appendix A3-2: Example 2

In this example, the effects of a 'mismatch' are explained.

As can be seen in Figure A3-2, no car was placed close enough to A3 that a (possible) match could be made. The intended matching of P3 to A3 is therefore justly not made. Not matching P3 to A3, however, caused the mismatch of P3 to A5. This mismatch has no effect on hit percentage, and only a small effect on average absolute error distance of hits and misses for this particular match (longitudinal + lateral distance = 26.60 m + 3.5 m = 30.10 m compared to 38 meters if it was counted as a miss). It does, however, add one to the number of wrongly indicated lanes, and the size of the error of indicated speeds is increased by one. This type of error can thus have some significant influence on those scores.

Table A3-3: Possible matches for each actual car, and their (absolute) longitudinal distances, for example 2.

Actual Car:	**Possible Matches Longitudinal distances (m)**					
A1	P1 14.03	X > 38	X > 38	P4 23.34	X X	X X
A2	X > 38	P2 33.52	P3 22.35	X > 38	X X	X X
A3	X > 38	X > 38	X > 38	X > 38	X X	X X
A4	X > 38	X > 38	X > 38	X > 38	X X	X X
A5	X > 38	P2 37.77	P3 26.60	X > 38	X X	X X
A6	P1 35.97	P2 0.48	P3 11.65	P4 26.66	X X	X X

Table A3-4: Possible matches for each actual car, and their error distances, for example 2. Error distances are the absolute longitudinal distance, plus five times the lane distance between A and P cars. Absolute error distances that differ from the longitudinal distance (i.e., contain a lane error) are indicated in boldface.

Actual Car:	**Possible Matches Absolute error distances (m)**					
A1	P1 14.03	X 38	X 38	P4 **58.34**	X X	X X
A2	X 38	P2 33.52	P3 **39.85**	X 38	X X	X X
A3	X 38	X 38	X 38	X 38	X X	X X
A4	X 38	X 38	X 38	X 38	X X	X X
A5	X 38	P2 **72.77**	P3 **44.10**	X 38	X X	X X
A6	P1 **70.97**	P2 **35.48**	P3 **29.15**	P4 26.66	X X	X X

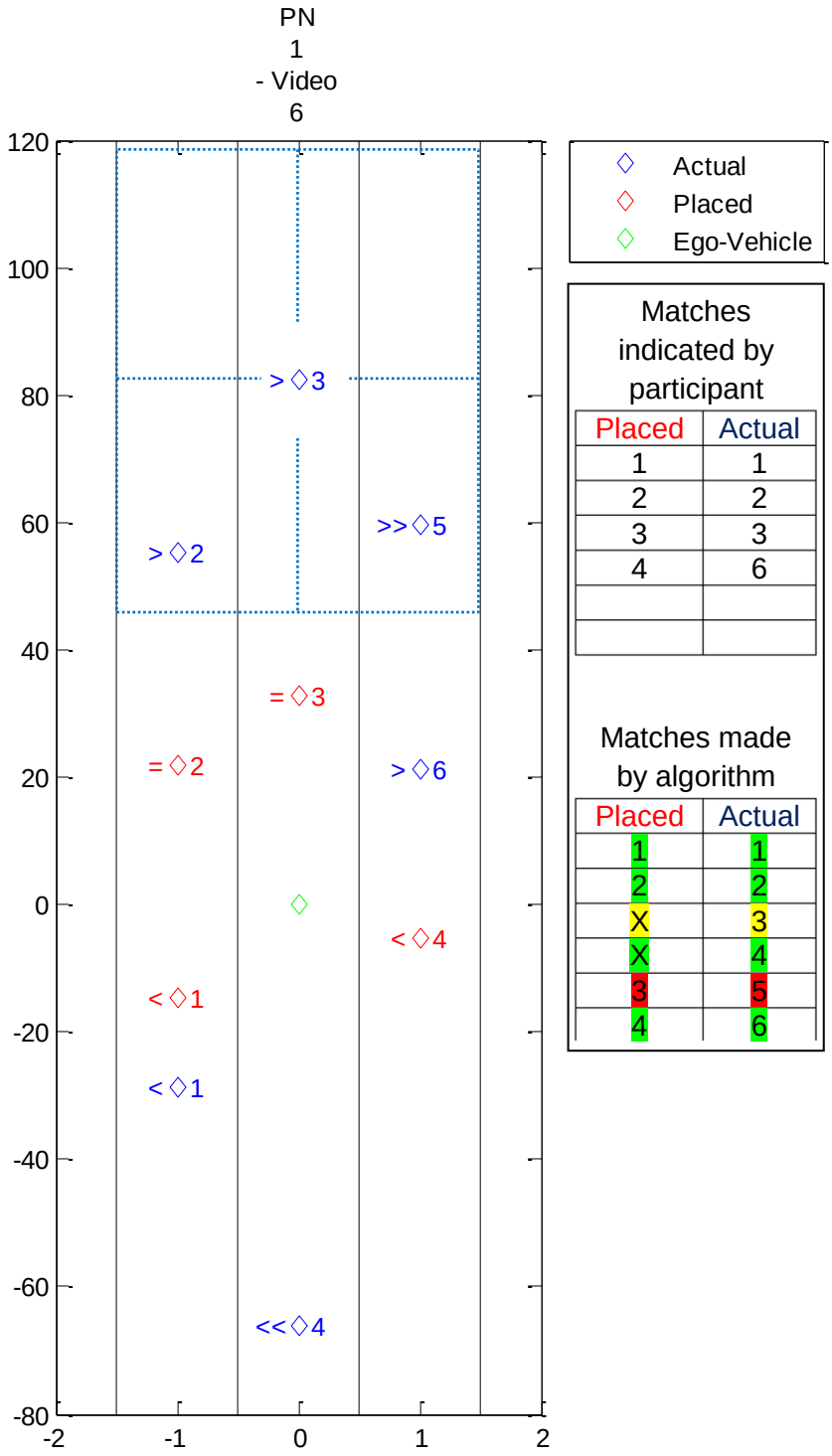


Figure A3-2: Example 2. Boundary boxes indicate the distance around actual cars within which placed cars are counted as a possible match. On the right, tables indicate the participant's intended matches and the matches made by the matching algorithm. Green highlighting indicates the match, made by the algorithm, agrees with the one indicated by the participant. Yellow highlighting indicates a miss that was not indicated by the participant. Red highlighting indicates a mismatch.

Appendix A4: Extra Statistical results

Statistical results regarding effects of video length within scenarios are presented here.

Appendix A4-1 Difficulty and Time Rating

Table A4-1: Results from paired samples t test for differences in self-reported difficulty rating due to differences in video length, where difficulty ratings between video lengths were compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars. Significant results ($p < 0.01$) are indicated in boldface.

Scenario number	Video Lengths (s)	*p*	*t*	*df*
1	7 – 9	0.199	1.343	15
2	9 – 12	0.698	-0.396	15
3	3 – 7	0.190	1.373	15
4	**3 – 9**	**< 0.001**	**4.816**	**15**
5	20 – 1	0.491	-0.706	15
6	9 – 20	0.298	1.082	14
7	12 – 20	0.270	1.145	15
8	**1 – 3**	**0.007**	**3.124**	**15**
9	20 – 3	0.091	-1.806	15
10	**1 – 7**	**0.004**	**3.450**	**15**
11	7 – 12	0.193	1.365	15
12	12 – 1	0.066	-1.981	15
13	9 – 12	0.170	1.443	15
14	12 – 20	0.324	1.020	15
15	20 – 1	0.044	-2.204	15
16	7 – 9	0.016	2.727	15
17	12 – 1	0.025	-2.482	15
18	3 – 7	0.059	2.045	15
19	7 – 12	0.201	1.339	15
20	1 – 3	0.015	2.745	15
21	9 – 20	0.205	1.329	14
22	3 – 9	0.379	0.907	15
23	1 – 7	0.014	2.773	15
24	**20 – 3**	**0.007**	**-3.140**	**15**

Table A4-2: Results from paired samples *t* test for differences in self-reported time rating due to differences in video length, where time ratings between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars. Significant results ($p < 0.01$) are indicated in boldface.

Scenario number	Video Lengths (s)	*p*	*t*	*df*
1	7 – 9	0.094	-1.800	14
2	9 – 12	0.034	-2.357	14
3	**3 – 7**	**0.001**	**-4.011**	**14**
4	**3 – 9**	**0.005**	**-3.292**	**14**
5	**20 – 1**	**< 0.001**	**9.539**	**14**
6	9 – 20	0.014	-2.852	13
7	12 – 20	0.012	-2.900	14
8	**1 – 3**	**0.001**	**-3.986**	**14**
9	**20 – 3**	**0.002**	**3.7367**	**14**
10	**1 – 7**	**< 0.001**	**-8.053**	**14**
11	7 – 12	0.032	-2.389	14
12	**12 – 1**	**< 0.001**	**6.688**	**14**
13	9 – 12	0.114	-1.384	14
14	12 – 20	0.275	-1.136	14
15	**20 – 1**	**< 0.001**	**8.149**	**14**
16	7 – 9	0.042	-2.241	14
17	**12 – 1**	**< 0.001**	**7.348**	**14**
18	**3 – 7**	**0.002**	**-3.755**	**14**
19	**7 – 12**	**0.007**	**-3.128**	**14**
20	**1 – 3**	**< 0.001**	**-5.030**	**14**
21	9 – 20	0.020	-2.658	13
22	3 – 9	0.011	-2.940	14
23	**1 – 7**	**< 0.001**	**-8.886**	**14**
24	**20 – 3**	**< 0.001**	**5.305**	**14**

Appendix A4-2 Basic Measure

Table A4-3: Results from Wilcoxon's signed rank test for differences in error percentage in total number of cars due to differences in video length, where error percentages between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Significant differences ($p < 0.01$) are indicated in boldface. Missing z values are caused by sample sizes being too small.

Scenario number	Video Lengths (s)	*p*	*z*	*T*
1	7 – 9	1	-	9
2	9 – 12	0.016	-	28
3	3 – 7	0.203	-	56.5
4	3 – 9	0.219	-	17.5
5	**20 – 1**	**< 0.001**	**-**	**0**
6	9 – 20	0.398	-	31.5
7	12 – 20	0.398	-	31.5
8	1 – 3	0.0361	-	57
9	20 – 3	0.484	-	50
10	1 – 7	0.090	-	37
11	7 – 12	0.625	-	7.5
12	12 – 1	0.434	-	19
13	9 – 12	1	-	1
14	12 – 20	1	-	1.5
15	20 – 1	0.016	-	0
16	7 – 9	1	-	1
17	12 – 1	0.406	-	5.5
18	3 – 7	0.500	-	3
19	7 – 12	0.063	-	15
20	1 – 3	0.022	-	49.5
21	9 – 20	1	-	0
22	3 – 9	0.375	-	12
23	**1 – 7**	**0.008**	**-**	**36**
24	20 – 3	1	-	1.5

Appendix A4-3 Advanced Measures

Table A4-4: Results from Wilcoxon's signed rank test for differences in hit percentage due to video length, where hit percentages between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Statistically significant differences ($p < 0.01$) are indicated in boldface. Missing z values are caused by sample sizes being too small.

Scenario number	Video Lengths (s)	*p*	*z*	*T*
1	7 – 9	1	-	12
2	9 – 12	0.063	-	0
3	3 – 7	0.086	-	7
4	3 – 9	0.129	-	12
5	**20 – 1**	**0.008**	**-**	**93**
6	9 – 20	0.563	-	12
7	12 – 20	0.313	-	2.5
8	1 – 3	0.137	-	12
9	20 – 3	0.497	-	24.5
10	1 – 7	0.017	-	11
11	7 – 12	0.766	-	10.5
12	12 – 1	0.303	-	52.5
13	9 – 12	1	-	18
14	12 – 20	1	-	4
15	20 – 1	0.813	-	12
16	7 – 9	0.500	-	0
17	12 – 1	0.109	-	24.5
18	3 – 7	1	-	5
19	7 – 12	0.063	-	0
20	1 – 3	0.045	-	10
21	9 – 20	1	-	10.5
22	3 – 9	0.453	-	8
23	1 – 7	0.018	-	4.5
24	20 – 3	1	-	4

Table A4-5: Results from paired samples *t* test for differences in average absolute error distance of hits due to differences in video length, where error distances between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Statistically significant differences ($p < 0.01$) are indicated in boldface.

Scenario number	Video Lengths (s)	***p***	***t***	***df***
1	7 – 9	0.438	-0.801	13
2	9 – 12	0.745	-0.332	13
3	3 – 7	0.635	-0.486	14
4	3 – 9	0.280	1.123	14
5	20 – 1	0.839	0.207	14
6	9 – 20	0.799	-0.261	13
7	12 – 20	0.018	2.705	13
8	1 – 3	0.172	1.439	14
9	20 – 3	0.643	-0.473	14
10	1 – 7	0.192	1.371	14
11	7 – 12	0.762	0.309	14
12	12 – 1	0.449	0.781	13
13	9 – 12	0.860	0.180	14
14	12 – 20	0.673	-0.431	14
15	20 – 1	0.545	-0.620	14
16	7 – 9	0.174	1.432	14
17	**12 – 1**	**0.005**	**-3.318**	**14**
18	3 – 7	0.809	0.246	14
19	7 – 12	0.608	-0.525	14
20	1 – 3	0.917	0.106	13
21	9 – 20	0.184	1.403	13
22	3 – 9	0.492	0.706	14
23	1 – 7	0.566	-0.588	14
24	20 – 3	0.315	-1.042	14

Table A4-6: Results from paired samples *t*-test for differences in average absolute error distance of hits and misses due to video length, where error distances between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Significant differences ($p < 0.01$) are indicated in boldface.

Scenario number	Video Lengths (s)	***p***	***t***	***df***
1	7 – 9	0.670	-0.436	13
2	9 – 12	0.421	0.832	13
3	3 – 7	0.053	2.113	14
4	3 – 9	0.058	2.069	14
5	20 – 1	0.059	-2.057	14
6	9 – 20	0.112	0.296	13
7	**12 – 20**	**0.002**	**3.828**	**13**
8	1 – 3	0.017	2.711	14
9	20 – 3	0.961	0.049	14
10	1 – 7	0.023	2.554	14
11	7 – 12	0.559	0.599	14
12	12 – 1	0.535	-0.638	13
13	9 – 12	0.849	0.194	14
14	12 – 20	0.595	-0.544	14
15	20 – 1	0.708	-0.382	14
16	7 – 9	0.074	1.932	14
17	**12 – 1**	**0.004**	**-3.440**	**14**
18	3 – 7	0.849	0.195	14
19	7 – 12	0.389	0.889	14
20	1 – 3	0.110	1.718	13
21	9 – 20	0.292	1.099	13
22	3 – 9	0.153	1.510	14
23	1 – 7	0.143	1.553	14
24	20 – 3	0.175	-1.430	14

Table A4-7: Results from Wilcoxon's signed rank test for differences in error percentage of indicated lanes due to video length, where error percentages between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Statistically significant differences ($p < 0.01$) are indicated in boldface. Missing z values are caused by sample sizes being too small.

Scenario number	Video Lengths (s)	*p*	*z*	*T*
1	7 – 9	0.111	-	59.5
2	9 – 12	0.299	-	38.5
3	3 – 7	0.102	-	6
4	3 – 9	0.222	-	72.5
5	20 – 1	0.247	-	33.5
6	9 – 20	0.712	-	37.5
7	12 – 20	0.922	-	21
8	1 – 3	0.842	-	56
9	**20 – 3**	**0.002**	**-**	**0**
10	1 – 7	0.625	-	5
11	7 – 12	0.156	-	7.5
12	12 – 1	0.117	-	6
13	9 – 12	0.516	-	9.5
14	12 – 20	0.250	-	0
15	20 – 1	0.563	-	10.5
16	7 – 9	0.438	-	4
17	12 – 1	0.281	-	6.5
18	3 – 7	0.813	-	9
19	7 – 12	0.688	-	9.5
20	1 – 3	0.295	-	45
21	9 – 20	1	-	6
22	3 – 9	0.349	-	44.5
23	1 – 7	0.125	-	10
24	20 – 3	0.563	-	10

Table A4-8: Results from Wilcoxon's signed rank test for differences in error percentage of indicated speeds due to video length, where error percentages between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Statistically significant differences ($p < 0.01$) are indicated in boldface. Missing z values are caused by sample sizes being too small.

Scenario number	Video Lengths (s)	*p*	*z*	*T*
1	7 – 9	0.796	-	57
2	9 – 12	0.145	-	58
3	3 – 7	0.060	-	72.5
4	**3 – 9**	**0.008**	**-**	**71.5**
5	**20 – 1**	**0.002**	**-**	**6**
6	9 – 20	0.085	-	70.5
7	12 – 20	0.140	-	50
8	1 – 3	0.628	-	51
9	20 – 3	0.016	-	15
10	1 – 7	0.069	-	62.5
11	7 – 12	0.935	-	47
12	12 – 1	0.112	-	27
13	9 – 12	0.018	-	89.5
14	12 – 20	0.031	-	48
15	20 – 1	0.028	-	9
16	7 – 9	0.248	-	39
17	**12 – 1**	**< 0.001**	**-**	**1**
18	**3 – 7**	**0.007**	**-**	**62**
19	7 – 12	0.484	-	23.5
20	1 – 3	0.636	-	45.5
21	9 – 20	0.139	-	50
22	3 – 9	0.051	-	46.5
23	1 – 7	0.067	-	71.5
24	**20 – 3**	**< 0.001**	**-**	**0**

Table A4-9: Results from Wilcoxon's signed rank test for differences in average size of errors in indicates speeds per hit in due to differences in video length, where average error sizes between video lengths are compared within each scenario. Scenarios 1–12 have six, and scenarios 13–24 have four cars of surrounding traffic. Scenario 7 and 11 are mirrored with each other, as are scenario 8 and 9. Significant differences ($p < 0.01$) are indicated in boldface. Missing *z* values are caused by sample sizes being too small.

Scenario number	Video Lengths (s)	***p***	***z***	***T***
1	7 – 9	0.556	-	47
2	9 – 12	0.086	-	61
3	3 – 7	0.054	-	73
4	**3 – 9**	**0.005**	**-**	**83.5**
5	**20 – 1**	**< 0.001**	**-**	**6**
6	9 – 20	0.026	-	87.5
7	12 – 20	0.174	-	57
8	1 – 3	0.571	-	49.5
9	20 – 3	0.034	-	19
10	**1 – 7**	**0.001**	**-**	**88**
11	7 – 12	0.503	-	55.5
12	12 – 1	0.267	-	34
13	9 – 12	0.041	-	95.5
14	12 – 20	0.038	-	56
15	20 – 1	0.020	-	10
16	7 – 9	0.759	-	37
17	**12 – 1**	**< 0.001**	**-**	**1**
18	**3 – 7**	**0.010**	**-**	**70.5**
19	7 – 12	0.678	-	32
20	1 – 3	1	-	39
21	9 – 20	0.057	-	63.5
22	3 – 9	0.158	-	49.5
23	1 – 7	0.105	-	78.5
24	**20 – 3**	**< 0.001**	**-**	**1**

Appendix A5: Regression analysis

A linear regression analysis was performed on the error distances of placed cars with respect to the distance of the actual cars they were matched with, for it is easier to estimate distances of closer cars than those of faraway cars. It was found that distance of actual cars was a highly significant predictor for error distance (Front: F = 779, p = 5.12e-144, Rear: F = 1.47e+03, p = 5.21e-220), where participants tended to increasingly underestimate distance of cars that were farther away (i.e., placing cars too close to the ego-vehicle), see Figures A5-1 and A5-2. Distance of actual cars to the ego-vehicle accounted for 29 and 51% of variance for the front and the rear, respectively. Error distances of hits were therefore corrected for distance of actual cars using the found fit lines as expected error distance. This was done separately for actual cars in front of and behind the ego vehicle, by finding the absolute difference between the expected error distance and the measured error distance. The expected error distances for the front and rear can be calculated using Eq. (5 and (6, respectively.

$$\text{Expected Error Distance Front} = -0.3439 \cdot d_L Cars_{Actual} + 17.94 \quad (5)$$

$$\text{Expected Error Distance Rear} = 0.5229 \cdot d_L Cars_{Actual} + 9.478 \quad (6)$$

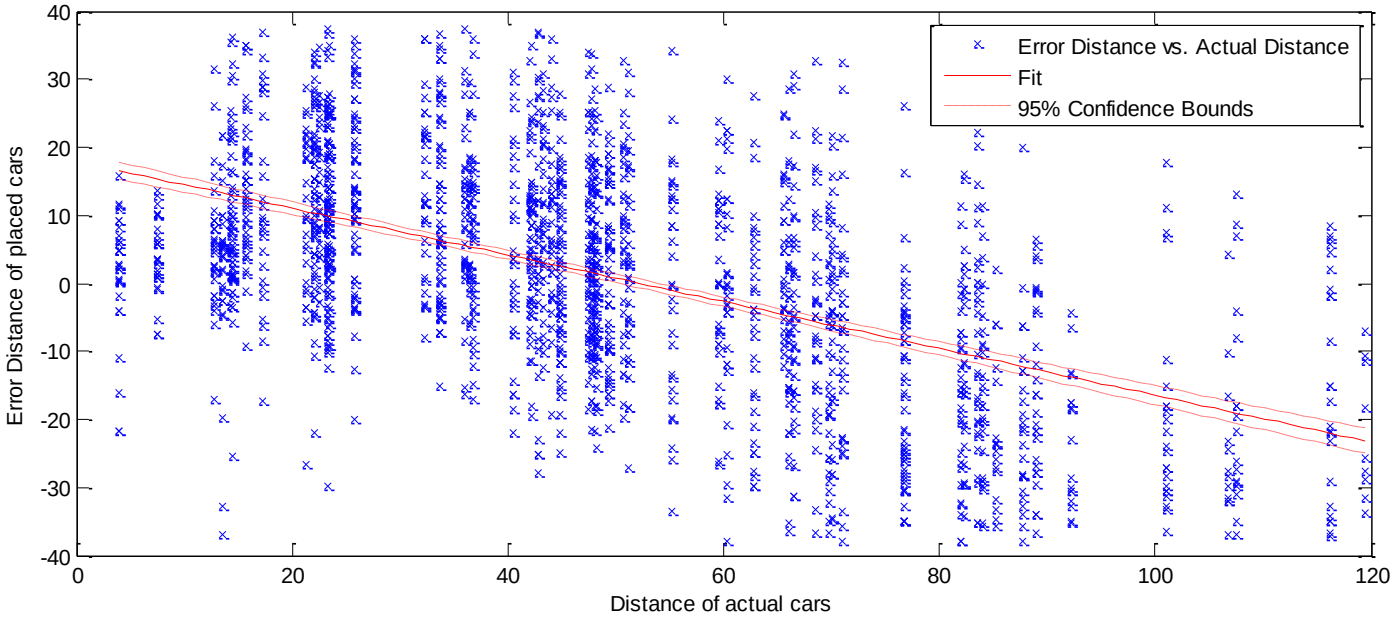


Figure A5-1: Regression of Error distance for distance of actual cars in the front, where a negative error distance means that the actual distance was underestimated.

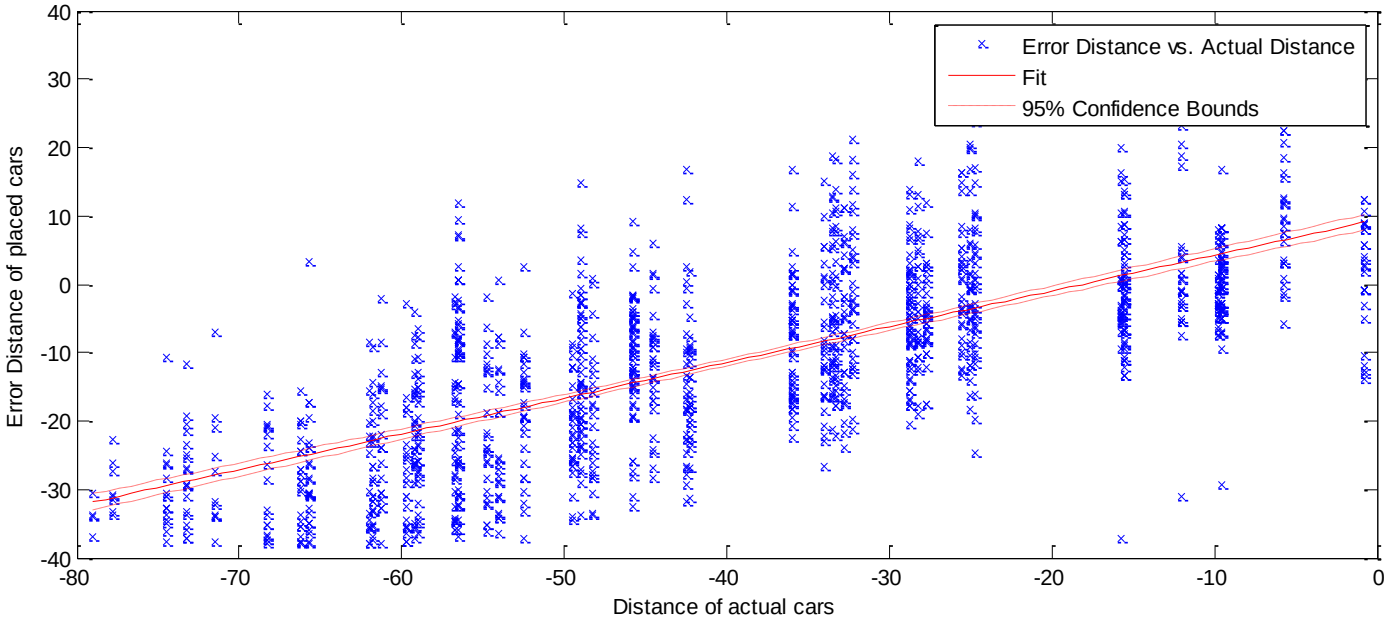


Figure A5-2: Regression of Error distance for distance of actual cars in the rear, where a negative error distance means that the actual distance was underestimated.

Appendix A6: Matlab Codes

Appendix A6-1: GUI (BigGui2.m)

This code provided the interface the participants worked with.

```
%Made By Xander Coster
%Student nr. 1268546

%in case of Matlab stall/crash: turn on or change following lines before restart
% 188, 193 and 196 (DeletedCars, result and vidcount) for click
% 253, 258 and 261 (DeletedCars, result and vidcount) for keypress

function varargout = BigGui2(varargin)
% BIGGUI2 MATLAB code for BigGui2.fig
%      BIGGUI2, by itself, creates a new BIGGUI2 or raises the existing
%      singleton*.
%
%      H = BIGGUI2 returns the handle to a new BIGGUI2 or the handle to
%      the existing singleton*.
%
%      BIGGUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BIGGUI2.M with the given input arguments.
%
%      BIGGUI2('Property','Value',...) creates a new BIGGUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before BigGui2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to BigGui2_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 BigGui2

% Last Modified by GUIDE v2.5 21-May-2015 10:12:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BigGui2_OpeningFcn, ...
                   'gui_OutputFcn',  @BigGui2_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 BigGui2 is made visible.
function BigGui2_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 BigGui2 (see VARARGIN)

% Choose default command line output for BigGui2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
uicontrol(handles.Info1)
% UIWAIT makes BigGui2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = BigGui2_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;
set(hObject, 'units','pixels','outerposition',[0 1080 1920 1200]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% First page / Panel1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --------------------------------------------------------------------
function Panel1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to Panel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Info1_Callback(hObject, eventdata, handles)
% hObject    handle to Info1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Info1 as text
%        str2double(get(hObject,'String')) returns contents of Info1 as a double
PN = str2double(get(hObject,'string'));     %get participant nr from input
assignin('base', 'PN', PN)                  %save participant nr
uicontrol(handles.Info2)

% --- Executes during object creation, after setting all properties.
function Info1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Info1 (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 Info2_Callback(hObject, eventdata, handles)
% hObject    handle to Info2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Info2 as text
%        str2double(get(hObject,'String')) returns contents of Info2 as a double
Age = str2double(get(hObject,'string'));        %get age from input
assignin('base', 'Age', Age)                    %save age
set(handles.PBSave,'visible','on')
uicontrol(handles.PBSave)

% --- Executes during object creation, after setting all properties.
function Info2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Info2 (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 PBSave.
function PBSave_Callback(hObject, eventdata, handles)
% hObject    handle to PBSave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%determine groupnr and video sequence:

PN = evalin('base','PN');
Age = evalin('base','Age');

for i = 1:PN
    randomizer1 = randperm(24)+2;
    randomizer2 = randomizer1(randperm(24));
```

```
    seq = [1 2 randomizer2];
end
% seq = linspace(1,26,26); %for testing purposes

if mod(PN,2)
    %PN is odd, group is 'A'
    Groupnr = 1; %A
    filepath = 'C:\Users\Coster\Dropbox\Afstuderen\Research\Videos\Big Test\A\Video';
else
    %PN is even, group is 'B'
    Groupnr = 2; %B
    filepath = 'C:\Users\Coster\Dropbox\Afstuderen\Research\Videos\Big Test\B\Video';
end

assignin('base','seq',seq)
    assignin('base','Groupnr', Groupnr)
    assignin('base','filepath',filepath)
    load('AnswersBig.mat');
    assignin('base','Answers',AnswersBig2);
    DeletedCars = zeros(26,8);
    DelFile = ['DeletedCars' num2str(PN) '.mat'];
    assignin('base','DelFile',DelFile);
%%%     load((DelFile)); % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%
    assignin('base','DeletedCars',DeletedCars);
    filename = ['Resultsp' int2str(PN) '.mat'];
    assignin('base','filename',filename);
    result = zeros(26,45); %create empty result matrix
%%%     load((filename)) % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%
    result(1:3,1)=[PN Age Groupnr]';
    assignin('base','result',result);
    vidcount = 1;       % Change this after Matlab stall/crach%%%%%%%%%%%%%%%%
    assignin('base', 'vidcount', vidcount)
    set(handles.Panel1,'visible','off')
    set(handles.PBStart,'visible','on')
    uicontrol(handles.PBStart)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Video display %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in PBStart.
function PBStart_Callback(hObject, eventdata, handles)
% hObject    handle to PBStart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f=figure('units','pixels','outerposition',[-10 1040 1940 1280],'Toolbar','none',...
            'Menubar','none','numbertitle','off','Tool','none','Tag','KeyCatch');
wmp = actxcontrol('WMPlayer.ocx.7',[0 0 1940 1280],f); % Create Controller
assignin('base','wmp',wmp);
set(wmp,'uiMode','none');
vidcount = evalin('base','vidcount');
disp(get(hObject,'String'))
seq = evalin('base','seq');
result = evalin('base','result');
filepath = evalin('base','filepath');
filepath = [filepath int2str(seq(vidcount)) '.avi'];
media = wmp.newMedia(filepath); % Create Media object
wmp.CurrentMedia = media;
wmp.Controls.play;
c1 = clock;
pause(2.5)
duration = get(wmp.CurrentMedia,'duration');
while strcmp(wmp.playState,'wmppsPlaying')==1
wmp.playState;
pause(0.01)
if strcmp(wmp.playState,'wmppsPlaying')~=1
    break
end
end
c2 = clock;
result(vidcount,[2,3,33:38,39:44]) = [seq(vidcount) duration c1 c2];
assignin('base','result',result);
close Figure 1
set(handles.PBStart,'visible','off')
set(handles.PosSlider,'Value', 0,'visible','on');
set(handles.LaneSlider,'Value',0,'visible','on');
set(handles.text3,'visible','on');
set(handles.text4,'visible','on');
set(handles.TxtSpeedInstr,'visible','on');
set(handles.text68,'visible','on');
set(handles.text69,'visible','on');
```

```
set(handles.text70,'visible','on');
set(handles.SAPanel,'visible','on')
set(handles.PBNewCar,'visible','on')
set(handles.PBDeleteCar,'visible','off')
set(handles.PBResetCar,'visible','off')
set(handles.AxAnswer,'visible','off')
axes(handles.AxTopview)
Tv = imread('TopviewBig.PNG');
imshow(Tv,'parent',handles.AxTopview);
uistack(handles.AxTopview,'bottom')
hold(handles.AxTopview);
CarCnt = 0;
CarDel = 0;
assignin('base','CarCnt',CarCnt);
assignin('base','CarDel',CarDel);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SAPanel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in PBNewCar.
function PBNewCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBNewCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
CarCnt = evalin('base','CarCnt');
CarCnt = CarCnt+1;
assignin('base','CarCnt',CarCnt);
if CarCnt >= 9
    h = msgbox({'Maximum number' 'of cars' 'reached'}, 'Error','error');
    set(h,'units','normalized','outerposition',[0.3 1.5 .1 .12]);
    CarCnt = 8;
    assignin('base','CarCnt',CarCnt);
else
set(handles.LaneSlider,'Value',0);
set(handles.PosSlider,'Value', 0);
CurrentAx = ['AxCar' num2str(CarCnt)];
CurrentTxt = ['TxtCar' num2str(CarCnt)];
RBSlower = ['RB' num2str(CarCnt) 'Slower'];
RBSame = ['RB' num2str(CarCnt) 'Same'];
RBFaster =['RB' num2str(CarCnt) 'Faster'];
set(handles.(CurrentTxt),'visible','on');
set(handles.(RBSlower),'visible','on');
set(handles.(RBSame),'visible','on');
set(handles.(RBFaster),'visible','on');
set(handles.PBDeleteCar,'visible','on');
set(handles.PBDoneSA,'visible','on');
set(handles.PBFinish,'visible','off');
assignin('base','CurrentAx',CurrentAx);
CarIm = imread('Car2.PNG');
CarFig = imshow(CarIm,'parent',handles.(CurrentAx));
set(CarFig,'Tag',(CurrentAx));
set(CarFig,'ButtonDownFcn',@ImageClickCallback);
set(handles.(CurrentAx),'position',[2.9 14.66 0.26 0.425]);
uistack(handles.(CurrentAx),'top')
text(-60,0,num2str(CarCnt),'backgroundcolor','none','parent',handles.(CurrentAx), 'color','b')
end

function ImageClickCallback (hObject, eventdata, handles)
axesHandle = get(hObject,'Parent');
Pos = get(axesHandle,'Position');
handles = guihandles(BigGui2);
Tag = get(hObject,'Tag');
assignin('base','CurrentAx',Tag);
set(handles.PosSlider,'Value',Pos(2)-14.66);
set(handles.LaneSlider,'Value',Pos(1)-2.9);

% --- Executes on button press in PBDeleteCar.
function PBDeleteCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBDeleteCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
CurrentAx = evalin('base','CurrentAx');
vidcount = evalin('base','vidcount');
CarCnt = evalin('base','CarCnt');
CarDel = evalin('base','CarDel');
CarDel = CarDel + 1;
DeletedCars = evalin('base','DeletedCars');
CarNum = str2double(CurrentAx(6));
if CarDel-1 ~= 0 %not the first deleted car
    if DeletedCars(vidcount,CarDel-1)== CarNum %DeleteCar has been pressed again for the same car
```

```
        CarDel = CarDel -1;
        assignin('base','CarDel',CarDel);
        return
    end
end
CurrentTxt = ['TxtCar' num2str(CarNum)];
CurrentRBSlow = ['RB' num2str(CarNum) 'Slower'];
CurrentRBSame = ['RB' num2str(CarNum) 'Same'];
CurrentRBFast = ['RB' num2str(CarNum) 'Faster'];
set(handles.(CurrentTxt),'visible','off');
set(handles.(CurrentRBSlow),'visible','off','Value',0);
set(handles.(CurrentRBSame),'visible','off','Value',0);
set(handles.(CurrentRBFast),'visible','off','Value',0);
RelVel = 5;
result = evalin('base','result');
result(vidcount,CarNum*3+5) = RelVel;
assignin('base','result',result);
set(handles.PBNewCar,'visible','off');
set(handles.PBResetCar,'visible','on');
assignin('base','CarDel',CarDel);
DeletedCars(vidcount, CarDel) = CarNum;
assignin('base','DeletedCars',DeletedCars);
set(handles.(CurrentAx),'position',[8+0.5*str2double(CurrentAx(6)) 30 0.26 0.425]);
    if CarCnt-CarDel <= 0
        set(hObject, 'visible', 'off');
    end

% --- Executes on button press in PBResetCar.
function PBResetCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBResetCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
DeletedCars = evalin('base','DeletedCars');
vidcount = evalin('base','vidcount');
CarDel = evalin('base','CarDel');
CarNum = DeletedCars(vidcount,CarDel);
Ax = ['AxCar' num2str(CarNum)];
set(handles.(Ax),'position',[2.9 14.66 0.255 0.425]);
CurrentRBSlow = ['RB' num2str(CarNum) 'Slower'];
CurrentRBSame = ['RB' num2str(CarNum) 'Same'];
CurrentRBFast = ['RB' num2str(CarNum) 'Faster'];
set(handles.(CurrentRBSlow),'visible','on');
set(handles.(CurrentRBSame),'visible','on');
set(handles.(CurrentRBFast),'visible','on');
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
DeletedCars(vidcount,CarDel) = 0;
assignin('base','DeletedCars',DeletedCars);
CarDel = CarDel - 1;
assignin('base','CarDel',CarDel);
set(handles.PosSlider,'Value',0);
set(handles.LaneSlider,'Value',0);
assignin('base','CurrentAx',Ax);
if CarDel == 0
    set(handles.PBNewCar,'visible','on');
    set(hObject,'visible','off');
end

%%%%%%%%%% Sliders
% --- Executes on slider movement.
function PosSlider_Callback(hObject, eventdata, handles)
% hObject    handle to PosSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
CurrentAx = evalin('base','CurrentAx');
CarNum = CurrentAx(6);
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
hight = get(hObject,'Value');
lane = get(handles.LaneSlider,'Value');

set(handles.(CurrentAx),'position',[(2.9+lane) (14.66+hight) 0.26 0.425]);

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

```
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function LaneSlider_Callback(hObject, eventdata, handles)
% hObject    handle to LaneSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
CurrentAx = evalin('base','CurrentAx');
CarNum = CurrentAx(6);
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
hight = get(handles.PosSlider,'Value');
lane = get(hObject,'Value');
if lane >= 0.175
    set(hObject,'Value',0.35);
    lane = 0.35;
elseif lane <= -0.175
    set(hObject,'Value',-0.35);
    lane = -0.35;
else
    set(hObject,'Value',0);
    lane = 0;
end
set(handles.(CurrentAx),'position',[(2.9+lane) (14.66+hight) 0.26 0.425]);

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

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

%%%%%%%%%%Radiobuttons
% --- Executes on button press in RB1Slower.
function RB1Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Slower (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.RB1Same,'Value',0);
    set(handles.RB1Faster,'Value',0);
    RelVel1 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,8) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Slower

% --- Executes on button press in RB1Same.
function RB1Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Same (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.RB1Slower,'Value',0);
    set(handles.RB1Faster,'Value',0);
    RelVel1 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,8) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Same

% --- Executes on button press in RB1Faster.
function RB1Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Faster (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.RB1Slower,'Value',0);
    set(handles.RB1Same,'Value',0);
    RelVel1 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,8) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Faster

% --- Executes on button press in RB2Slower.
function RB2Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Slower (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.RB2Same,'Value',0);
    set(handles.RB2Faster,'Value',0);
    RelVel2 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,11) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Slower

% --- Executes on button press in RB2Same.
function RB2Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Same (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.RB2Slower,'Value',0);
    set(handles.RB2Faster,'Value',0);
    RelVel2 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,11) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Same

% --- Executes on button press in RB2Faster.
function RB2Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Faster (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.RB2Slower,'Value',0);
    set(handles.RB2Same,'Value',0);
    RelVel2 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,11) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Faster

% --- Executes on button press in RB3Slower.
function RB3Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Slower (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.RB3Same,'Value',0);
    set(handles.RB3Faster,'Value',0);
    RelVel3 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,14) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Slower

% --- Executes on button press in RB3Same.
function RB3Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Same (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.RB3Slower,'Value',0);
    set(handles.RB3Faster,'Value',0);
    RelVel3 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,14) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Same

% --- Executes on button press in RB3Faster.
function RB3Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Faster (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.RB3Slower,'Value',0);
    set(handles.RB3Same,'Value',0);
    RelVel3 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,14) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Faster

% --- Executes on button press in RB4Slower.
function RB4Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Slower (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.RB4Same,'Value',0);
    set(handles.RB4Faster,'Value',0);
    RelVel4 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,17) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Slower

% --- Executes on button press in RB4Same.
function RB4Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Same (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.RB4Slower,'Value',0);
    set(handles.RB4Faster,'Value',0);
    RelVel4 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,17) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Same

% --- Executes on button press in RB4Faster.
function RB4Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Faster (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.RB4Slower,'Value',0);
    set(handles.RB4Same,'Value',0);
    RelVel4 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,17) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Faster

% --- Executes on button press in RB5Slower.
function RB5Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Slower (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.RB5Same,'Value',0);
    set(handles.RB5Faster,'Value',0);
    RelVel5 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,20) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Slower

% --- Executes on button press in RB5Same.
function RB5Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Same (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.RB5Slower,'Value',0);
    set(handles.RB5Faster,'Value',0);
    RelVel5 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,20) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Same

% --- Executes on button press in RB5Faster.
function RB5Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Faster (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.RB5Slower,'Value',0);
    set(handles.RB5Same,'Value',0);
    RelVel5 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,20) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Faster

% --- Executes on button press in RB6Slower.
function RB6Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Slower (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.RB6Same,'Value',0);
    set(handles.RB6Faster,'Value',0);
    RelVel6 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,23) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Slower

% --- Executes on button press in RB6Same.
function RB6Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Same (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.RB6Slower,'Value',0);
    set(handles.RB6Faster,'Value',0);
    RelVel6 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,23) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Same

% --- Executes on button press in RB6Faster.
function RB6Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Faster (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.RB6Slower,'Value',0);
```

```
    set(handles.RB6Same,'Value',0);
    RelVel6 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,23) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Faster

% --- Executes on button press in RB7Slower.
function RB7Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Slower (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.RB7Same,'Value',0);
    set(handles.RB7Faster,'Value',0);
    RelVel7 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,26) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Slower

% --- Executes on button press in RB7Same.
function RB7Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Same (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.RB7Slower,'Value',0);
    set(handles.RB7Faster,'Value',0);
    RelVel7 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,26) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Same

% --- Executes on button press in RB7Faster.
function RB7Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Faster (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.RB7Slower,'Value',0);
    set(handles.RB7Same,'Value',0);
    RelVel7 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,26) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Faster

% --- Executes on button press in RB8Slower.
function RB8Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Slower (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.RB8Same,'Value',0);
    set(handles.RB8Faster,'Value',0);
    RelVel8 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,29) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Slower

% --- Executes on button press in RB8Same.
function RB8Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Same (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.RB8Slower,'Value',0);
    set(handles.RB8Faster,'Value',0);
```

```
    RelVel8 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,29) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Same

% --- Executes on button press in RB8Faster.
function RB8Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Faster (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.RB8Slower,'Value',0);
    set(handles.RB8Same,'Value',0);
    RelVel8 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,29) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Faster

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

% if nothing is filled in or if filled in incompletely, give warning message
% and get out
vidcount = evalin('base','vidcount');
result = evalin('base','result');
CarCnt = evalin('base','CarCnt');
CarDel = evalin('base','CarDel');
filename = evalin('base','filename');

if CarCnt == 0 || CarCnt-CarDel <= 0
    h = msgbox({'No cars were placed'}, 'Error','error');
    set(h,'units','normalized','outerposition',[0.45 1.6 .1 .12]);
    return
end

for i = 1:CarCnt
    CurrentTxt = ['TxtCar' num2str(i)];
    CurrentRBSlow = ['RB' num2str(i) 'Slower'];
    CurrentRBSame = ['RB' num2str(i) 'Same'];
    CurrentRBFast = ['RB' num2str(i) 'Faster'];
    if strcmp(get(handles.(CurrentTxt),'visible'),'on') && ...
            get(handles.(CurrentRBSlow),'Value') == 0 && ...
            get(handles.(CurrentRBSame),'Value') == 0 && ...
            get(handles.(CurrentRBFast),'Value') == 0

        h = msgbox({'Please indicate' 'the relative speed' ['of car' num2str(i)]}, 'Error','error');
        set(h,'units','normalized','outerposition',[0.45 1.6 .1 .12]);
        return
    end
end
for i = 1:CarCnt
    CurrentTxt = ['TxtCar' num2str(i)];
    CurrentRBSlow = ['RB' num2str(i) 'Slower'];
    CurrentRBSame = ['RB' num2str(i) 'Same'];
    CurrentRBFast = ['RB' num2str(i) 'Faster'];
    RelVel = result(vidcount,i*3+5);
    CurrentAx = ['AxCar' num2str(i)];
    CarIm = imread('Car2.PNG');
    CarFig = imshow(CarIm,'parent',handles.(CurrentAx));

    if RelVel == -1
    text(-80,0,{'<'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
    elseif RelVel == 0
    text(-80,0,{'='
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
    elseif RelVel == 1
        text(-80,0,{'>'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
    end
```

```
    %%%%Save positions and velocities
    NumP = CarCnt - CarDel; %number of placed cars - deleted cars
    allNumP = CarCnt; %just placed cars
    AX = ['AxCar' num2str(i)];
    Pos = get(handles.(AX),'position');
    Pos(1) = Pos(1)-2.9;
    Pos(2) = Pos(2)-14.66;
    result(vidcount,4) = allNumP;
    result(vidcount,5) = NumP;
    result(vidcount,[i*3+3 i*3+4]) = [Pos(1) Pos(2)];
    assignin('base','result',result)
    save((filename),'result');

%%%%
    set(handles.(CurrentTxt),'visible','off');
    set(handles.TxtSpeedInstr,'visible','off');
    set(handles.text68,'visible','off');
    set(handles.text69,'visible','off');
    set(handles.text70,'visible','off');
    set(handles.(CurrentRBSlow),'visible','off','value',0);
    set(handles.(CurrentRBSame),'visible','off','value',0);
    set(handles.(CurrentRBFast),'visible','off','value',0);
end
    set(hObject,'visible','off');
    set(handles.PBNewCar,'visible','off');
    set(handles.PBDeleteCar,'visible','off');
    set(handles.PBResetCar,'visible','off');
    set(handles.PosSlider,'visible','off');
    set(handles.LaneSlider,'visible','off');
    set(handles.text3,'visible','off');
    set(handles.text4,'visible','off');
    set(handles.TxtYourAnswer,'visible','on');
    set(handles.MentSlid,'Value',50);
    set(handles.SafeSlid,'Value',50);
    set(handles.TimeSlid,'Value',50);
    set(handles.TLXPanel,'visible','on');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TLX Panel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on slider movement.
function MentSlid_Callback(hObject, eventdata, handles)
% hObject    handle to MentSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Mental = get(hObject,'Value');
Mental = roundn(Mental,-1);
set(hObject,'Value',Mental)
vidcount = evalin('base','vidcount');
result = evalin('base','result');
result(vidcount,30) = Mental;
assignin('base','result',result);

% --- Executes during object creation, after setting all properties.
function MentSlid_CreateFcn(hObject, eventdata, ~)
% hObject    handle to MentSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function SafeSlid_Callback(hObject, eventdata, handles)
% hObject    handle to SafeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Safety = get(hObject,'Value');
Safety = roundn(Safety,-1);
set(hObject,'Value',Safety)
vidcount = evalin('base','vidcount');
```

```
result = evalin('base','result');
result(vidcount,31) = Safety;
assignin('base','result',result);
% --- Executes during object creation, after setting all properties.
function SafeSlid_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SafeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function TimeSlid_Callback(hObject, eventdata, handles)
% hObject    handle to TimeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Temporal = get(hObject,'Value');
Temporal = roundn(Temporal,-1);
set(hObject,'Value',Temporal)
vidcount = evalin('base','vidcount');
result = evalin('base','result');
result(vidcount,32) = Temporal;
assignin('base','result',result);
% --- Executes during object creation, after setting all properties.
function TimeSlid_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TimeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in PBDoneTLX.
function PBDoneTLX_Callback(hObject, eventdata, handles)
% hObject    handle to PBDoneTLX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
vidcount=evalin('base','vidcount');
Answers = evalin('base','Answers');
PosSelf = 860;
seq = evalin('base','seq');
Answer = Answers(seq(vidcount),:);
NumOCars = Answer(1);
assignin('base','NumOCars',NumOCars);

    filename = evalin('base','filename');
    result = evalin('base','result');
    result(vidcount,30)=get(handles.MentSlid,'Value');
    result(vidcount,31)=get(handles.SafeSlid,'Value');
    result(vidcount,32)=get(handles.TimeSlid,'Value');
    assignin('base','result',result);
    save((filename),'result');
    PN = evalin('base','PN');
    save('PN.mat','PN')
    DeletedCars = evalin('base','DeletedCars');
    DelFile = evalin('base','DelFile');
    save(DelFile,'DeletedCars');

    % Show answer
    set(handles.AxAnswer,'visible','on');
    axes(handles.AxAnswer);
    Tv = imread('TopviewBig.PNG');
    imshow(Tv,'parent',handles.AxAnswer);
    uistack(handles.AxAnswer,'bottom')
    hold(handles.AxAnswer);

        for i = 1:NumOCars
            CurrentAx = ['AxAns' num2str(i)];
            CarIm = imread('Car2.PNG');
            CarFig = imshow(CarIm,'parent',handles.(CurrentAx));
            Lanepos = (Answer(3*i+2)-PosSelf)/10;
            Lane = Answer(3*i+1)*0.35;
```

```
            set(handles.(CurrentAx),'position',[(17.47+Lane) Lanepos+14.66 0.255 0.425]);
            if Answer(3*i+3) > 28
text(-80,0,'>','backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
            elseif Answer (3*i+3) == 28
text(-80,0,'=','backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
            elseif Answer (3*i+3) < 28
text(-80,0,'<','backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize',14)
            end
        end
    set(handles.TxtActual,'visible','on');
    set(handles.TLXPanel,'visible','off');
    set(handles.PBFinish,'visible','on');
    uicontrol(handles.PBFinish)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in PBFinish.
function PBFinish_Callback(hObject, eventdata, handles)
% hObject    handle to PBFinish (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename = evalin('base','filename');
result = evalin('base','result');
save((filename),'result');

vidcount = evalin('base','vidcount');
NumOCars = evalin('base','NumOCars');

for j = 1:8
    AX = ['AxCar' num2str(j)];
    set(handles.AxAnswer,'visible','off');
    set(handles.(AX),'position',[8+0.5*j 30 0.252 0.42]);
    axes(handles.(AX));
    cla reset
    set(handles.(AX),'visible','off');
end

for k = 1:NumOCars
    CurrentAx = ['AxAns' num2str(k)];
    set(handles.(CurrentAx),'position',[12+0.5*k 30 0.252 0.42]);
    axes(handles.(CurrentAx));
    cla reset
    set(handles.(CurrentAx),'visible','off');
end
    set(handles.SAPanel,'visible','off');
    axes(handles.AxAnswer);
    cla reset
    set(handles.AxAnswer,'visible','off');
    set(handles.TxtYourAnswer,'visible','off');
    set(handles.TxtActual,'visible','off');
    set(hObject,'visible','off');

vidcount = vidcount+1;
assignin('base','vidcount',vidcount);
CarCnt = 0;
assignin('base','CarCnt',CarCnt);
if vidcount >= 27
    set(handles.text71,'visible','on');
    pause(5)
    close all

elseif vidcount == 2
    set(handles.PBStart,'String','Start Training 2');
    set(handles.PBStart,'visible','on');
    uicontrol(handles.PBStart)
else
    Num = vidcount - 2;
    set(handles.PBStart,'String',['Start Video ' num2str(Num)]);
    set(handles.PBStart,'visible','on');
    uicontrol(handles.PBStart)
end
```

Appendix A6-2: Matching algorithm (AdvancedMeasures5x.m)

This code was used to make matches between placed cars and actual cars and to calculate the advanced scores from these matches.

```
% Made by Xander Coster
% 1268546
```

```
% For calculating advanced scores

%% 1) Load and set constants
% load('AnswersBig.mat');
load('AllData.mat')

% load('PN.mat'); %total number of participants
PN = 34;
MaxDist = 38;       % maximal distance within which a car is eligible for a 'hit'
wa_Front_NonABS = -0.3439;
wb_Front_NonABS = 17.94;
wa_Rear_NonABS = 0.5229;
wb_Rear_NonABS = 9.478;
wa_Front = 0.08753;           %   factor for weighing factor for distance: e_d_i = wa*d_i + wb
wb_Front = 9.226;             % constant for ""
wa_Rear = -0.3649;
wb_Rear = -0.0123;
% Choose threshold for close by/far away
FrontDist = 60;
RearDist = -40;

%% 2 Make empty matrices, used for plotting results and interim calculations
FA_M = nan(24,34);        %Number of False Alarms + Misses
Hits = nan(24,34);        %Number of Hits
FA = nan(24,34);          %False for NumP > NumA
Miss = nan(24,34);        %Misses for NumP < NumA
False = nan(24,34);       %Number of False Alarms
Misses = nan(24,34);      %Number of Misses
SAScores_Raw_Scrambled = nan(24,34);
SAScores_Raw_Combined_Scrambled = nan(24,34);
SAScores_Scrambled = nan(24,34);
SAScoresABS_Scrambled = nan(24,34);
SAScoresNonABS_Scrambled = nan(24,34);
SAScores_Combined_Ordered = nan(24,34);
SAScoresABS_Combined_Ordered = nan(24,34);
SAScoresNonABS_Combined_Ordered = nan(24,34);
SAScores_Combined_Scrambled = nan(24,34);
SAScoresABS_Combined_Scrambled = nan(24,34);
SAScoresNonABS_Combined_Scrambled = nan(24,34);
HitPerc_Ordered = nan(24,34);     %Number of hits/number of actual cars
HitPerc_Scrambled = nan(24,34);
MFAPerc_Scrambled = nan(24,34);
CombSAScores = nan(24,34);%Combined measure SA score
Num = zeros(24,34);         %Number of cars that are tried to be matched (greatest of NumA and NumP)
AllLanesP = nan(24,272); AllDistsP = nan(24,272); AllSpeedP = nan(24,272); AllSpeedsA = nan(24,6);
ErrorDists_Raw_Ordered = nan(24,34);
ErrorDists_Raw_Scrambled = nan(24,34);
ErrorDistsNonABS_Raw_Ordered =  nan(24,34);
ErrorDistsNonABS_Raw_Scrambled = nan(24,34);
ErrorDists_Ordered = nan(24,34);
ErrorDistsABS_Ordered = nan(24,34);
ErrorDistsNonABS_Ordered = nan(24,34);
ErrorMiss_Ordered = nan(24,34);
ErrorDists_Scrambled = nan(24,34);
ErrorLaneNum = nan(24,34);  %number of errors in lanes [#]
ErrorLaneDist = nan(24,34); %sum of error distance of lanes [# of lanes]
ErrorSpeedNum = nan(24,34); %Number of errors in speeds [#]
ErrorSpeed = nan(24,34);
BestComb = nan(24,8);
BestComb_All = nan(24,272);
Posses = nan(192,272);
Dists = nan(192,272);
Errorses = nan(192,272);
ErrorDist_PerMatch_Front = nan(24,272);
ErrorDist_PerMatch_Rear = nan(24,272);
ErrorSpeed_PerMatch = nan(24,272);
ErrorDistNonABS_PerMatch_Front = nan(24,272);
ErrorDistNonABS_PerMatch_Rear = nan(24,272);
ErrorDistNonABS_PerMatch = nan(24,272);
ErrD_Front = nan(500,12);
ErrD_Rear = nan(500,8);
DA_Front = nan(500,12);
DA_Rear = nan(500,8);
DistA_PerMatch = nan(24,272); LaneA_PerMatch = nan(24,272); SpeedA_PerMatch = nan(24,272);
DistA_PerMatch_Front = nan(24,272); DistA_PerMatch_Rear = nan(24,272);
DistP_PerMatch = nan(24,272); LaneP_PerMatch = nan(24,272); SpeedP_PerMatch = nan(24,272);
DistA_PerMiss = nan(24,272);
ErrorDist_PerMiss = nan(24,272);
ErrorDist_Expected_PerMatch = nan(24,272);
ErrorDistNonABS_Expected_PerMatch = nan(24,272);
ErrorDist_Expected_PerMiss = nan(24,272);
Hits_Duration_All = nan(136,6); Hits_Duration_All_6cars = nan(68,6); Hits_Duration_All_4cars = nan(68,6);
```

```
HitPerc_Duration_All = nan(136,6); HitPerc_Duration_All_6cars = nan(68,6); HitPerc_Duration_All_4cars =
nan(68,6);
MFAPerc_Duration_All = nan(136,6); MFAPerc_Duration_All_6cars = nan(68,6); MFAPerc_Duration_All_4cars =
nan(68,6);
Misses_Duration_All = nan(136,6); Misses_Duration_All_6cars = nan(68,6); Misses_Duration_All_4cars =
nan(68,6);
False_Duration_All = nan(136,6); False_Duration_All_6cars = nan(68,6); False_Duration_All_4cars = nan(68,6);
FA_M_Duration_All = nan(136,6); FA_M_Duration_All_6cars = nan(68,6); FA_M_Duration_All_4cars = nan(68,6);
SAScores_Raw_Duration_All = nan(136,6); SAScores_Raw_Duration_All_6cars = nan(68,6);
SAScores_Raw_Duration_All_4cars = nan(68,6);
SAScores_Raw_Combined_Duration_All = nan(136,6); SAScores_Raw_Combined_Duration_All_6cars = nan(68,6);
SAScores_Raw_Combined_Duration_All_4cars = nan(68,6);
SAScores_Duration_All = nan(136,6); SAScores_Duration_All_6cars = nan(68,6); SAScores_Duration_All_4cars =
nan(68,6);
SAScoresABS_Duration_All = nan(136,6); SAScoresABS_Duration_All_6cars = nan(68,6);
SAScoresABS_Duration_All_4cars = nan(68,6);
SAScoresNonABS_Duration_All = nan(136,6); SAScoresNonABS_Duration_All_6cars = nan(68,6);
SAScoresNonABS_Duration_All_4cars = nan(68,6);
SAScores_Combined_Duration_All = nan(136,6); SAScores_Combined_Duration_All_6cars = nan(68,6);
SAScores_Combined_Duration_All_4cars = nan(68,6);
SAScoresABS_Combined_Duration_All = nan(136,6); SAScoresABS_Combined_Duration_All_6cars = nan(68,6);
SAScoresABS_Combined_Duration_All_4cars = nan(68,6);
SAScoresNonABS_Combined_Duration_All = nan(136,6); SAScoresNonABS_Combined_Duration_All_6cars = nan(68,6);
SAScoresNonABS_Combined_Duration_All_4cars = nan(68,6);
ErrorLaneDist_Duration_All = nan(136,6); ErrorLaneDist_Duration_All_6cars = nan(68,6);
ErrorLaneDist_Duration_All_4cars = nan(68,6);
ErrorLaneDistPerc_Duration_All = nan(136,6); ErrorLaneDistPerc_Duration_All_6cars = nan(68,6);
ErrorLaneDistPerc_Duration_All_4cars = nan(68,6);
ErrorLaneNum_Duration_All = nan(136,6); ErrorLaneNum_Duration_All_6cars = nan(68,6);
ErrorLaneNum_Duration_All_4cars = nan(68,6);
LaneErrorPerc_Duration_All = nan(136,6); LaneErrorPerc_Duration_All_6cars = nan(68,6);
LaneErrorPerc_Duration_All_4cars = nan(68,6);
ErrorSpeed_Duration_All = nan(136,6); ErrorSpeed_Duration_All_6cars = nan(68,6);
ErrorSpeed_Duration_All_4cars = nan(68,6);
ErrorSpeedPerc_Duration_All = nan(136,6); ErrorSpeedPerc_Duration_All_6cars = nan(68,6);
ErrorSpeedPerc_Duration_All_4cars = nan(68,6);
ErrorSpeedNum_Duration_All = nan(136,6); ErrorSpeedNum_Duration_All_6cars = nan(68,6);
ErrorSpeedNum_Duration_All_4cars = nan(68,6);
SpeedErrorPerc_Duration_All = nan(136,6); SpeedErrorPerc_Duration_All_6cars = nan(68,6);
SpeedErrorPerc_Duration_All_4cars = nan(68,6);

countF1 = 0; countF2 = 0; countF3 = 0; countF4 = 0; countF5 = 0; countF6 = 0; countF7 = 0; countF8 = 0;
countF9 = 0; countF10 = 0; countF11 = 0; countF12 = 0;
countR1 = 0; countR2 = 0; countR3 = 0; countR4 = 0; countR5 = 0; countR6 = 0; countR7 = 0; countR8 = 0;
%% 3) Fill empty matrices and calculate the different results
for i = 1:PN
% seq = seq_i(:,i);

    % determine the SA scores for each separate video
for j = 1:24
%     PosSelf = 860; %position of own car
    NumA = NumA_Ordered(j+2);
    NumP = NumP_Ordered_All(j+2,i);         %number of placed cars - deleted cars
    AllNumP = AllNumP_Ordered_All(j+2,i);   %placed cars & deleted cars

    if NumA >= NumP
        PossibleMatch = zeros(NumA);
        DistMatch = zeros(NumA);
        LaneMatch = zeros(NumA);
        SpeedMatch = zeros(NumA);
        Errors = zeros(NumA);
        Num(j,i) = NumA;
        Miss(j,i) = NumA - NumP;
        FA(j,i) = 0;
    else
        if isnan(NumP) == 0
        PossibleMatch = zeros(NumP);
        DistMatch = zeros(NumP);
        LaneMatch = zeros(NumP);
        SpeedMatch = zeros(NumP);
        Errors = zeros(NumP);
        Num(j,i) = NumP;
        FA(j,i) = NumP - NumA;
        Miss(j,i) = 0;
        end
    end

    DistA = DistA_Ordered(j+2,1:NumA);
    LaneA = LaneA_Ordered(j+2,1:NumA);
    SpeedA = SpeedA_Ordered(j+2,1:NumA);
    for k = 1:NumA
       if SpeedA(k) == 25.5
```

```
            SpeedA(k) = -2;
        elseif SpeedA(k) == 26.75
            SpeedA(k) = -1;
        elseif SpeedA(k) == 28
            SpeedA(k) = 0;
        elseif SpeedA(k) == 29.25
            SpeedA(k) = 1;
        elseif SpeedA(k) == 30.5
            SpeedA(k) = 2;
        end
    end
        if isnan(NumP)
                LaneDistCount = nan;
        end
    AllSpeedsA(j,1:NumA) = SpeedA;
        if isnan(NumP) == 0
    DelNum = AllNumP-NumP;      %number of deleted cars
    DistP = DistP_Ordered_All(j+2,(i-1)*8+1:(i-1)*8+AllNumP);
    LaneP = round(LaneP_Ordered_All(j+2,(i-1)*8+1:(i-1)*8+AllNumP)/0.35);
    SpeedP = SpeedP_Ordered_All(j+2,(i-1)*8+1:(i-1)*8+AllNumP);
    PCars = linspace(1,AllNumP,AllNumP);
    LaneDistCount = 0;
    SpeedCount = 0;

    if DelNum ~= 0
    DelP = AllDel_Ordered(j+2,(i-1)*8+1:(i-1)*8+8);   %array with carnumbers of deleted cars and zeros
    Del = zeros(1,DelNum);
        for p = 1:DelNum
            Del(p) = DelP(p) ;%array with only carnumbers of deleted cars
        end

    %delete entries of deleted cars with Del
    DistP(Del) = [];   %relative longitudinal distance of placed, non-deleted cars [m]
    LaneP(Del) = [];   %lane of placed, non-deleted cars  (-1, 0 or 1)
    SpeedP(Del) = [];  %relateve speed of placed, non-deleted cars (-1, 0 or 1)
    PCars(Del) = [];
     end
        AllLanesP(j,(i-1)*8+1:(i-1)*8+NumP)=LaneP;
        AllDistsP(j,(i-1)*8+1:(i-1)*8+NumP)=DistP;
        AllSpeedP(j,(i-1)*8+1:(i-1)*8+NumP)=SpeedP;
        PossibleMatch(1:NumA,1:NumP)= repmat(PCars,NumA,1);

   for a = 1:NumA
       for b = 1:NumP
           DistMatch(a,b) = abs(DistA(a)-DistP(b)); %actual error distances
             for k = 1:size(DistMatch,1)
                 for l = 1:size(DistMatch,2)
            if DistMatch(k,l) > MaxDist  || 0   %determine possible matches to each Acar
                DistMatch(k,l) = MaxDist;       %misses and false alarms get a distance of MaxDist
                PossibleMatch(a,b) = 0;         %outside MaxDist no possible match
            end
                 end
             end

Posses((j-1)*8+1:(j-1)*8+Num(j,i),(i-1)*8+1:(i-1)*8+Num(j,i)) = PossibleMatch;
            LaneMatch(a,b) = roundn(abs(LaneA(a)-LaneP(b)),0);

            if LaneMatch(a,b) == 0
                LaneMatch(a,b) = 0;
            elseif LaneMatch(a,b) == 1
                LaneMatch(a,b) = 3.5;
            elseif LaneMatch(a,b) == 2
                LaneMatch(a,b) = 7;
            end
Errors(a,b) = DistMatch(a,b)+5*LaneMatch(a,b);
            for c = 1:size(PossibleMatch,1)
                for d = 1:size(PossibleMatch,2)
            if PossibleMatch(c,d) == 0
                Errors(c,d) = 73;
            end
            Errorses((j-1)*8+1:(j-1)*8+Num(j,i),(i-1)*8+1:(i-1)*8+Num(j,i)) = Errors;
            Dists((j-1)*8+1:(j-1)*8+Num(j,i),(i-1)*8+1:(i-1)*8+Num(j,i)) = DistMatch;
                end
            end
       end
   end
if isnan(DistMatch(:,1:NumP))
    PossibleMatch = zeros(size(PossibleMatch));
end

% %for each A see how many P's can match, P's and A's can only be matched once
% %             C = A((perms(1:N) - 1) * N + meshgrid(1:N, 1:factorial(N)))
```

```
N = size(PossibleMatch, 1);
X = perms(1:N);                     % # Permuations of column indices
Y = meshgrid(1:N, 1:factorial(N)); % # Row indices
idx = (X - 1) * N + Y;              % # Convert to linear indexing
Comb = PossibleMatch(idx);          % # Extract combinations
comb = unique(Comb,'rows');
S = zeros(size(comb,1),1);
ErrorN = zeros(size(comb));

% get specific error from combinations
for a = 1:size(comb,1)
    error = 0;
    for b = 1:size(comb,2)
        if comb(a,b) == 0
            error = 73;
        else
            colError = find(PossibleMatch(b,:) == comb(a,b));
            error = Errors(b, colError);
        end
        ErrorN(a,b)=error;
    end
    S(a) = sum(ErrorN(a,:));    %error scores for every possible combination
end

[BestError, BestIndex] = min(S(:));
combi = comb(BestIndex,:);  %combination of best match
FA_M(j,i) = sum(combi(:)==0); %total nr of false alarms and misses

% find combination of best match
BestComb(j,1:length(combi)) = combi;
BestComb_All(j,(i-1)*8+1:(i-1)*8+8) = BestComb(j,:);
% [row, col] = ind2sub(size(S), MinIndex)

% calculate number of hits, misses, and false alarms
Hits(j,i) = length(combi)-sum(combi(:)==0);
if Hits(j,i) == 0
    Hits(j,i) = nan;
end
Misses(j,i) = NumA - Hits(j,i);
False(j,i) = NumP - Hits(j,i);
% calculate hit percentage (Hits/NumA)
HitPerc_Ordered(j,i) = Hits(j,i)/NumA_Ordered(j+2);

% calculate average error distance per hit
errordist_raw = 0;
errordist_Corr = 0;

for c = 1: length(combi)
%     combi(combi==0)=nan;
    if combi(c) ~= 0
    colDistMatch = find(PossibleMatch(c,:) == combi(c));
    DistA_PerMatch(j,(i-1)*8+c) = DistA(c);
    LaneA_PerMatch(j,(i-1)*8+c) = LaneA(c);
    SpeedA_PerMatch(j,(i-1)*8+c) = SpeedA(c);
    DistP_PerMatch(j,(i-1)*8+c) = DistP(colDistMatch);
    LaneP_PerMatch(j,(i-1)*8+c) = LaneP(colDistMatch);
    SpeedP_PerMatch(j,(i-1)*8+c) = SpeedP(colDistMatch);
    if sign(SpeedA_PerMatch(j,(i-1)*8+c)) == sign(SpeedP_PerMatch(j,(i-1)*8+c));
        ErrorSpeed_PerMatch(j,(i-1)*8+c) = 0;
    else
        ErrorSpeed_PerMatch(j,(i-1)*8+c) = abs(SpeedA_PerMatch(j,(i-1)*8+c) - SpeedP_PerMatch(j,(i-1)*8+c));
    end
    if ErrorSpeed_PerMatch(j,(i-1)*8+c) ~= 0 && isnan(ErrorSpeed_PerMatch(j,(i-1)*8+c)) == 0
        SpeedCount = SpeedCount + 1;
    end
ErrorSpeed(j,i) = nansum(ErrorSpeed_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
ErrorLaneDist_PerMatch = abs(LaneA_PerMatch-LaneP_PerMatch);
    if ErrorLaneDist_PerMatch(j,(i-1)*8+c) ~= 0 && isnan(ErrorLaneDist_PerMatch(j,(i-1)*8+c)) == 0
    LaneDistCount = LaneDistCount + 1;
    end
    elseif combi(c) == 0 && c <= NumA
        DistA_PerMiss(j,(i-1)*8+c) = DistA(c);
        ErrorDist_PerMiss(j,(i-1)*8+c) = MaxDist;
        if DistA(c) >= 0
        ErrorDist_Expected_PerMiss(j,(i-1)*8+c) = wa_Front*DistA(c) + wb_Front;
        end
        if DistA(c) <=0
        ErrorDist_Expected_PerMiss(j,(i-1)*8+c) = wa_Rear*DistA(c) + wb_Rear;
        end
    end
end
ErrorLaneDist(j,i) = nansum(ErrorLaneDist_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
```

```
        end
ErrorLaneNum(j,i) = LaneDistCount;
ErrorLaneNum(isnan(Hits)) = nan;
ErrorSpeedNum(j,i) = SpeedCount;
ErrorSpeedNum(isnan(Hits)) = nan;
end
end

% make sure deleted data is deleted in every variable
ErrorDist_PerMatch = abs(DistA_PerMatch-DistP_PerMatch)+ErrorLaneDist_PerMatch*3.5;
DistA_PerMatch(isnan(ErrorDist_PerMatch)) = nan;
LaneA_PerMatch(isnan(ErrorDist_PerMatch)) = nan;
SpeedA_PerMatch(isnan(ErrorDist_PerMatch)) = nan;
DistA_PerMatch_Front(isnan(ErrorDist_PerMatch_Front)) = nan;
DistA_PerMatch_Rear(isnan(ErrorDist_PerMatch_Rear)) = nan;
BestComb_All(isnan(ErrorDist_PerMatch)) = nan;
ErrorDist_Corr_PerMatch = ErrorDist_PerMatch - ErrorDist_Expected_PerMatch;
ErrorDistABS_Corr_PerMatch = abs(ErrorDist_PerMatch - ErrorDist_Expected_PerMatch);
ErrorDistNonABS_Corr_PerMatch = abs(ErrorDistNonABS_PerMatch - ErrorDistNonABS_Expected_PerMatch);
ErrorDist_Corr_PerMiss = ErrorDist_PerMiss - ErrorDist_Expected_PerMiss;
ErrorDist_Corr_PerMiss(ErrorDist_Corr_PerMiss == 0) = nan;
ErrorLaneDist(isnan(Hits)) = nan;
% ErrorDist_Corr_PerMatch(isnan(ErrorDist_PerMatch)) = nan;
% ErrorDist_Corr_PerMiss(isnan(ErrorDist_PerMatch)) = nan;
MFAPerc_Ordered = (Misses+FA)./repmat(NumA_Ordered(3:end),1,34);
%% Splitting/grouping data for plotting
LaneErrorPerc = ErrorLaneNum ./ Hits;
ErrorLaneDistPerc = ErrorLaneDist ./ Hits;
SpeedErrorPerc = ErrorSpeedNum ./ Hits;
ErrorSpeedPerc = ErrorSpeed ./ Hits;

count1_A = 0; count1_A_4 = 0; count1_A_6 = 0; count1_B = 0; count1_B_4 = 0; count1_B_6 = 0;
count3_A = 0; count3_A_4 = 0; count3_A_6 = 0; count3_B = 0; count3_B_4 = 0; count3_B_6 = 0;
count7_A = 0; count7_A_4 = 0; count7_A_6 = 0; count7_B = 0; count7_B_4 = 0; count7_B_6 = 0;
count9_A = 0; count9_A_4 = 0; count9_A_6 = 0; count9_B = 0; count9_B_4 = 0; count9_B_6 = 0;
count12_A = 0; count12_A_4 = 0; count12_A_6 = 0; count12_B = 0; count12_B_4 = 0; count12_B_6 = 0;
count20_A = 0; count20_A_4 = 0; count20_A_6 = 0; count20_B = 0; count20_B_4 = 0; count20_B_6 = 0;
for i = 1:34
for j = 1:24
ErrorDists_Raw_Ordered(j,i) = nansum(ErrorDist_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
ErrorDists_Raw_Ordered(ErrorDists_Raw_Ordered == 0) = nan;
ErrorDistsNonABS_Raw_Ordered(j,i) = nansum(abs(ErrorDistNonABS_PerMatch(j,(i-1)*8+1:(i-1)*8+8)));
ErrorDistsNonABS_Raw_Ordered(ErrorDistsNonABS_Raw_Ordered == 0) = nan; %same as 'normal'
ErrorDists_Raw_Ordered

SAScores_Raw_Ordered= ErrorDists_Raw_Ordered./Hits;
% SAScoresNonABS_Raw_Ordered =
SAScores_Raw_Combined_Ordered = (ErrorDists_Raw_Ordered+MaxDist*Misses)./(Hits+Misses);
ErrorDists_Raw_Scrambled(j,i) = ErrorDists_Raw_Ordered(seq_i(j+2,i)-2,i);
SAScores_Raw_Scrambled(j,i) = SAScores_Raw_Ordered((seq_i(j+2,i))-2,i);
SAScores_Raw_Combined_Scrambled(j,i) = SAScores_Raw_Combined_Ordered(seq_i(j+2,i)-2,i);
HitPerc_Scrambled(j,i) = HitPerc_Ordered(seq_i(j+2,i)-2,i);
MFAPerc_Scrambled(j,i) = MFAPerc_Ordered(seq_i(j+2,i)-2,i);
ErrorDists_Ordered(j,i) = nansum(ErrorDist_Corr_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
ErrorDistsABS_Ordered(j,i) = nansum(ErrorDistABS_Corr_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
ErrorDistsNonABS_Ordered(j,i) = nansum(ErrorDistNonABS_Corr_PerMatch(j,(i-1)*8+1:(i-1)*8+8));
ErrorDists_Ordered(ErrorDists_Ordered == 0) = nan;
ErrorMiss_Ordered(j,i) = nansum(ErrorDist_Corr_PerMiss(j,(i-1)*8+1:(i-1)*8+8));
ErrorMiss_Ordered(ErrorMiss_Ordered == 0) = nan;
SAScores_Ordered = ErrorDists_Ordered./Hits;
SAScoresABS_Ordered = ErrorDistsABS_Ordered./Hits;
SAScoresNonABS_Ordered = ErrorDistsNonABS_Ordered./Hits;
SAScores_Combined_Ordered(j,i) =
(nansum(ErrorDists_Ordered(j,i))+nansum(ErrorMiss_Ordered(j,i)))./(Hits(j,i)+Misses(j,i));
SAScoresABS_Combined_Ordered(j,i) =
(nansum(ErrorDistsABS_Ordered(j,i))+nansum(ErrorMiss_Ordered(j,i)))./(Hits(j,i)+Misses(j,i));
SAScoresNonABS_Combined_Ordered(j,i) =
(nansum(ErrorDistsNonABS_Ordered(j,i))+MaxDist*Misses(j,i))./(Hits(j,i)+Misses(j,i));
ErrorDists_Scrambled(j,i) = ErrorDists_Ordered(seq_i(j+2,i)-2,i);
SAScores_Scrambled(j,i) = SAScores_Ordered(seq_i(j+2,i)-2,i);
SAScoresABS_Scrambled(j,i) = SAScoresABS_Ordered(seq_i(j+2,i)-2,i);
SAScoresNonABS_Scrambled(j,i) = SAScoresNonABS_Ordered(seq_i(j+2,i)-2,i);
SAScores_Combined_Scrambled(j,i) = SAScores_Combined_Ordered(seq_i(j+2,i)-2,i);
SAScoresABS_Combined_Scrambled(j,i) = SAScoresABS_Combined_Ordered(seq_i(j+2,i)-2,i);
SAScoresNonABS_Combined_Scrambled(j,i) = SAScoresNonABS_Combined_Ordered(seq_i(j+2,i)-2,i);
*-----------------------------------------------------------------------------------------------------------------------------------------------
if mod(i, 2) == 0
  % i is even, Grp B
  if Duration_Ordered_GrpB(j+2,1) == 1
    count1_B = count1_B + 1;
      Hits_Duration_All(68+count1_B,1) = Hits(j,i);
      HitPerc_Duration_All(68+count1_B,1) = HitPerc_Ordered(j,i);
```

```
      MFAPerc_Duration_All(68+count1_B,1) = MFAPerc_Ordered(j,i);
      Misses_Duration_All(68+count1_B,1) = Misses(j,i);
      False_Duration_All(68+count1_B,1) = False(j,i);
      FA_M_Duration_All(68+count1_B,1) = FA_M(j,i);
      SAScores_Raw_Duration_All(68+count1_B,1) = SAScores_Raw_Ordered(j,i);
      SAScores_Raw_Combined_Duration_All(68+count1_B,1) = SAScores_Raw_Combined_Ordered(j,i);
      SAScores_Duration_All(68+count1_B,1) = SAScores_Ordered(j,i);
      SAScoresABS_Duration_All(68+count1_B,1) = SAScoresABS_Ordered(j,i);
      SAScoresNonABS_Duration_All(68+count1_B,1) = SAScoresNonABS_Ordered(j,i);
      SAScores_Combined_Duration_All(68+count1_B,1) = SAScores_Combined_Ordered(j,i);
      SAScoresABS_Combined_Duration_All(68+count1_B,1) = SAScoresABS_Combined_Ordered(j,i);
      SAScoresNonABS_Combined_Duration_All(68+count1_B,1) = SAScoresNonABS_Combined_Ordered(j,i);
      ErrorLaneDist_Duration_All(68+count1_B,1) = ErrorLaneDist(j,i);
      ErrorLaneDistPerc_Duration_All(68+count1_B,1) = ErrorLaneDistPerc(j,i);
      ErrorLaneNum_Duration_All(68+count1_B,1) = ErrorLaneNum(j,i);
      LaneErrorPerc_Duration_All(68+count1_B,1) = LaneErrorPerc(j,i);
      ErrorSpeed_Duration_All(68+count1_B,1) = ErrorSpeed(j,i);
      ErrorSpeedPerc_Duration_All(68+count1_B,1) = ErrorSpeedPerc(j,i);
      ErrorSpeedNum_Duration_All(68+count1_B,1) = ErrorSpeedNum(j,i);
      SpeedErrorPerc_Duration_All(68+count1_B,1) = SpeedErrorPerc(j,i);
    if j<=12
    count1_B_6 = count1_B_6 + 1;
      Hits_Duration_All_6cars(34+count1_B_6,1) = Hits(j,i);
      HitPerc_Duration_All_6cars(34+count1_B_6,1) = HitPerc_Ordered(j,i);
      MFAPerc_Duration_All_6cars(34+count1_B_6,1) = MFAPerc_Ordered(j,i);
      Misses_Duration_All_6cars(34+count1_B_6,1) = Misses(j,i);
      False_Duration_All_6cars(34+count1_B_6,1) = False(j,i);
      FA_M_Duration_All_6cars(34+count1_B_6,1) = FA_M(j,i);
      SAScores_Raw_Duration_All_6cars(34+count1_B_6,1) = SAScores_Raw_Ordered(j,i);
      SAScores_Raw_Combined_Duration_All_6cars(34+count1_B_6,1) = SAScores_Raw_Combined_Ordered(j,i);
      SAScores_Duration_All_6cars(34+count1_B_6,1) = SAScores_Ordered(j,i);
      SAScoresABS_Duration_All_6cars(34+count1_B_6,1) = SAScoresABS_Ordered(j,i);
      SAScoresNonABS_Duration_All_6cars(34+count1_B_6,1) = SAScoresNonABS_Ordered(j,i);
      SAScores_Combined_Duration_All_6cars(34+count1_B_6,1) = SAScores_Combined_Ordered(j,i);
      SAScoresABS_Combined_Duration_All_6cars(34+count1_B_6,1) = SAScoresABS_Combined_Ordered(j,i);
      SAScoresNonABS_Combined_Duration_All_6cars(34+count1_B_6,1) = SAScoresNonABS_Combined_Ordered(j,i);
      ErrorLaneDist_Duration_All_6cars(34+count1_B_6,1) = ErrorLaneDist(j,i);
      ErrorLaneDistPerc_Duration_All_6cars(34+count1_B_6,1) = ErrorLaneDistPerc(j,i);
      ErrorLaneNum_Duration_All_6cars(34+count1_B_6,1) = ErrorLaneNum(j,i);
      LaneErrorPerc_Duration_All_6cars(34+count1_B_6,1) = LaneErrorPerc(j,i);
      ErrorSpeed_Duration_All_6cars(34+count1_B_6,1) = ErrorSpeed(j,i);
      ErrorSpeedPerc_Duration_All_6cars(34+count1_B_6,1) = ErrorSpeedPerc(j,i);
      ErrorSpeedNum_Duration_All_6cars(34+count1_B_6,1) = ErrorSpeedNum(j,i);
      SpeedErrorPerc_Duration_All_6cars(34+count1_B_6,1) = SpeedErrorPerc(j,i);
    elseif j>12
    count1_B_4 = count1_B_4 + 1;
      Hits_Duration_All_4cars(34+count1_B_4,1) = Hits(j,i);
      HitPerc_Duration_All_4cars(34+count1_B_4,1) = HitPerc_Ordered(j,i);
      MFAPerc_Duration_All_4cars(34+count1_B_4,1) = MFAPerc_Ordered(j,i);
      Misses_Duration_All_4cars(34+count1_B_4,1) = Misses(j,i);
      False_Duration_All_4cars(34+count1_B_4,1) = False(j,i);
      FA_M_Duration_All_4cars(34+count1_B_4,1) = FA_M(j,i);
      SAScores_Raw_Duration_All_4cars(34+count1_B_4,1) = SAScores_Raw_Ordered(j,i);
      SAScores_Raw_Combined_Duration_All_4cars(34+count1_B_4,1) = SAScores_Raw_Combined_Ordered(j,i);
      SAScores_Duration_All_4cars(34+count1_B_4,1) = SAScores_Ordered(j,i);
      SAScoresABS_Duration_All_4cars(34+count1_B_4,1) = SAScoresABS_Ordered(j,i);
      SAScoresNonABS_Duration_All_4cars(34+count1_B_4,1) = SAScoresNonABS_Ordered(j,i);
      SAScores_Combined_Duration_All_4cars(34+count1_B_4,1) = SAScores_Combined_Ordered(j,i);
      SAScoresABS_Combined_Duration_All_4cars(34+count1_B_4,1) = SAScoresABS_Combined_Ordered(j,i);
      SAScoresNonABS_Combined_Duration_All_4cars(34+count1_B_4,1) = SAScoresNonABS_Combined_Ordered(j,i);
      ErrorLaneDist_Duration_All_4cars(34+count1_B_4,1) = ErrorLaneDist(j,i);
      ErrorLaneDistPerc_Duration_All_4cars(34+count1_B_4,1) = ErrorLaneDistPerc(j,i);
      ErrorLaneNum_Duration_All_4cars(34+count1_B_4,1) = ErrorLaneNum(j,i);
      LaneErrorPerc_Duration_All_4cars(34+count1_B_4,1) = LaneErrorPerc(j,i);
      ErrorSpeed_Duration_All_4cars(34+count1_B_4,1) = ErrorSpeed(j,i);
      ErrorSpeedPerc_Duration_All_4cars(34+count1_B_4,1) = ErrorSpeedPerc(j,i);
      ErrorSpeedNum_Duration_All_4cars(34+count1_B_4,1) = ErrorSpeedNum(j,i);
      SpeedErrorPerc_Duration_All_4cars(34+count1_B_4,1) = SpeedErrorPerc(j,i);
    end
*
```

```
save('AdvancedMeasures.mat')
```

* The code between the red lines is repeated for each duration (i.e., video length), and again for group A.

Appendix A6-3: Grouping data (SplittingData.m)

This code grouped the data as preparation for calculating scores from it.

```
% Made by Xander Coster
% 1268546
% Splitting data into separate chunks for all 34 participants

load('AllData_Scrambled.mat')
load('AllData_Ordered.mat')
load('sequences.mat') %seq_i (26x34)
load('AnswersBig.mat')

% Choose threshold for close by/far away
FrontDist = 60;
RearDist = -40;

%% Make empty separate datafiles
    % pre-test questionnaire
Ages_All = nan(1,34); Ages_GrpA = nan(1,17); Ages_GrpB = nan(1,17);                    %Checked OK
Gender_All = nan(1,34); Gender_GrpA = nan(1,17); Gender_GrpB = nan(1,17);              %Checked OK
DrLiAge_All = nan(1,34); DrLiAge_GrpA = nan(1,17); DrLiAge_GrpB = nan(1,17);           %Checked OK
VehUseFreq_All = nan(1,34); VehUseFreq_GrpA = nan(1,17); VehUseFreq_GrpB = nan(1,17);%Checked OK
VehKm_All = nan(1,34); VehKm_GrpA = nan(1,17); VehKm_GrpB = nan(1,17);                 %Checked OK
GoogleCar_All = nan(1,34);                                                             %Checked OK

    % Subjective measures (both ordered and scrambled)
Diff_Ordered_All = nan(26,34); Diff_Ordered_GrpA = nan(26,17); Diff_Ordered_GrpB = nan(26,17);
%Checked OK
Diff_Scrambled_All = nan(26,34); Diff_Scrambled_GrpA = nan(26,17); Diff_Scrambled_GrpB = nan(26,17);
%Checked OK
Diff_Duration_All = nan(136,6); Diff_Duration_GrpA = nan(68,6);Diff_Duration_GrpB = nan(68,6);
Diff_Duration_All_4cars = nan(68,6); Diff_Duration_GrpA_4cars = nan(34,6); Diff_Duration_GrpB_4cars =
nan(34,6);
Diff_Duration_All_6cars = nan(68,6); Diff_Duration_GrpA_6cars = nan(34,6); Diff_Duration_GrpB_6cars =
nan(34,6);
Time_Ordered_All = nan(26,34); Time_Ordered_GrpA = nan(26,17); Time_Ordered_GrpB = nan(26,17);
%Checked OK
Time_Scrambled_All = nan(26,34); Time_Scrambled_GrpA = nan(26,17); Time_Scrambled_GrpB = nan(26,17);
%Checked OK
Time_Duration_All = nan(136,6); Time_Duration_GrpA = nan(68,6); Time_Duration_GrpB = nan(68,6);
Time_Duration_All_4cars = nan(68,6); Time_Duration_GrpA_4cars = nan(34,6); Time_Duration_GrpB_4cars =
nan(34,6);
Time_Duration_All_6cars = nan(68,6); Time_Duration_GrpA_6cars = nan(34,6); Time_Duration_GrpB_6cars =
nan(34,6);

    % Basic and/or advanced measures (both ordered and scrambled) per trial
        %Total placed cars (= placed - deleted)
NumP_Ordered_All = nan(26,34); NumP_Ordered_GrpA = nan(26,17); NumP_Ordered_GrpB = nan(26,17);
%Checked OK
NumP_Scrambled_All = nan(26,34); NumP_Scrambled_GrpA = nan(26,17); NumP_Scrambled_GrpB = nan(26,17);
%Checked OK
AllNumP_Ordered_All = nan(26,34);
NumP_Duration_All = nan(136,6); NumP_Duration_GrpA = nan(68,6); NumP_Duration_GrpB = nan(68,6);
NumP_Duration_All_4cars = nan(68,6); NumP_Duration_GrpA_4cars = nan(34,6); NumP_Duration_GrpB_4cars =
nan(34,6);
NumP_Duration_All_6cars = nan(68,6); NumP_Duration_GrpA_6cars = nan(34,6); NumP_Duration_GrpB_6cars =
nan(34,6);
        % for 8 possible cars
DistP_Ordered_All = nan(26,272); DistP_Ordered_GrpA = nan(26,136); DistP_Ordered_GrpB = nan(26,136);
%Checked OK
DistP_Scrambled_All = nan(26,272); DistP_Scrambled_GrpA = nan(26,136); DistP_Scrambled_GrpB = nan(26,136);
%Checked OK
LaneP_Ordered_All = nan(26,272); LaneP_Ordered_GrpA = nan(26,136); LaneP_Ordered_GrpB = nan(26,136);
%Checked OK
LaneP_Scrambled_All = nan(26,272); LaneP_Scrambled_GrpA = nan(26,136); LaneP_Scrambled_GrpB = nan(26,136);
%Checked OK
LaneNumP_Ordered_All = nan(26,102);
LaneNumP_Ordered_All_Left = nan(26,34); LaneNumP_Ordered_All_Middle = nan(26,34); LaneNumP_Ordered_All_Right
= nan(26,34);      %Checked OK
LaneNumP_Ordered_GrpA = nan(26,51);
LaneNumP_Ordered_GrpA_Left = nan(26,17); LaneNumP_Ordered_GrpA_Middle = nan(26,17);
LaneNumP_Ordered_GrpA_Right = nan(26,17);     %Checked OK
LaneNumP_Ordered_GrpB = nan(26,51);
LaneNumP_Ordered_GrpB_Left = nan(26,17); LaneNumP_Ordered_GrpB_Middle = nan(26,17);
LaneNumP_Ordered_GrpB_Right = nan(26,17);     %Checked OK
LaneNumP_Scrambled_All = nan(26,102);
LaneNumP_Scrambled_All_Left = nan(26,34); LaneNumP_Scrambled_All_Middle = nan(26,34);
LaneNumP_Scrambled_All_Right = nan(26,34);    %Checked OK
LaneNumP_Scrambled_GrpA = nan(26,51);
LaneNumP_Scrambled_GrpA_Left = nan(26,17); LaneNumP_Scrambled_GrpA_Middle = nan(26,17);
LaneNumP_Scrambled_GrpA_Right = nan(26,17);   %Checked OK
LaneNumP_Scrambled_GrpB = nan(26,51);
LaneNumP_Scrambled_GrpB_Left = nan(26,17); LaneNumP_Scrambled_GrpB_Middle = nan(26,17);
LaneNumP_Scrambled_GrpB_Right = nan(26,17);   %Checked OK
```

```
LaneNumP_Duration_All_Left = nan(136,6); LaneNumP_Duration_GrpA_Left = nan(68,6); LaneNumP_Duration_GrpB_Left
= nan(68,6);
LaneNumP_Duration_All_Left_4cars = nan(68,6); LaneNumP_Duration_GrpA_Left_4cars = nan(34,6);
LaneNumP_Duration_GrpB_Left_4cars = nan(34,6);
LaneNumP_Duration_All_Left_6cars = nan(68,6); LaneNumP_Duration_GrpA_Left_6cars = nan(34,6);
LaneNumP_Duration_GrpB_Left_6cars = nan(34,6);
LaneNumP_Duration_All_Middle = nan(136,6); LaneNumP_Duration_GrpA_Middle = nan(68,6);
LaneNumP_Duration_GrpB_Middle = nan(68,6);
LaneNumP_Duration_All_Middle_4cars = nan(68,6); LaneNumP_Duration_GrpA_Middle_4cars = nan(34,6);
LaneNumP_Duration_GrpB_Middle_4cars = nan(34,6);
LaneNumP_Duration_All_Middle_6cars = nan(68,6); LaneNumP_Duration_GrpA_Middle_6cars = nan(34,6);
LaneNumP_Duration_GrpB_Middle_6cars = nan(34,6);
LaneNumP_Duration_All_Right = nan(136,6); LaneNumP_Duration_GrpA_Right = nan(68,6);
LaneNumP_Duration_GrpB_Right = nan(68,6);
LaneNumP_Duration_All_Right_4cars = nan(68,6); LaneNumP_Duration_GrpA_Right_4cars = nan(34,6);
LaneNumP_Duration_GrpB_Right_4cars = nan(34,6);
LaneNumP_Duration_All_Right_6cars = nan(68,6); LaneNumP_Duration_GrpA_Right_6cars = nan(34,6);
LaneNumP_Duration_GrpB_Right_6cars = nan(34,6);

FrontNumP_Ordered_All = nan(26,34); FrontNumP_Ordered_GrpA = nan(26,17); FrontNumP_Ordered_GrpB = nan(26,17);
%Checked OK
FrontNumP_Scrambled_All = nan(26,34); FrontNumP_Scrambled_GrpA = nan(26,17); FrontNumP_Scrambled_GrpB =
nan(26,17);        %Checked OK
RearNumP_Ordered_All = nan(26,34); RearNumP_Ordered_GrpA = nan(26,17); RearNumP_Ordered_GrpB = nan(26,17);
%Checked OK
RearNumP_Scrambled_All = nan(26,34); RearNumP_Scrambled_GrpA = nan(26,17); RearNumP_Scrambled_GrpB =
nan(26,17);         %Checked OK
FrontNumP_Duration_All = nan(136,6); FrontNumP_Duration_GrpA = nan(68,6); FrontNumP_Duration_GrpB =
nan(68,6);
FrontNumP_Duration_All_4cars = nan(68,6); FrontNumP_Duration_GrpA_4cars = nan(34,6);
FrontNumP_Duration_GrpB_4cars = nan(34,6);
FrontNumP_Duration_All_6cars = nan(68,6); FrontNumP_Duration_GrpA_6cars = nan(34,6);
FrontNumP_Duration_GrpB_6cars = nan(34,6);
RearNumP_Duration_All = nan(136,6); RearNumP_Duration_GrpA = nan(68,6); RearNumP_Duration_GrpB = nan(68,6);
RearNumP_Duration_All_4cars = nan(68,6); RearNumP_Duration_GrpA_4cars = nan(34,6);
RearNumP_Duration_GrpB_4cars = nan(34,6);
RearNumP_Duration_All_6cars = nan(68,6); RearNumP_Duration_GrpA_6cars = nan(34,6);
RearNumP_Duration_GrpB_6cars = nan(34,6);
FrontCloseNumP_Ordered_All = nan(26,34); FrontCloseNumP_Ordered_GrpA = nan(26,17);
FrontCloseNumP_Ordered_GrpB = nan(26,17);     %Checked OK
FrontCloseNumP_Scrambled_All = nan(26,34); FrontCloseNumP_Scrambled_GrpA = nan(26,17);
FrontCloseNumP_Scrambled_GrpB = nan(26,17);   %Checked OK
FrontFarNumP_Ordered_All = nan(26,34); FrontFarNumP_Ordered_GrpA = nan(26,17); FrontFarNumP_Ordered_GrpB =
nan(26,17);       %Checked OK
FrontFarNumP_Scrambled_All = nan(26,34); FrontFarNumP_Scrambled_GrpA = nan(26,17);
FrontFarNumP_Scrambled_GrpB = nan(26,17);     %Checked OK
FrontCloseNumP_Duration_All = nan(136,6); FrontCloseNumP_Duration_GrpA = nan(68,6);
FrontCloseNumP_Duration_GrpB = nan(68,6);
FrontCloseNumP_Duration_All_4cars = nan(68,6); FrontCloseNumP_Duration_GrpA_4cars = nan(34,6);
FrontCloseNumP_Duration_GrpB_4cars = nan(34,6);
FrontCloseNumP_Duration_All_6cars = nan(68,6); FrontCloseNumP_Duration_GrpA_6cars = nan(34,6);
FrontCloseNumP_Duration_GrpB_6cars = nan(34,6);
FrontFarNumP_Duration_All = nan(136,6); FrontFarNumP_Duration_GrpA = nan(68,6); FrontFarNumP_Duration_GrpB =
nan(68,6);
FrontFarNumP_Duration_All_4cars = nan(68,6); FrontFarNumP_Duration_GrpA_4cars = nan(34,6);
FrontFarNumP_Duration_GrpB_4cars = nan(34,6);
FrontFarNumP_Duration_All_6cars = nan(68,6); FrontFarNumP_Duration_GrpA_6cars = nan(34,6);
FrontFarNumP_Duration_GrpB_6cars = nan(34,6);
RearCloseNumP_Ordered_All = nan(26,34); RearCloseNumP_Ordered_GrpA = nan(26,17); RearCloseNumP_Ordered_GrpB =
nan(26,17);       %Checked OK
RearCloseNumP_Scrambled_All = nan(26,34); RearCloseNumP_Scrambled_GrpA = nan(26,17);
RearCloseNumP_Scrambled_GrpB = nan(26,17);    %Checked OK
RearFarNumP_Ordered_All = nan(26,34); RearFarNumP_Ordered_GrpA = nan(26,17); RearFarNumP_Ordered_GrpB =
nan(26,17);         %Checked OK
RearFarNumP_Scrambled_All = nan(26,34); RearFarNumP_Scrambled_GrpA = nan(26,17); RearFarNumP_Scrambled_GrpB =
nan(26,17);       %Checked OK
RearCloseNumP_Duration_All = nan(136,6); RearCloseNumP_Duration_GrpA = nan(68,6); RearCloseNumP_Duration_GrpB
= nan(68,6);
RearCloseNumP_Duration_All_4cars = nan(68,6); RearCloseNumP_Duration_GrpA_4cars = nan(34,6);
RearCloseNumP_Duration_GrpB_4cars = nan(34,6);
RearCloseNumP_Duration_All_6cars = nan(68,6); RearCloseNumP_Duration_GrpA_6cars = nan(34,6);
RearCloseNumP_Duration_GrpB_6cars = nan(34,6);
RearFarNumP_Duration_All = nan(136,6); RearFarNumP_Duration_GrpA = nan(68,6); RearFarNumP_Duration_GrpB =
nan(68,6);
RearFarNumP_Duration_All_4cars = nan(68,6); RearFarNumP_Duration_GrpA_4cars = nan(34,6);
RearFarNumP_Duration_GrpB_4cars = nan(34,6);
RearFarNumP_Duration_All_6cars = nan(68,6); RearFarNumP_Duration_GrpA_6cars = nan(34,6);
RearFarNumP_Duration_GrpB_6cars = nan(34,6);
SpeedP_Ordered_All = nan(26,272); SpeedP_Ordered_GrpA = nan(26,136); SpeedP_Ordered_GrpB = nan(26,136);
%Checked OK
SpeedP_Scrambled_All = nan(26,272); SpeedP_Scrambled_GrpA = nan(26,136); SpeedP_Scrambled_GrpB = nan(26,136);
%Checked OK
```

```
    % From Answers
Duration_Ordered_GrpA = AnswersBig2(:,2);       %Checked OK
Duration_Ordered_GrpB = AnswersBig2(:,3);       %Checked OK
Duration_Scrambled_All = zeros(26,34); Duration_Scrambled_GrpA = zeros(26,17); Duration_Scrambled_GrpB =
zeros(26,17);         %Checked OK
NumA_Ordered = AnswersBig2(:,1);                %Checked OK
NumA_Scrambled_All = zeros(26,34); NumA_Scrambled_GrpA = zeros(26,17); NumA_Scrambled_GrpB = zeros(26,17);
%Checked OK
NumA_Duration_All = zeros(136,6); NumA_Duration_GrpA = zeros(68,6); NumA_Duration_GrpB = zeros(68,6);
NumA_Duration_All_4cars = zeros(68,6); NumA_Duration_GrpA_4cars = zeros(34,6); NumA_Duration_GrpB_4cars =
zeros(34,6);
NumA_Duration_All_6cars = zeros(68,6); NumA_Duration_GrpA_6cars = zeros(34,6); NumA_Duration_GrpB_6cars =
zeros(34,6);
DistA_Ordered = zeros(26,8);     %x8 to make it same size as P %Checked OK
DistA_Scrambled_All = zeros(26,272); DistA_Scrambled_GrpA = zeros(26,136); DistA_Scrambled_GrpB =
zeros(26,136);           %Checked OK
LaneA_Ordered = zeros(26,8);                    %Checked OK
LaneA_Scrambled_All = zeros(26,272); LaneA_Scrambled_GrpA = zeros(26,136); LaneA_Scrambled_GrpB =
zeros(26,136);           %Checked OK
LaneNumA_Ordered = [1 1 1; 1 1 1;...            %Checked OK
3 1 2; 2 1 3; 1 2 3; 3 2 1; 3 1 2; 2 2 2; 2 1 3; 3 2 1; 1 2 3; 2 2 2; 3 1 2; 3 2 1;...
1 2 1; 1 1 2; 2 1 1; 2 0 2; 0 2 2; 2 2 0; 1 2 1; 1 1 2; 2 1 1; 2 0 2; 0 2 2; 2 2 0];
LaneNumA_Ordered_Left = LaneNumA_Ordered(:,1); LaneNumA_Ordered_Middle = LaneNumA_Ordered(:,2);
LaneNumA_Ordered_Right = LaneNumA_Ordered(:,3); %Checked OK
LaneNumA_Duration_All_Left = zeros(136,6); LaneNumA_Duration_GrpA_Left = zeros(68,6);
LaneNumA_Duration_GrpB_Left = zeros(68,6);
LaneNumA_Duration_All_Left_4cars = zeros(68,6); LaneNumA_Duration_GrpA_Left_4cars = zeros(34,6);
LaneNumA_Duration_GrpB_Left_4cars = zeros(34,6);
LaneNumA_Duration_All_Left_6cars = zeros(68,6); LaneNumA_Duration_GrpA_Left_6cars = zeros(34,6);
LaneNumA_Duration_GrpB_Left_6cars = zeros(34,6);
LaneNumA_Duration_All_Middle = zeros(136,6); LaneNumA_Duration_GrpA_Middle = zeros(68,6);
LaneNumA_Duration_GrpB_Middle = zeros(68,6);
LaneNumA_Duration_All_Middle_4cars = zeros(68,6); LaneNumA_Duration_GrpA_Middle_4cars = zeros(34,6);
LaneNumA_Duration_GrpB_Middle_4cars = zeros(34,6);
LaneNumA_Duration_All_Middle_6cars = zeros(68,6); LaneNumA_Duration_GrpA_Middle_6cars = zeros(34,6);
LaneNumA_Duration_GrpB_Middle_6cars = zeros(34,6);
LaneNumA_Duration_All_Right = zeros(136,6); LaneNumA_Duration_GrpA_Right = zeros(68,6);
LaneNumA_Duration_GrpB_Right = zeros(68,6);
LaneNumA_Duration_All_Right_4cars = zeros(68,6); LaneNumA_Duration_GrpA_Right_4cars = zeros(34,6);
LaneNumA_Duration_GrpB_Right_4cars = zeros(34,6);
LaneNumA_Duration_All_Right_6cars = zeros(68,6); LaneNumA_Duration_GrpA_Right_6cars = zeros(34,6);
LaneNumA_Duration_GrpB_Right_6cars = zeros(34,6);
LaneNumA_Scrambled_All = zeros(26,102);
LaneNumA_Scrambled_All_Left = zeros(26,34); LaneNumA_Scrambled_All_Middle = zeros(26,34);
LaneNumA_Scrambled_All_Right = zeros(26,34);    %Checked OK
LaneNumA_Scrambled_GrpA = zeros(26,51);
LaneNumA_Scrambled_GrpA_Left = zeros(26,17); LaneNumA_Scrambled_GrpA_Middle = zeros(26,17);
LaneNumA_Scrambled_GrpA_Right = zeros(26,17);   %Checked OK
LaneNumA_Scrambled_GrpB = zeros(26,51);
LaneNumA_Scrambled_GrpB_Left = zeros(26,17); LaneNumA_Scrambled_GrpB_Middle = zeros(26,17);
LaneNumA_Scrambled_GrpB_Right = zeros(26,17);   %Checked OK
FrontNumA_Ordered = zeros(26,34);               %Checked OK
FrontNumA_Scrambled_All = zeros(26,34); FrontNumA_Scrambled_GrpA = zeros(26,17); FrontNumA_Scrambled_GrpB =
zeros(26,17);        %Checked OK
RearNumA_Ordered = zeros(26,34);                %Checked OK
RearNumA_Scrambled_All = zeros(26,34); RearNumA_Scrambled_GrpA = zeros(26,17); RearNumA_Scrambled_GrpB =
zeros(26,17);         %Checked OK
FrontNumA_Duration_All = zeros(136,6); FrontNumA_Duration_GrpA = zeros(68,6); FrontNumA_Duration_GrpB =
zeros(68,6);
FrontNumA_Duration_All_4cars = zeros(68,6); FrontNumA_Duration_GrpA_4cars = zeros(34,6);
FrontNumA_Duration_GrpB_4cars = zeros(34,6);
FrontNumA_Duration_All_6cars = zeros(68,6); FrontNumA_Duration_GrpA_6cars = zeros(34,6);
FrontNumA_Duration_GrpB_6cars = zeros(34,6);
RearNumA_Duration_All = zeros(136,6); RearNumA_Duration_GrpA = zeros(68,6); RearNumA_Duration_GrpB =
zeros(68,6);
RearNumA_Duration_All_4cars = zeros(68,6); RearNumA_Duration_GrpA_4cars = zeros(34,6);
RearNumA_Duration_GrpB_4cars = zeros(34,6);
RearNumA_Duration_All_6cars = zeros(68,6); RearNumA_Duration_GrpA_6cars = zeros(34,6);
RearNumA_Duration_GrpB_6cars = zeros(34,6);
FrontCloseNumA_Ordered = zeros(26,34);          %Checked OK
FrontCloseNumA_Scrambled_All = zeros(26,34); FrontCloseNumA_Scrambled_GrpA = zeros(26,17);
FrontCloseNumA_Scrambled_GrpB = zeros(26,17);   %Checked OK
FrontFarNumA_Ordered = zeros(26,34);            %Checked OK
FrontFarNumA_Scrambled_All = zeros(26,34); FrontFarNumA_Scrambled_GrpA = zeros(26,17);
FrontFarNumA_Scrambled_GrpB = zeros(26,17);     %Checked OK
FrontCloseNumA_Duration_All = zeros(136,6); FrontCloseNumA_Duration_GrpA = zeros(68,6);
FrontCloseNumA_Duration_GrpB = zeros(68,6);
FrontCloseNumA_Duration_All_4cars = zeros(68,6); FrontCloseNumA_Duration_GrpA_4cars = zeros(34,6);
FrontCloseNumA_Duration_GrpB_4cars = zeros(34,6);
FrontCloseNumA_Duration_All_6cars = zeros(68,6); FrontCloseNumA_Duration_GrpA_6cars = zeros(34,6);
FrontCloseNumA_Duration_GrpB_6cars = zeros(34,6);
```

```
FrontFarNumA_Duration_All = zeros(136,6); FrontFarNumA_Duration_GrpA = zeros(68,6);
FrontFarNumA_Duration_GrpB = zeros(68,6);
FrontFarNumA_Duration_All_4cars = zeros(68,6); FrontFarNumA_Duration_GrpA_4cars = zeros(34,6);
FrontFarNumA_Duration_GrpB_4cars = zeros(34,6);
FrontFarNumA_Duration_All_6cars = zeros(68,6); FrontFarNumA_Duration_GrpA_6cars = zeros(34,6);
FrontFarNumA_Duration_GrpB_6cars = zeros(34,6);
RearCloseNumA_Ordered = zeros(26,34);           %Checked OK
RearCloseNumA_Scrambled_All = zeros(26,34); RearCloseNumA_Scrambled_GrpA = zeros(26,17);
RearCloseNumA_Scrambled_GrpB = zeros(26,17);    %Checked OK
RearFarNumA_Ordered = zeros(26,34);             %Checked OK
RearFarNumA_Scrambled_All = zeros(26,34); RearFarNumA_Scrambled_GrpA = zeros(26,17);
RearFarNumA_Scrambled_GrpB = zeros(26,17);      %Checked OK
RearCloseNumA_Duration_All = zeros(136,6); RearCloseNumA_Duration_GrpA = zeros(68,6);
RearCloseNumA_Duration_GrpB = zeros(68,6);
RearCloseNumA_Duration_All_4cars = zeros(68,6); RearCloseNumA_Duration_GrpA_4cars = zeros(34,6);
RearCloseNumA_Duration_GrpB_4cars = zeros(34,6);
RearCloseNumA_Duration_All_6cars = zeros(68,6); RearCloseNumA_Duration_GrpA_6cars = zeros(34,6);
RearCloseNumA_Duration_GrpB_6cars = zeros(34,6);
RearFarNumA_Duration_All = zeros(136,6); RearFarNumA_Duration_GrpA = zeros(68,6); RearFarNumA_Duration_GrpB =
zeros(68,6);
RearFarNumA_Duration_All_4cars = zeros(68,6); RearFarNumA_Duration_GrpA_4cars = zeros(34,6);
RearFarNumA_Duration_GrpB_4cars = zeros(34,6);
RearFarNumA_Duration_All_6cars = zeros(68,6); RearFarNumA_Duration_GrpA_6cars = zeros(34,6);
RearFarNumA_Duration_GrpB_6cars = zeros(34,6);

SpeedA_Ordered = zeros(26,8);                   %Checked OK
SpeedA_Scrambled_All = zeros(26,272); SpeedA_Scrambled_GrpA = zeros(26,136); SpeedA_Scrambled_GrpB =
zeros(26,136);          %Checked OK

%% Fill separate datafiles
for i = 1:34
    Data_i_Ordered = cell2mat(Data_Ordered(1,i));
    Data_i_Scrambled = cell2mat(Data_Scrambled(1,i));

 % single values per participant
    %All participants
    Ages_All(1,i) = Data_i_Ordered(2,1);
    Gender_All(1,i) = Data_i_Ordered(5,1);
    DrLiAge_All(1,i) = Data_i_Ordered(6,1);
    VehUseFreq_All(1,i) = Data_i_Ordered(7,1);
    VehKm_All(1,i) = Data_i_Ordered(8,1);
    GoogleCar_All(1,i) = Data_i_Ordered(9,1);

    %groups
    if mod(i, 2) == 0
  % i is even, Grp B
    Ages_GrpB(i/2) = Data_i_Ordered(2,1);
    Gender_GrpB(i/2) = Data_i_Ordered(5,1);
    DrLiAge_GrpB(i/2) = Data_i_Ordered(6,1);
    VehUseFreq_GrpB(i/2) = Data_i_Ordered(7,1);
    VehKm_GrpB(i/2) = Data_i_Ordered(8,1);
    else
  % i is odd, Grp A
    Ages_GrpA((i+1)/2) = Data_i_Ordered(2,1);
    Gender_GrpA((i+1)/2) = Data_i_Ordered(5,1);
    DrLiAge_GrpA((i+1)/2) = Data_i_Ordered(6,1);
    VehUseFreq_GrpA((i+1)/2) = Data_i_Ordered(7,1);
    VehKm_GrpA((i+1)/2) = Data_i_Ordered(8,1);
    end

% values per trial per participant
    for j = 1:26
        %All participants
        Diff_Ordered_All(j,i) = Data_i_Ordered(j,30);
        Time_Ordered_All(j,i) = Data_i_Ordered(j,32);
        NumP_Ordered_All(j,i) = Data_i_Ordered(j,5);    %#P - Del
        AllNumP_Ordered_All(j,i) = Data_i_Ordered(j,4);         %#P
        DelNum_Orderered_All = AllNumP_Ordered_All - NumP_Ordered_All;
        Diff_Scrambled_All(j,i) = Data_i_Scrambled(j,30);
        Time_Scrambled_All(j,i) = Data_i_Scrambled(j,32);
        NumP_Scrambled_All(j,i) = Data_i_Scrambled(j,5);
        NumA_Scrambled_All(j,i) = AnswersBig2(seq_i(j,i),1);

        %groups
    if mod(i, 2) == 0
  % i is even, Grp B
        Diff_Ordered_GrpB(j,i/2) = Data_i_Ordered(j,30);
        Time_Ordered_GrpB(j,i/2) = Data_i_Ordered(j,32);
        NumP_Ordered_GrpB(j,i/2) = Data_i_Ordered(j,5);
        Diff_Scrambled_GrpB(j,i/2) = Data_i_Scrambled(j,30);
        Time_Scrambled_GrpB(j,i/2) = Data_i_Scrambled(j,32);
        NumP_Scrambled_GrpB(j,i/2) = Data_i_Scrambled(j,5);
```

```
        Duration_Scrambled_All(j,i) = AnswersBig2(seq_i(j,i),3);
        Duration_Scrambled_GrpB(j,i/2) = AnswersBig2(seq_i(j,i),3);
        NumA_Scrambled_GrpB(j,i/2) = AnswersBig2(seq_i(j,i),1);
    else
  % i is odd, Grp A
        Diff_Ordered_GrpA(j,(i+1)/2) = Data_i_Ordered(j,30);
        Time_Ordered_GrpA(j,(i+1)/2) = Data_i_Ordered(j,32);
        NumP_Ordered_GrpA(j,(i+1)/2) = Data_i_Ordered(j,5);
        Diff_Scrambled_GrpA(j,(i+1)/2) = Data_i_Scrambled(j,30);
        Time_Scrambled_GrpA(j,(i+1)/2) = Data_i_Scrambled(j,32);
        NumP_Scrambled_GrpA(j,(i+1)/2) = Data_i_Scrambled(j,5);
        Duration_Scrambled_All(j,i) = AnswersBig2(seq_i(j,i),2);
        Duration_Scrambled_GrpA(j,(i+1)/2) = AnswersBig2(seq_i(j,i),2);
        NumA_Scrambled_GrpA(j,(i+1)/2) = AnswersBig2(seq_i(j,i),1);
    end

% values for maximally 8 cars
    for k = 1:8
        %All participants
        DistP_Ordered_All(j,(i-1)*8+k) = Data_i_Ordered(j,k*3+4)*10;
        LaneP_Ordered_All(j,(i-1)*8+k) = Data_i_Ordered(j,k*3+3);
        SpeedP_Ordered_All(j,(i-1)*8+k) = Data_i_Ordered(j,k*3+5);
        DistP_Scrambled_All(j,(i-1)*8+k) = Data_i_Scrambled(j,k*3+4)*10;
        LaneP_Scrambled_All(j,(i-1)*8+k) = Data_i_Scrambled(j,k*3+3);
        SpeedP_Scrambled_All(j,(i-1)*8+k) = Data_i_Scrambled(j,k*3+5);

        %groups
    if mod(i, 2) == 0
  % i is even, Grp B
        DistP_Ordered_GrpB(j,(i/2-1)*8+k) = Data_i_Ordered(j,k*3+4)*10;
        LaneP_Ordered_GrpB(j,(i/2-1)*8+k) = Data_i_Ordered(j,k*3+3);
        SpeedP_Ordered_GrpB(j,(i/2-1)*8+k) = Data_i_Ordered(j,k*3+5);
        DistP_Scrambled_GrpB(j,(i/2-1)*8+k) = Data_i_Scrambled(j,k*3+4)*10;
        LaneP_Scrambled_GrpB(j,(i/2-1)*8+k) = Data_i_Scrambled(j,k*3+3);
        SpeedP_Scrambled_GrpB(j,(i/2-1)*8+k) = Data_i_Scrambled(j,k*3+5);

    else
  % i is odd, Grp A
        DistP_Ordered_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Ordered(j,k*3+4)*10;
        LaneP_Ordered_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Ordered(j,k*3+3);
        SpeedP_Ordered_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Ordered(j,k*3+5);
        DistP_Scrambled_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Scrambled(j,k*3+4)*10;
        LaneP_Scrambled_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Scrambled(j,k*3+3);
        SpeedP_Scrambled_GrpA(j,((i+1)/2-1)*8+k) = Data_i_Scrambled(j,k*3+5);
    end
    end
    %%  NumP_Ordered_All
    if isnan(NumP_Ordered_All(j,i)) == 0; %non-deleted data
      LaneNumP_Ordered_All(j,(i-1)*3+1:(i-1)*3+3) = 0;
      FrontNumP_Ordered_All(j,i) = 0;
      FrontFarNumP_Ordered_All(j,i) = 0;
      FrontCloseNumP_Ordered_All(j,i) = 0;
      RearNumP_Ordered_All(j,i) = 0;
      RearFarNumP_Ordered_All(j,i) = 0;
      RearCloseNumP_Ordered_All(j,i) = 0;
      if mod(i, 2) == 0
  % i is even, Grp B
      LaneNumP_Ordered_GrpB(j,(i/2-1)*3+1:(i/2-1)*3+3) = 0;
      FrontNumP_Ordered_GrpB(j,i/2) = 0;
      FrontFarNumP_Ordered_GrpB(j,i/2) = 0;
      FrontCloseNumP_Ordered_GrpB(j,i/2) = 0;
      RearNumP_Ordered_GrpB(j,i/2) = 0;
      RearFarNumP_Ordered_GrpB(j,i/2) = 0;
      RearCloseNumP_Ordered_GrpB(j,i/2) = 0;
      else
  % i is odd, Grp A
      LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+1:((i+1)/2-1)*3+3) = 0;
      FrontNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      FrontFarNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      FrontCloseNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      RearNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      RearFarNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      RearCloseNumP_Ordered_GrpA(j,(i+1)/2) = 0;
      end

    for m = 1:NumP_Ordered_All(j,i)
        Lane_Ordered = LaneP_Ordered_All(j,(i-1)*8+m);
        Dist_Ordered = DistP_Ordered_All(j,(i-1)*8+m);
    if mod(i, 2) == 0
  % i is even, Grp B
        if roundn(Lane_Ordered,-2) == -0.35
            LaneNumP_Ordered_All(j,(i-1)*3+1) = LaneNumP_Ordered_All(j,(i-1)*3+1) + 1;
```

```
            LaneNumP_Ordered_GrpB(j,(i/2-1)*3+1) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+1) + 1;
        elseif roundn(Lane_Ordered,-2) == 0.00
            LaneNumP_Ordered_All(j,(i-1)*3+2) = LaneNumP_Ordered_All(j,(i-1)*3+2) + 1;
            LaneNumP_Ordered_GrpB(j,(i/2-1)*3+2) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+2) + 1;
        elseif roundn(Lane_Ordered,-2) == 0.35
            LaneNumP_Ordered_All(j,(i-1)*3+3) = LaneNumP_Ordered_All(j,(i-1)*3+3) + 1;
            LaneNumP_Ordered_GrpB(j,(i/2-1)*3+3) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+3) + 1;
        end
    LaneNumP_Ordered_GrpB_Left(j,i/2) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+1);
    LaneNumP_Ordered_GrpB_Middle(j,i/2) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+2);
    LaneNumP_Ordered_GrpB_Right(j,i/2) = LaneNumP_Ordered_GrpB(j,(i/2-1)*3+3);

        if Dist_Ordered >= 0
            FrontNumP_Ordered_All(j,i) = FrontNumP_Ordered_All(j,i) + 1;
            FrontNumP_Ordered_GrpB(j,i/2) = FrontNumP_Ordered_GrpB(j,i/2) + 1;
            if Dist_Ordered >= FrontDist
                FrontFarNumP_Ordered_All(j,i) = FrontFarNumP_Ordered_All(j,i) + 1;
                FrontFarNumP_Ordered_GrpB(j,i/2) = FrontFarNumP_Ordered_GrpB(j,i/2) + 1;
            else
                FrontCloseNumP_Ordered_All(j,i) = FrontCloseNumP_Ordered_All(j,i) + 1;
                FrontCloseNumP_Ordered_GrpB(j,i/2) = FrontCloseNumP_Ordered_GrpB(j,i/2) + 1;
            end
        end
        if Dist_Ordered <= 0
           RearNumP_Ordered_All(j,i) =  RearNumP_Ordered_All(j,i) + 1;
           RearNumP_Ordered_GrpB(j,i/2) = RearNumP_Ordered_GrpB(j,i/2) + 1;
           if Dist_Ordered <= RearDist
                RearFarNumP_Ordered_All(j,i) = RearFarNumP_Ordered_All(j,i) + 1;
                RearFarNumP_Ordered_GrpB(j,i/2) = RearFarNumP_Ordered_GrpB(j,i/2) + 1;
            else
                RearCloseNumP_Ordered_All(j,i) = RearCloseNumP_Ordered_All(j,i) + 1;
                RearCloseNumP_Ordered_GrpB(j,i/2) = RearCloseNumP_Ordered_GrpB(j,i/2) + 1;
            end
        end
    else
% i is odd, Grp A
        if roundn(Lane_Ordered,-2) == -0.35
            LaneNumP_Ordered_All(j,(i-1)*3+1) = LaneNumP_Ordered_All(j,(i-1)*3+1) + 1;
            LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+1) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+1) + 1;
        elseif roundn(Lane_Ordered,-2) == 0.00
            LaneNumP_Ordered_All(j,(i-1)*3+2) = LaneNumP_Ordered_All(j,(i-1)*3+2) + 1;
            LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+2) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+2) + 1;
        elseif roundn(Lane_Ordered,-2) == 0.35
            LaneNumP_Ordered_All(j,(i-1)*3+3) = LaneNumP_Ordered_All(j,(i-1)*3+3) + 1;
            LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+3) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+3) + 1;
        end
    LaneNumP_Ordered_GrpA_Left(j,(i+1)/2) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+1);
    LaneNumP_Ordered_GrpA_Middle(j,(i+1)/2) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+2);
    LaneNumP_Ordered_GrpA_Right(j,(i+1)/2) = LaneNumP_Ordered_GrpA(j,((i+1)/2-1)*3+3);

        if Dist_Ordered >= 0
            FrontNumP_Ordered_All(j,i) = FrontNumP_Ordered_All(j,i) + 1;
            FrontNumP_Ordered_GrpA(j,(i+1)/2) = FrontNumP_Ordered_GrpA(j,(i+1)/2) + 1;
            if Dist_Ordered >= FrontDist
                FrontFarNumP_Ordered_All(j,i) = FrontFarNumP_Ordered_All(j,i) + 1;
                FrontFarNumP_Ordered_GrpA(j,(i+1)/2) = FrontFarNumP_Ordered_GrpA(j,(i+1)/2) + 1;
            else
                FrontCloseNumP_Ordered_All(j,i) = FrontCloseNumP_Ordered_All(j,i) + 1;
                FrontCloseNumP_Ordered_GrpA(j,(i+1)/2) = FrontCloseNumP_Ordered_GrpA(j,(i+1)/2) + 1;
            end
        end
        if Dist_Ordered <= 0
           RearNumP_Ordered_All(j,i) =  RearNumP_Ordered_All(j,i) + 1;
           RearNumP_Ordered_GrpA(j,(i+1)/2) = RearNumP_Ordered_GrpA(j,(i+1)/2) + 1;
           if Dist_Ordered <= RearDist
                RearFarNumP_Ordered_All(j,i) = RearFarNumP_Ordered_All(j,i) + 1;
                RearFarNumP_Ordered_GrpA(j,(i+1)/2) = RearFarNumP_Ordered_GrpA(j,(i+1)/2) + 1;
            else
                RearCloseNumP_Ordered_All(j,i) = RearCloseNumP_Ordered_All(j,i) + 1;
                RearCloseNumP_Ordered_GrpA(j,(i+1)/2) = RearCloseNumP_Ordered_GrpA(j,(i+1)/2) + 1;
            end
        end
    end
    LaneNumP_Ordered_All_Left(j,i) = LaneNumP_Ordered_All(j,(i-1)*3+1);
    LaneNumP_Ordered_All_Middle(j,i) = LaneNumP_Ordered_All(j,(i-1)*3+2);
    LaneNumP_Ordered_All_Right(j,i) = LaneNumP_Ordered_All(j,(i-1)*3+3);

    end
    end
%%  NumP_Scrambled_All
    if isnan(NumP_Scrambled_All(j,i))==0 %non-deleted data
        LaneNumP_Scrambled_All(j,(i-1)*3+1:(i-1)*3+3) = 0;
```

```
      FrontNumP_Scrambled_All(j,i) = 0;
      FrontFarNumP_Scrambled_All(j,i) = 0;
      FrontCloseNumP_Scrambled_All(j,i) = 0;
      RearNumP_Scrambled_All(j,i) = 0;
      RearFarNumP_Scrambled_All(j,i) = 0;
      RearCloseNumP_Scrambled_All(j,i) = 0;
      if mod(i, 2) == 0
% i is even, Grp B
      LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+1:(i/2-1)*3+3) = 0;
      FrontNumP_Scrambled_GrpB(j,i/2) = 0;
      FrontFarNumP_Scrambled_GrpB(j,i/2) = 0;
      FrontCloseNumP_Scrambled_GrpB(j,i/2) = 0;
      RearNumP_Scrambled_GrpB(j,i/2) = 0;
      RearFarNumP_Scrambled_GrpB(j,i/2) = 0;
      RearCloseNumP_Scrambled_GrpB(j,i/2) = 0;
      else
% i is odd, Grp A
      LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+1) = 0;
      FrontNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      FrontFarNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      FrontCloseNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      RearNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      RearFarNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      RearCloseNumP_Scrambled_GrpA(j,(i+1)/2) = 0;
      end

  for n = 1:NumP_Scrambled_All(j,i)
      Lane_Scrambled = LaneP_Scrambled_All(j,(i-1)*8+n);
      Dist_Scrambled = DistP_Scrambled_All(j,(i-1)*8+n);

  if mod(i, 2) == 0
% i is even, Grp B
      if roundn(Lane_Scrambled,-2) == -0.35
          LaneNumP_Scrambled_All(j,(i-1)*3+1) = LaneNumP_Scrambled_All(j,(i-1)*3+1) + 1;
          LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+1) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+1) + 1;
      elseif roundn(Lane_Scrambled,-2) == 0.00
          LaneNumP_Scrambled_All(j,(i-1)*3+2) = LaneNumP_Scrambled_All(j,(i-1)*3+2) + 1;
          LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+2) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+2) + 1;
      elseif roundn(Lane_Scrambled,-2) == 0.35
          LaneNumP_Scrambled_All(j,(i-1)*3+3) = LaneNumP_Scrambled_All(j,(i-1)*3+3) + 1;
          LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+3) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+3) + 1;
      end
  LaneNumP_Scrambled_GrpB_Left(j,i/2) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+1);
  LaneNumP_Scrambled_GrpB_Middle(j,i/2) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+2);
  LaneNumP_Scrambled_GrpB_Right(j,i/2) = LaneNumP_Scrambled_GrpB(j,(i/2-1)*3+3);

      if Dist_Scrambled >= 0
         FrontNumP_Scrambled_All(j,i) = FrontNumP_Scrambled_All(j,i) + 1;
         FrontNumP_Scrambled_GrpB(j,i/2) = FrontNumP_Scrambled_GrpB(j,i/2) + 1;
         if Dist_Scrambled >= FrontDist
              FrontFarNumP_Scrambled_All(j,i) = FrontFarNumP_Scrambled_All(j,i) + 1;
              FrontFarNumP_Scrambled_GrpB(j,i/2) = FrontFarNumP_Scrambled_GrpB(j,i/2) + 1;
          else
              FrontCloseNumP_Scrambled_All(j,i) = FrontCloseNumP_Scrambled_All(j,i) + 1;
              FrontCloseNumP_Scrambled_GrpB(j,i/2) = FrontCloseNumP_Scrambled_GrpB(j,i/2) + 1;
          end
      end
      if Dist_Scrambled <= 0
         RearNumP_Scrambled_All(j,i) = RearNumP_Scrambled_All(j,i) + 1;
         RearNumP_Scrambled_GrpB(j,i/2) = RearNumP_Scrambled_GrpB(j,i/2) + 1;
         if Dist_Scrambled <= RearDist
              RearFarNumP_Scrambled_All(j,i) = RearFarNumP_Scrambled_All(j,i) + 1;
              RearFarNumP_Scrambled_GrpB(j,i/2) = RearFarNumP_Scrambled_GrpB(j,i/2) + 1;
          else
              RearCloseNumP_Scrambled_All(j,i) = RearCloseNumP_Scrambled_All(j,i) + 1;
              RearCloseNumP_Scrambled_GrpB(j,i/2) = RearCloseNumP_Scrambled_GrpB(j,i/2) + 1;
          end
      end
   else
% i is odd, Grp A
      if roundn(Lane_Scrambled,-2) == -0.35
          LaneNumP_Scrambled_All(j,(i-1)*3+1) = LaneNumP_Scrambled_All(j,(i-1)*3+1) + 1;
          LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+1) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+1) + 1;
      elseif roundn(Lane_Scrambled,-2) == 0.00
          LaneNumP_Scrambled_All(j,(i-1)*3+2) = LaneNumP_Scrambled_All(j,(i-1)*3+2) + 1;
          LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+2) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+2) + 1;
      elseif roundn(Lane_Scrambled,-2) == 0.35
          LaneNumP_Scrambled_All(j,(i-1)*3+3) = LaneNumP_Scrambled_All(j,(i-1)*3+3) + 1;
          LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+3) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+3) + 1;
      end
  LaneNumP_Scrambled_GrpA_Left(j,(i+1)/2) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+1);
  LaneNumP_Scrambled_GrpA_Middle(j,(i+1)/2) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+2);
```

```
    LaneNumP_Scrambled_GrpA_Right(j,(i+1)/2) = LaneNumP_Scrambled_GrpA(j,((i+1)/2-1)*3+3);

        if Dist_Scrambled >= 0
           FrontNumP_Scrambled_All(j,i) = FrontNumP_Scrambled_All(j,i) + 1;
           FrontNumP_Scrambled_GrpA(j,(i+1)/2) = FrontNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
           if Dist_Scrambled >= FrontDist
                FrontFarNumP_Scrambled_All(j,i) = FrontFarNumP_Scrambled_All(j,i) + 1;
                FrontFarNumP_Scrambled_GrpA(j,(i+1)/2) = FrontFarNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
            else
                FrontCloseNumP_Scrambled_All(j,i) = FrontCloseNumP_Scrambled_All(j,i) + 1;
                FrontCloseNumP_Scrambled_GrpA(j,(i+1)/2) = FrontCloseNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
            end
        end
        if Dist_Scrambled <= 0
           RearNumP_Scrambled_All(j,i) = RearNumP_Scrambled_All(j,i) + 1;
           RearNumP_Scrambled_GrpA(j,(i+1)/2) = RearNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
           if Dist_Scrambled <= RearDist
                RearFarNumP_Scrambled_All(j,i) = RearFarNumP_Scrambled_All(j,i) + 1;
                RearFarNumP_Scrambled_GrpA(j,(i+1)/2) = RearFarNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
            else
                RearCloseNumP_Scrambled_All(j,i) = RearCloseNumP_Scrambled_All(j,i) + 1;
                RearCloseNumP_Scrambled_GrpA(j,(i+1)/2) = RearCloseNumP_Scrambled_GrpA(j,(i+1)/2) + 1;
            end
        end
    end
    end
    LaneNumP_Scrambled_All_Left(j,i) = LaneNumP_Scrambled_All(j,(i-1)*3+1);
    LaneNumP_Scrambled_All_Middle(j,i) = LaneNumP_Scrambled_All(j,(i-1)*3+2);
    LaneNumP_Scrambled_All_Right(j,i) = LaneNumP_Scrambled_All(j,(i-1)*3+3);
    end
 %% NumA_Ordered
    for l = 1:NumA_Ordered(j)
        DistA_Ordered(j,l) = AnswersBig2(j,l*3+2)-860;
        LaneA_Ordered(j,l) = AnswersBig2(j,l*3+1);
        SpeedA_Ordered(j,l) = AnswersBig2(j,l*3+3);

        if DistA_Ordered(j,l) >= 0
            FrontNumA_Ordered(j,i) =  FrontNumA_Ordered(j,i) + 1;
            if DistA_Ordered(j,l) >= FrontDist
               FrontFarNumA_Ordered(j,i) = FrontFarNumA_Ordered(j,i) + 1;
            else
               FrontCloseNumA_Ordered(j,i) = FrontCloseNumA_Ordered(j,i) + 1;
            end
        end
        if DistA_Ordered(j,l) <= 0
            RearNumA_Ordered(j,i) =  RearNumA_Ordered(j,i) + 1;
            if DistA_Ordered(j,l) <= RearDist
                RearFarNumA_Ordered(j,i) = RearFarNumA_Ordered(j,i) + 1;
            else
                RearCloseNumA_Ordered(j,i) = RearCloseNumA_Ordered(j,i) + 1;
            end
        end
    end
%% NumA_Scrambled
    for o = 1:NumA_Scrambled_All(j,i)
        DistA_Scrambled_All(j,(i-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+2)-860;
        LaneA_Scrambled_All(j,(i-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+1);
        SpeedA_Scrambled_All(j,(i-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+3);
        LaneNumA_Scrambled_All(j,(i-1)*3+1:(i-1)*3+3) = LaneNumA_Ordered(seq_i(j,i),:);
        LaneNumA_Scrambled_All_Left(j,i)= LaneNumA_Scrambled_All(j,(i-1)*3+1);
        LaneNumA_Scrambled_All_Middle(j,i)= LaneNumA_Scrambled_All(j,(i-1)*3+2);
        LaneNumA_Scrambled_All_Right(j,i)= LaneNumA_Scrambled_All(j,(i-1)*3+3);

        if mod(i, 2) == 0
  % i is even, Grp B
        DistA_Scrambled_GrpB(j,(i/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+2)-860;
        LaneA_Scrambled_GrpB(j,(i/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+1);
        SpeedA_Scrambled_GrpB(j,(i/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+3);
        LaneNumA_Scrambled_GrpB(j,(i/2-1)*3+1:(i/2-1)*3+3) = LaneNumA_Scrambled_All(j,(i-1)*3+1:(i-1)*3+3);
        LaneNumA_Scrambled_GrpB_Left(j,i/2) = LaneNumA_Scrambled_GrpB(j,(i/2-1)*3+1);
        LaneNumA_Scrambled_GrpB_Middle(j,i/2) = LaneNumA_Scrambled_GrpB(j,(i/2-1)*3+2);
        LaneNumA_Scrambled_GrpB_Right(j,i/2) =LaneNumA_Scrambled_GrpB(j,(i/2-1)*3+3);

        if DistA_Scrambled_All(j,(i-1)*8+o) >= 0
            FrontNumA_Scrambled_All(j,i) = FrontNumA_Scrambled_All(j,i) + 1;
            FrontNumA_Scrambled_GrpB(j,i/2) = FrontNumA_Scrambled_GrpB(j,i/2) + 1;
            if DistA_Scrambled_All(j,(i-1)*8+o) >= FrontDist
                FrontFarNumA_Scrambled_All(j,i) = FrontFarNumA_Scrambled_All(j,i) + 1;
                FrontFarNumA_Scrambled_GrpB(j,i/2) = FrontFarNumA_Scrambled_GrpB(j,i/2) + 1;
            else
                FrontCloseNumA_Scrambled_All(j,i) = FrontCloseNumA_Scrambled_All(j,i) + 1;
                FrontCloseNumA_Scrambled_GrpB(j,i/2) = FrontCloseNumA_Scrambled_GrpB(j,i/2) + 1;
```

```
            end
        end
        if DistA_Scrambled_All(j,(i-1)*8+o) <= 0
            RearNumA_Scrambled_All(j,i) = RearNumA_Scrambled_All(j,i) + 1;
            RearNumA_Scrambled_GrpB(j,i/2) = RearNumA_Scrambled_GrpB(j,i/2) + 1;
            if DistA_Scrambled_All(j,(i-1)*8+o) <= RearDist
                RearFarNumA_Scrambled_All(j,i) = RearFarNumA_Scrambled_All(j,i) + 1;
                RearFarNumA_Scrambled_GrpB(j,i/2) = RearFarNumA_Scrambled_GrpB(j,i/2) + 1;
            else
                RearCloseNumA_Scrambled_All(j,i) = RearCloseNumA_Scrambled_All(j,i) + 1;
                RearCloseNumA_Scrambled_GrpB(j,i/2) = RearCloseNumA_Scrambled_GrpB(j,i/2) + 1;
            end
        end
        else
  % i is odd, Grp A
DistA_Scrambled_GrpA(j,((i+1)/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+2)-860;
LaneA_Scrambled_GrpA(j,((i+1)/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+1);
SpeedA_Scrambled_GrpA(j,((i+1)/2-1)*8+o) = AnswersBig2(seq_i(j,i),o*3+3);
LaneNumA_Scrambled_GrpA(j,((i+1)/2-1)*3+1:((i+1)/2-1)*3+3) = LaneNumA_Scrambled_All(j,(i-1)*3+1:(i-1)*3+3);
LaneNumA_Scrambled_GrpA_Left(j,(i+1)/2) = LaneNumA_Scrambled_GrpA(j,((i+1)/2-1)*3+1);
LaneNumA_Scrambled_GrpA_Middle(j,(i+1)/2) = LaneNumA_Scrambled_GrpA(j,((i+1)/2-1)*3+2);
LaneNumA_Scrambled_GrpA_Right(j,(i+1)/2) = LaneNumA_Scrambled_GrpA(j,((i+1)/2-1)*3+3);

        if DistA_Scrambled_All(j,(i-1)*8+o) >= 0
            FrontNumA_Scrambled_All(j,i) = FrontNumA_Scrambled_All(j,i) + 1;
            FrontNumA_Scrambled_GrpA(j,(i+1)/2) = FrontNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            if DistA_Scrambled_All(j,(i-1)*8+o) >= FrontDist
                FrontFarNumA_Scrambled_All(j,i) = FrontFarNumA_Scrambled_All(j,i) + 1;
                FrontFarNumA_Scrambled_GrpA(j,(i+1)/2) = FrontFarNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            else
                FrontCloseNumA_Scrambled_All(j,i) = FrontCloseNumA_Scrambled_All(j,i) + 1;
                FrontCloseNumA_Scrambled_GrpA(j,(i+1)/2) = FrontCloseNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            end
        end
        if DistA_Scrambled_All(j,(i-1)*8+o) <= 0
            RearNumA_Scrambled_All(j,i) = RearNumA_Scrambled_All(j,i) + 1;
            RearNumA_Scrambled_GrpA(j,(i+1)/2) = RearNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            if DistA_Scrambled_All(j,(i-1)*8+o) <= RearDist
                RearFarNumA_Scrambled_All(j,i) = RearFarNumA_Scrambled_All(j,i) + 1;
                RearFarNumA_Scrambled_GrpA(j,(i+1)/2) = RearFarNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            else
                RearCloseNumA_Scrambled_All(j,i) = RearCloseNumA_Scrambled_All(j,i) + 1;
                RearCloseNumA_Scrambled_GrpA(j,(i+1)/2) = RearCloseNumA_Scrambled_GrpA(j,(i+1)/2) + 1;
            end
        end
        end
    end
    count1_A = 0; count1_A_4 = 0; count1_A_6 = 0; count1_B = 0; count1_B_4 = 0; count1_B_6 = 0;
    count3_A = 0; count3_A_4 = 0; count3_A_6 = 0; count3_B = 0; count3_B_4 = 0; count3_B_6 = 0;
    count7_A = 0; count7_A_4 = 0; count7_A_6 = 0; count7_B = 0; count7_B_4 = 0; count7_B_6 = 0;
    count9_A = 0; count9_A_4 = 0; count9_A_6 = 0; count9_B = 0; count9_B_4 = 0; count9_B_6 = 0;
    count12_A = 0; count12_A_4 = 0; count12_A_6 = 0; count12_B = 0; count12_B_4 = 0; count12_B_6 = 0;
    count20_A = 0; count20_A_4 = 0; count20_A_6 = 0; count20_B = 0; count20_B_4 = 0; count20_B_6 = 0;

%% Durations
for p = 1:24
    if mod(i, 2) == 0
*
```

```
  % i is even, Grp B
        if Duration_Ordered_GrpB(p+2,1) == 1
            count1_B = count1_B + 1;
        Diff_Duration_All((i-1)*4+count1_B,1) = Diff_Ordered_All(p+2,i);
        Diff_Duration_GrpB((i/2-1)*4+count1_B,1) = Diff_Ordered_GrpB(p+2,i/2);
        Time_Duration_All((i-1)*4+count1_B,1) = Time_Ordered_All(p+2,i);
        Time_Duration_GrpB((i/2-1)*4+count1_B,1) = Time_Ordered_GrpB(p+2,i/2);
        NumP_Duration_All((i-1)*4+count1_B,1) = NumP_Ordered_All(p+2,i);
        NumP_Duration_GrpB((i/2-1)*4+count1_B,1) = NumP_Ordered_GrpB(p+2,i/2);
        LaneNumP_Duration_All_Left((i-1)*4+count1_B,1) = LaneNumP_Ordered_All_Left(p+2,i);
        LaneNumP_Duration_GrpB_Left((i/2-1)*4+count1_B,1) = LaneNumP_Ordered_GrpB_Left(p+2,i/2);
        LaneNumP_Duration_All_Middle((i-1)*4+count1_B,1) = LaneNumP_Ordered_All_Middle(p+2,i);
        LaneNumP_Duration_GrpB_Middle((i/2-1)*4+count1_B,1) = LaneNumP_Ordered_GrpB_Middle(p+2,i/2);
        LaneNumP_Duration_All_Right((i-1)*4+count1_B,1) = LaneNumP_Ordered_All_Right(p+2,i);
        LaneNumP_Duration_GrpB_Right((i/2-1)*4+count1_B,1) = LaneNumP_Ordered_GrpB_Right(p+2,i/2);
        FrontNumP_Duration_All((i-1)*4+count1_B,1) = FrontNumP_Ordered_All(p+2,i);
        FrontNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontNumP_Ordered_GrpB(p+2,i/2);
        RearNumP_Duration_All((i-1)*4+count1_B,1) = RearNumP_Ordered_All(p+2,i);
        RearNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = RearNumP_Ordered_GrpB(p+2,i/2);
        FrontCloseNumP_Duration_All((i-1)*4+count1_B,1) = FrontCloseNumP_Ordered_All(p+2,i);
        FrontCloseNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontCloseNumP_Ordered_GrpB(p+2,i/2);
        FrontFarNumP_Duration_All((i-1)*4+count1_B,1) = FrontFarNumP_Ordered_All(p+2,i);
        FrontFarNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontFarNumP_Ordered_GrpB(p+2,i/2);
```

```
RearCloseNumP_Duration_All((i-1)*4+count1_B,1) = RearCloseNumP_Ordered_All(p+2,i);
RearCloseNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = RearCloseNumP_Ordered_GrpB(p+2,i/2);
RearFarNumP_Duration_All((i-1)*4+count1_B,1) = RearFarNumP_Ordered_All(p+2,i);
RearFarNumP_Duration_GrpB((i/2-1)*4+count1_B,1) = RearFarNumP_Ordered_GrpB(p+2,i/2);
NumA_Duration_All((i-1)*4+count1_B,1) = NumA_Ordered(p+2);
NumA_Duration_GrpB((i/2-1)*4+count1_B,1) = NumA_Ordered(p+2);
LaneNumA_Duration_All_Left((i-1)*4+count1_B,1) = LaneNumA_Ordered_Left(p+2);
LaneNumA_Duration_GrpB_Left((i/2-1)*4+count1_B,1) = LaneNumA_Ordered_Left(p+2);
LaneNumA_Duration_All_Middle((i-1)*4+count1_B,1) = LaneNumA_Ordered_Middle(p+2);
LaneNumA_Duration_GrpB_Middle((i/2-1)*4+count1_B,1) = LaneNumA_Ordered_Middle(p+2);
LaneNumA_Duration_All_Right((i-1)*4+count1_B,1) = LaneNumA_Ordered_Right(p+2);
LaneNumA_Duration_GrpB_Right((i/2-1)*4+count1_B,1) = LaneNumA_Ordered_Right(p+2);
FrontNumA_Duration_All((i-1)*4+count1_B,1) = FrontNumA_Ordered(p+2,i);
FrontNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontNumA_Ordered(p+2,i);
RearNumA_Duration_All((i-1)*4+count1_B,1) = RearNumA_Ordered(p+2,i);
RearNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = RearNumA_Ordered(p+2,i);
FrontCloseNumA_Duration_All((i-1)*4+count1_B,1) = FrontCloseNumA_Ordered(p+2,i);
FrontCloseNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontCloseNumA_Ordered(p+2,i);
FrontFarNumA_Duration_All((i-1)*4+count1_B,1) = FrontFarNumA_Ordered(p+2,i);
FrontFarNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = FrontFarNumA_Ordered(p+2,i);
RearCloseNumA_Duration_All((i-1)*4+count1_B,1) = RearCloseNumA_Ordered(p+2,i);
RearCloseNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = RearCloseNumA_Ordered(p+2,i);
RearFarNumA_Duration_All((i-1)*4+count1_B,1) = RearFarNumA_Ordered(p+2,i);
RearFarNumA_Duration_GrpB((i/2-1)*4+count1_B,1) = RearFarNumA_Ordered(p+2,i);

    if p<=12 %6 cars
        count1_B_6 = count1_B_6 +1;
Diff_Duration_All_6cars((i-1)*2+count1_B_6,1) = Diff_Ordered_All(p+2,i);
Diff_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = Diff_Ordered_GrpB(p+2,i/2);
Time_Duration_All_6cars((i-1)*2+count1_B_6,1) = Time_Ordered_All(p+2,i);
Time_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = Time_Ordered_GrpB(p+2,i/2);
NumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = NumP_Ordered_All(p+2,i);
NumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = NumP_Ordered_GrpB(p+2,i/2);
LaneNumP_Duration_All_Left_6cars((i-1)*2+count1_B_6,1) = LaneNumP_Ordered_All_Left(p+2,i);
LaneNumP_Duration_GrpB_Left_6cars((i/2-1)*2+count1_B_6,1) = LaneNumP_Ordered_GrpB_Left(p+2,i/2);
LaneNumP_Duration_All_Middle_6cars((i-1)*2+count1_B_6,1) = LaneNumP_Ordered_All_Middle(p+2,i);
LaneNumP_Duration_GrpB_Middle_6cars((i/2-1)*2+count1_B_6,1) = LaneNumP_Ordered_GrpB_Middle(p+2,i/2);
LaneNumP_Duration_All_Right_6cars((i-1)*2+count1_B_6,1) = LaneNumP_Ordered_All_Right(p+2,i);
LaneNumP_Duration_GrpB_Right_6cars((i/2-1)*2+count1_B_6,1) = LaneNumP_Ordered_GrpB_Right(p+2,i/2);
FrontNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontNumP_Ordered_All(p+2,i);
FrontNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontNumP_Ordered_GrpB(p+2,i/2);
RearNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearNumP_Ordered_All(p+2,i);
RearNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearNumP_Ordered_GrpB(p+2,i/2);
FrontCloseNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontCloseNumP_Ordered_All(p+2,i);
FrontCloseNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontCloseNumP_Ordered_GrpB(p+2,i/2);
FrontFarNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontFarNumP_Ordered_All(p+2,i);
FrontFarNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontFarNumP_Ordered_GrpB(p+2,i/2);
RearCloseNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearCloseNumP_Ordered_All(p+2,i);
RearCloseNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearCloseNumP_Ordered_GrpB(p+2,i/2);
RearFarNumP_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearFarNumP_Ordered_All(p+2,i);
RearFarNumP_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearFarNumP_Ordered_GrpB(p+2,i/2);
NumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = NumA_Ordered(p+2);
NumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = NumA_Ordered(p+2);
LaneNumA_Duration_All_Left_6cars((i-1)*2+count1_B_6,1) = LaneNumA_Ordered_Left(p+2);
LaneNumA_Duration_GrpB_Left_6cars((i/2-1)*2+count1_B_6,1) = LaneNumA_Ordered_Left(p+2);
LaneNumA_Duration_All_Middle_6cars((i-1)*2+count1_B_6,1) = LaneNumA_Ordered_Middle(p+2);
LaneNumA_Duration_GrpB_Middle_6cars((i/2-1)*2+count1_B_6,1) = LaneNumA_Ordered_Middle(p+2);
LaneNumA_Duration_All_Right_6cars((i-1)*2+count1_B_6,1) = LaneNumA_Ordered_Right(p+2);
LaneNumA_Duration_GrpB_Right_6cars((i/2-1)*2+count1_B_6,1) = LaneNumA_Ordered_Right(p+2);
FrontNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontNumA_Ordered(p+2,i);
FrontNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontNumA_Ordered(p+2,i);
RearNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearNumA_Ordered(p+2,i);
RearNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearNumA_Ordered(p+2,i);
FrontCloseNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontCloseNumA_Ordered(p+2,i);
FrontCloseNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontCloseNumA_Ordered(p+2,i);
FrontFarNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = FrontFarNumA_Ordered(p+2,i);
FrontFarNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = FrontFarNumA_Ordered(p+2,i);
RearCloseNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearCloseNumA_Ordered(p+2,i);
RearCloseNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearCloseNumA_Ordered(p+2,i);
RearFarNumA_Duration_All_6cars((i-1)*2+count1_B_6,1) = RearFarNumA_Ordered(p+2,i);
RearFarNumA_Duration_GrpB_6cars((i/2-1)*2+count1_B_6,1) = RearFarNumA_Ordered(p+2,i);

    elseif p>12    %4 cars
        count1_B_4 = count1_B_4 +1;
Diff_Duration_All_4cars((i-1)*2+count1_B_4,1) = Diff_Ordered_All(p+2,i);
Diff_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = Diff_Ordered_GrpB(p+2,i/2);
Time_Duration_All_4cars((i-1)*2+count1_B_4,1) = Time_Ordered_All(p+2,i);
Time_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = Time_Ordered_GrpB(p+2,i/2);
NumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = NumP_Ordered_All(p+2,i);
NumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = NumP_Ordered_GrpB(p+2,i/2);
LaneNumP_Duration_All_Left_4cars((i-1)*2+count1_B_4,1) = LaneNumP_Ordered_All_Left(p+2,i);
LaneNumP_Duration_GrpB_Left_4cars((i/2-1)*2+count1_B_4,1) = LaneNumP_Ordered_GrpB_Left(p+2,i/2);
```

```
        LaneNumP_Duration_All_Middle_4cars((i-1)*2+count1_B_4,1) = LaneNumP_Ordered_All_Middle(p+2,i);
        LaneNumP_Duration_GrpB_Middle_4cars((i/2-1)*2+count1_B_4,1) = LaneNumP_Ordered_GrpB_Middle(p+2,i/2);
        LaneNumP_Duration_All_Right_4cars((i-1)*2+count1_B_4,1) = LaneNumP_Ordered_All_Right(p+2,i);
        LaneNumP_Duration_GrpB_Right_4cars((i/2-1)*2+count1_B_4,1) = LaneNumP_Ordered_GrpB_Right(p+2,i/2);
        FrontNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontNumP_Ordered_All(p+2,i);
        FrontNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontNumP_Ordered_GrpB(p+2,i/2);
        RearNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearNumP_Ordered_All(p+2,i);
        RearNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearNumP_Ordered_GrpB(p+2,i/2);
        FrontCloseNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontCloseNumP_Ordered_All(p+2,i);
        FrontCloseNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontCloseNumP_Ordered_GrpB(p+2,i/2);
        FrontFarNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontFarNumP_Ordered_All(p+2,i);
        FrontFarNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontFarNumP_Ordered_GrpB(p+2,i/2);
        RearCloseNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearCloseNumP_Ordered_All(p+2,i);
        RearCloseNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearCloseNumP_Ordered_GrpB(p+2,i/2);
        RearFarNumP_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearFarNumP_Ordered_All(p+2,i);
        RearFarNumP_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearFarNumP_Ordered_GrpB(p+2,i/2);
        NumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = NumA_Ordered(p+2);
        NumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = NumA_Ordered(p+2);
        LaneNumA_Duration_All_Left_4cars((i-1)*2+count1_B_4,1) = LaneNumA_Ordered_Left(p+2);
        LaneNumA_Duration_GrpB_Left_4cars((i/2-1)*2+count1_B_4,1) = LaneNumA_Ordered_Left(p+2);
        LaneNumA_Duration_All_Middle_4cars((i-1)*2+count1_B_4,1) = LaneNumA_Ordered_Middle(p+2);
        LaneNumA_Duration_GrpB_Middle_4cars((i/2-1)*2+count1_B_4,1) = LaneNumA_Ordered_Middle(p+2);
        LaneNumA_Duration_All_Right_4cars((i-1)*2+count1_B_4,1) = LaneNumA_Ordered_Right(p+2);
        LaneNumA_Duration_GrpB_Right_4cars((i/2-1)*2+count1_B_4,1) = LaneNumA_Ordered_Right(p+2);
        FrontNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontNumA_Ordered(p+2,i);
        FrontNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontNumA_Ordered(p+2,i);
        RearNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearNumA_Ordered(p+2,i);
        RearNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearNumA_Ordered(p+2,i);
        FrontCloseNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontCloseNumA_Ordered(p+2,i);
        FrontCloseNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontCloseNumA_Ordered(p+2,i);
        FrontFarNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = FrontFarNumA_Ordered(p+2,i);
        FrontFarNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = FrontFarNumA_Ordered(p+2,i);
        RearCloseNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearCloseNumA_Ordered(p+2,i);
        RearCloseNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearCloseNumA_Ordered(p+2,i);
        RearFarNumA_Duration_All_4cars((i-1)*2+count1_B_4,1) = RearFarNumA_Ordered(p+2,i);
        RearFarNumA_Duration_GrpB_4cars((i/2-1)*2+count1_B_4,1) = RearFarNumA_Ordered(p+2,i);
            end
```

*

```
%% averaging for statistical tests
Avg_Diff_Duration_All = nan(34,6);
Avg_Diff_Duration_4cars = nan(34,6);
Avg_Diff_Duration_6cars = nan(34,6);
Avg_Time_Duration_All = nan(34,6);
Avg_Time_Duration_4cars = nan(34,6);
Avg_Time_Duration_6cars = nan(34,6);
for i = 1:34
    Avg_Diff_Duration_All(i,:) = nanmean(Diff_Duration_All(4*i-3:4*i,:));
    Avg_Diff_Duration_4cars(i,:) = nanmean(Diff_Duration_All_4cars(2*i-1:2*i,:));
    Avg_Diff_Duration_6cars(i,:) = nanmean(Diff_Duration_All_6cars(2*i-1:2*i,:));
        Avg_Time_Duration_All(i,:) = nanmean(Time_Duration_All(4*i-3:4*i,:));
    Avg_Time_Duration_4cars(i,:) = nanmean(Time_Duration_All_4cars(2*i-1:2*i,:));
    Avg_Time_Duration_6cars(i,:) = nanmean(Time_Duration_All_6cars(2*i-1:2*i,:));
end
Avg_Diff_4cars = nanmean(Avg_Diff_Duration_4cars,2);
Avg_Diff_6cars = nanmean(Avg_Diff_Duration_6cars,2);
Avg_Time_4cars = nanmean(Avg_Time_Duration_4cars,2);
Avg_Time_6cars = nanmean(Avg_Time_Duration_6cars,2);
%% Saving all data in one file
save('AllData.mat')
```

* The code between the red lines is repeated for each duration (i.e., video length), and again for group A.

Appendix A6-4: Basic Measure calculation (BasicMeasures.m)

This code was used to calculate the basic score.

```
% Made by Xander Coster
% 1268546
% Calculating Basic Measures`
load('AllData.mat')
%% Absolute error in total number of cars:
CarnumError_Ordered_All = abs(NumP_Ordered_All(3:end,:) - repmat(NumA_Ordered(3:end),1,34));
CarnumErrorPerc_Ordered_All = CarnumError_Ordered_All ./ repmat(NumA_Ordered(3:end),1,34);
Total_CarnumError_Ordered_All_PPart = nansum(CarnumError_Ordered_All,1);
Mean_CarnumError_Ordered_All_PPart = nanmean(CarnumError_Ordered_All,1);
Total_CarnumError_Ordered_All_PScene= nansum(CarnumError_Ordered_All,2);
Mean_CarnumError_Ordered_All_PScene= nanmean(CarnumError_Ordered_All,2);
Total_CarnumError_Ordered_All = nansum(nansum(CarnumError_Ordered_All));
Mean_CarnumError_Ordered_All = nanmean(nanmean(CarnumError_Ordered_All));
CarnumError_Duration_All = abs(NumP_Duration_All - NumA_Duration_All);
CarnumError_Duration_All_4cars = abs(NumP_Duration_All_4cars - NumA_Duration_All_4cars);
CarnumErrorPerc_Duration_All_4cars = CarnumError_Duration_All_4cars./NumA_Duration_All_4cars;
CarnumError_Duration_All_6cars = abs(NumP_Duration_All_6cars - NumA_Duration_All_6cars);
CarnumErrorPerc_Duration_All_6cars = CarnumError_Duration_All_6cars./NumA_Duration_All_6cars;
CarnumError_Ordered_GrpA = abs(NumP_Ordered_GrpA(3:end,:) - repmat(NumA_Ordered(3:end),1,17));
CarnumErrorPerc_Ordered_GrpA = CarnumError_Ordered_GrpA./repmat(NumA_Ordered(3:end),1,17);
Total_CarnumError_Ordered_GrpA_PPart = nansum(CarnumError_Ordered_GrpA,1);
Mean_CarnumError_Ordered_GrpA_PPart = nanmean(CarnumError_Ordered_GrpA,1);
Total_CarnumError_Ordered_GrpA_PScene = nansum(CarnumError_Ordered_GrpA,2);
Mean_CarnumError_Ordered_GrpA_PScene = nanmean(CarnumError_Ordered_GrpA,2);
Total_CarnumError_Ordered_GrpA = nansum(nansum(CarnumError_Ordered_GrpA));
Mean_CarnumError_Ordered_GrpA = nanmean(nanmean(CarnumError_Ordered_GrpA));
CarnumError_Duration_GrpA = abs(NumP_Duration_GrpA - NumA_Duration_GrpA);
CarnumErrorPerc_Duration_GrpA = CarnumError_Duration_GrpA./NumA_Duration_GrpA;
CarnumError_Ordered_GrpB = abs(NumP_Ordered_GrpB(3:end,:) - repmat(NumA_Ordered(3:end),1,17));
CarnumErrorPerc_Ordered_GrpB = CarnumError_Ordered_GrpB./repmat(NumA_Ordered(3:end),1,17);
Total_CarnumError_Ordered_GrpB_PPart = nansum(CarnumError_Ordered_GrpB,1);
Mean_CarnumError_Ordered_GrpB_PPart = nanmean(CarnumError_Ordered_GrpB,1);
Total_CarnumError_Ordered_GrpB_PScene = nansum(CarnumError_Ordered_GrpB,2);
Mean_CarnumError_Ordered_GrpB_PScene = nanmean(CarnumError_Ordered_GrpB,2);
Total_CarnumError_Ordered_GrpB = nansum(nansum(CarnumError_Ordered_GrpB));
Mean_CarnumError_Ordered_GrpB = nanmean(nanmean(CarnumError_Ordered_GrpB));
CarnumError_Duration_GrpB = abs(NumP_Duration_GrpB - NumA_Duration_GrpB);
CarnumErrorPerc_Duration_GrpB = CarnumError_Duration_GrpB./NumA_Duration_GrpB;
CarnumError_Scrambled_All = abs(NumP_Scrambled_All(3:end,:) - NumA_Scrambled_All(3:end,:));
Total_CarnumError_Scrambled_All_PPart = nansum(CarnumError_Scrambled_All,1);
CarnumErrorPerc_Scrambled_All = CarnumError_Scrambled_All ./ NumA_Scrambled_All(3:end,:);
CarnumErrorPerc_Scrambled_6cars = nan(24,34);
CarnumErrorPerc_Scrambled_4cars = nan(24,34);
for i = 1:34
    for j = 1:24
        if NumA_Scrambled_All(j+2,i) == 6
            CarnumErrorPerc_Scrambled_6cars(j,i) = CarnumErrorPerc_Scrambled_All(j,i);
        elseif NumA_Scrambled_All(j+2,i) == 4
            CarnumErrorPerc_Scrambled_4cars(j,i) = CarnumErrorPerc_Scrambled_All(j,i);
        end
    end
end
Mean_CarnumError_Scrambled_All_PPart = nanmean(CarnumError_Scrambled_All,1);
Total_CarnumError_Scrambled_All_PScene= nansum(CarnumError_Scrambled_All,2);
Mean_CarnumError_Scrambled_All_PScene= nanmean(CarnumError_Scrambled_All,2);
Total_CarnumError_Scrambled_All = nansum(nansum(CarnumError_Scrambled_All));
Mean_CarnumError_Scrambled_All = nanmean(nanmean(CarnumError_Scrambled_All));
    %Scrambled Group A
CarnumError_Scrambled_GrpA = abs(NumP_Scrambled_GrpA(3:end,:) - NumA_Scrambled_GrpA(3:end,:));
Total_CarnumError_Scrambled_GrpA_PPart = nansum(CarnumError_Scrambled_GrpA,1);
Mean_CarnumError_Scrambled_GrpA_PPart = nanmean(CarnumError_Scrambled_GrpA,1);
Total_CarnumError_Scrambled_GrpA_PScene= nansum(CarnumError_Scrambled_GrpA,2);
Mean_CarnumError_Scrambled_GrpA_PScene= nanmean(CarnumError_Scrambled_GrpA,2);
Total_CarnumError_Scrambled_GrpA = nansum(nansum(CarnumError_Scrambled_GrpA));
Mean_CarnumError_Scrambled_GrpA = nanmean(nanmean(CarnumError_Scrambled_GrpA));
    %Scrambled Group B
CarnumError_Scrambled_GrpB = abs(NumP_Scrambled_GrpB(3:end,:) - NumA_Scrambled_GrpB(3:end,:));
Total_CarnumError_Scrambled_GrpB_PPart = nansum(CarnumError_Scrambled_GrpB,1);
Mean_CarnumError_Scrambled_GrpB_PPart = nanmean(CarnumError_Scrambled_GrpB,1);
Total_CarnumError_Scrambled_GrpB_PScene= nansum(CarnumError_Scrambled_GrpB,2);
Mean_CarnumError_Scrambled_GrpB_PScene= nanmean(CarnumError_Scrambled_GrpB,2);
Total_CarnumError_Scrambled_GrpB = nansum(nansum(CarnumError_Scrambled_GrpB));
Mean_CarnumError_Scrambled_GrpB = nanmean(nanmean(CarnumError_Scrambled_GrpB));

%% Saving
save('BasicMeasures.mat')
```

Appendix B: Material Regarding the Preliminary (Pilot) Study

Appendix B1: Report

A Method to Determine How Much Time People Need to Understand Traffic Situations: A Pilot Study

The goal of this study was to find out what video lengths should be used for the full-scale experiment, next to test and improve the experimental design of the full-scale experiment. The purpose of the full-scale study is to investigate if the method used by Gugerty [1] is suitable for finding out how much time people need to regain situation awareness (SA) in traffic situations.

Setup

Seven individuals (all male), aged between 20 and 27 (M = 23.9, SD = 3.0 years), participated in this experiment and were requested to watch videos of simulated traffic scenes, and to assess the traffic scenes. They were asked to stop the videos when they had assessed the situation to such an extent that they would feel safe to take over control of the vehicle, by pressing the spacebar. When a video was not stopped by the participant, it stopped automatically after 30 seconds. Directly after a video stopped, participants were asked to reproduce the traffic scenario in a top-down view, and to fill out a questionnaire.

The experiment consisted of twelve videos of traffic scenarios, two of which were used as training. Training trials contained three cars of surrounding traffic, experimental trials contained either four or six cars of surrounding traffic (both numbers used five times). The experiment lasted approximately 30 minutes.

The ten experimental trials were randomized by Matlab, the sequence of videos this provided was used for each participant with an odd participant number, the reverse of this sequence for each participant with an even participant number. The sequences can be seen in Table 1.

Table 1: Sequence of videos for participants with an even or odd participant number. The order of training videos is the same for both groups, the order of videos for experimental trials was randomized, and was reversed for the second ('even') group.

PN/Seq	TR1	TR 2	1	2	3	4	5	6	7	8	9	10
Odd	1	2	8	3	9	6	11	7	10	5	12	4
Even	1	2	4	12	5	10	7	11	6	9	3	8

Used method and materials, as described in the full-scale experiment, are very similar to the method used in this study, therefore only the differences are provided here.

- Even though the same equipment was used, no eye-tracking data was collected.
- The scenarios, programmed with PreScan 7.0, were recorded at different resolutions:
    - 1248x720 pixels for the first person view
    - 1472x720 pixels for the wing mirrors
    - 2592x720 pixels for the inner mirror
- The four separate videos were edited into one, with sound, with the use of VSDC Free Video Editor version 2.3.0.377 as well, but the resolution was 1280X720 pixels (see Appendix B3-2: Editing four videos into one using VSDC Free Video Editor).
- The Matlab program used VLC Media Player version 2.2.1 [2] ActiveX control to play the videos.
- Next to a standard Dell mouse, a standard Dell keyboard was used.
- Time from the start of a scenario until the stopping (i.e., pressing of the spacebar) of a scenario was recorded.

- Different forms were used for informed consent and instructions:
    - The informed consent form used for this experiment can be seen in Appendix B2-1: Informed consent form.
    - A separate form with instructions was used, see Appendix B2-2: Instructions.
- The layout and questionnaire within the Matlab interface differed on some points:
    - Feedback regarding the actual situation was shown before the questionnaire was presented.
    - The questionnaire contained two different questions, as is shown in Figure 1:
        1. “How mentally demanding was the task?”, which could be rated on a continuous scale between ‘Very Low’ and ‘Very High’.
        2. “I feel safe to take over control”, which could be rated on a continuous scale between ‘Completely Disagree’ and ‘Completely Agree’.

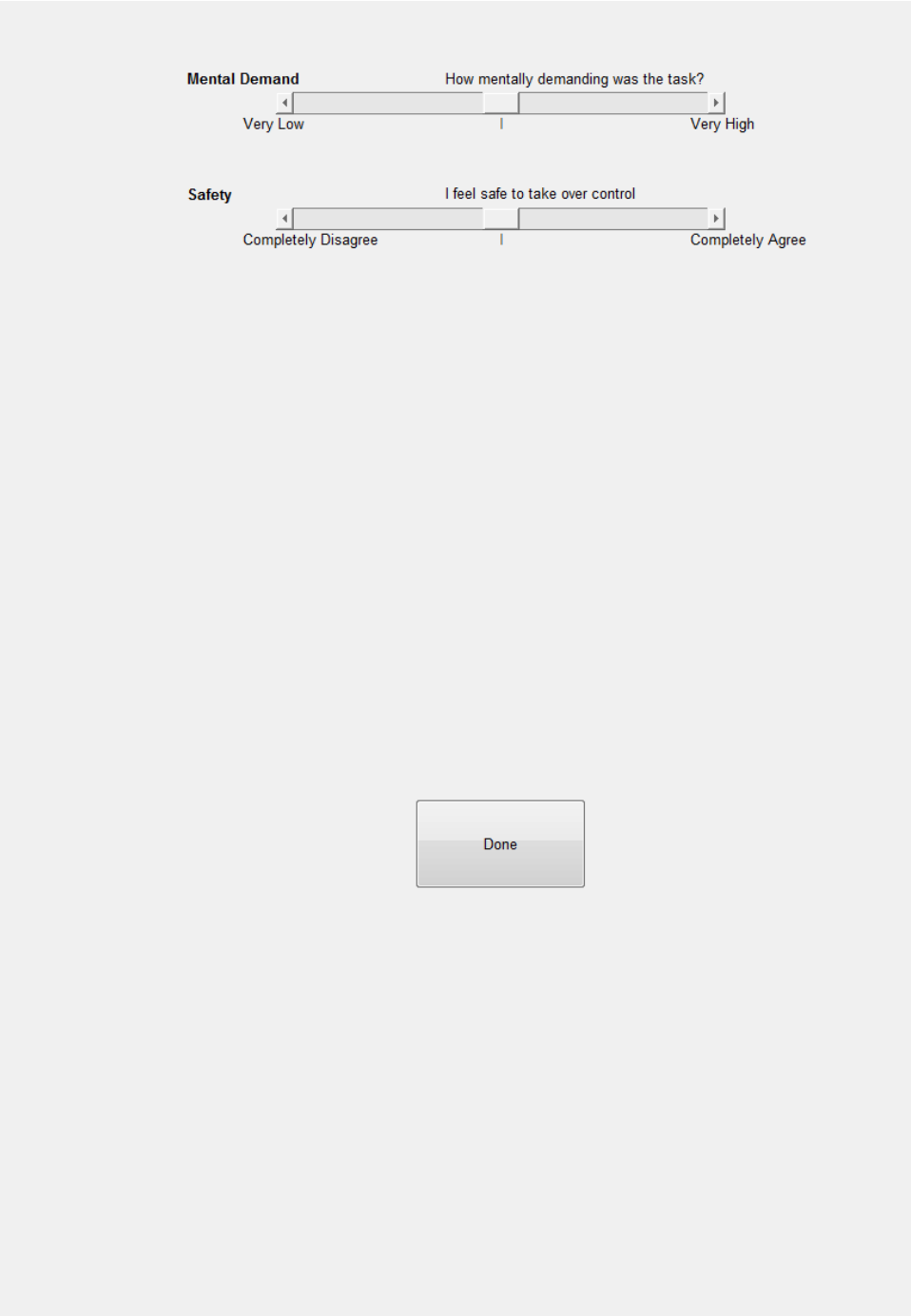


Figure 1: Questionnaire containing questions to measure subjective ‘mental demand’ and ‘safety’.

- Scenarios were built to last 30 seconds, and followed different parameters than the scenarios in the full-scale study:
    - Distances of cars could be outside the margins of the top-down view.
    - The margins of the top-down view were 150 meters ahead and 80 meters behind the ego-vehicle.
    - Cars could be out of sight for part of the scenario.
    - Cars could start or end in the blind spot.
    - Six different constant speeds were used for surrounding traffic: 26, 27, 28, 29, 30 or 31 meters/second.
    - Six different colours were used for surrounding traffic: Black, blue, green, purple, red, and yellow.
    - Some surrounding cars, and their attributes (i.e., start distance, speed, lane, and car type), were used in several scenarios.

A summary of scenario setups can be seen in Table 2, a complete overview of setups can be seen in Appendix B3-1: Complete overview of set up of scenarios.

Table 2: Setup per scenario, indicating number of cars in the scenario, and information about surrounding cars: Distribution over the lanes, start distance, and speed. Distances are ordered per lane, from left to right lane. Grey cells indicate the left and right lanes. Distances and speeds of cars that were used in several scenarios are indicated in boldface. The ego-vehicle had a starting distance of 300 meters, and a speed of 28 m/s.

Scenario Nr.	# Cars	Distri-bution	Start Distances (m) / Speeds (m/s)					
TR1	3	1-1-1	310 / 28		350 / 28		270 / 28	
TR2	3	1-1-1	260 / 28		**280 / 28**		330 / 28	
1	6	2-2-2	200 / 31	190 / 31	**320 / 28**	270 / 28	350 / 27	320 / 27
2	6	1-2-3	260 / 30	**280 / 28**	220 / 29	**290 / 30**	265 / 29	240 / 28
3	6	2-2-2	315 / 30	260 / 28	310 / 29	**260 / 28**	330 / 27	310 / 26
4	6	1-2-3	290 / 28	330 / 29	200 / 29	360 / 28	**340 / 27**	320 / 26
5	6	1-3-2	**290 / 30**	315 / 28	340 / 30	200 / 30	310 / 29	290 / 28
6	4	2-1-1	270 / 30	**250 / 29**	**320 / 28**		340 / 26	
7	4	1-2-1	**305 / 28**		320 / 30	285 / 26	**290 / 30**	
8	4	1-1-2	370 / 26		**260 / 28**		300 / 30	290 / 27
9	4	2-1-1	**305 / 28**	**250 / 29**	220 / 30		**340 / 27**	
10	4	1-1-2	**290 / 30**		360 / 28		360 / 26	250 / 29

Due to the Matlab program being unstable, only two participants were able to finish all experimental trials. From all participants, data was collected for 48 trials out of 70 possible.

Results

Space-press times of each participant, for the experimental trials, seems to have a gap in space-press times around twelve seconds, as is shown in Figure 2. The minimum time after which a participant stopped a video is 3.05 seconds, one participant did not stop the video (i.e., had a time of 30 seconds).

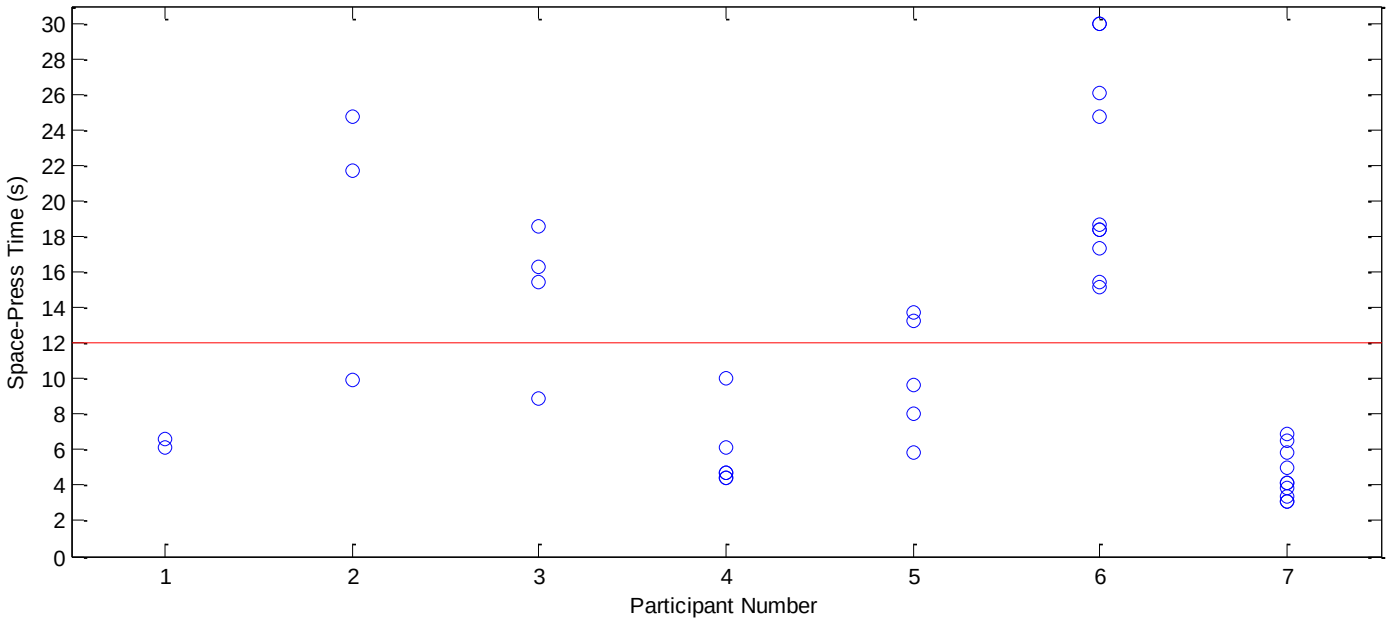


Figure 2: Space-Press time per participant for all completed experimental trials. The red line at the twelve second mark indicates a gap around which no participant stopped a video.

When looking at space-press times per scenario, the gap seems even more clear. No clear trend in space-press times for the two different traffic densities can be seen in Figure 3.

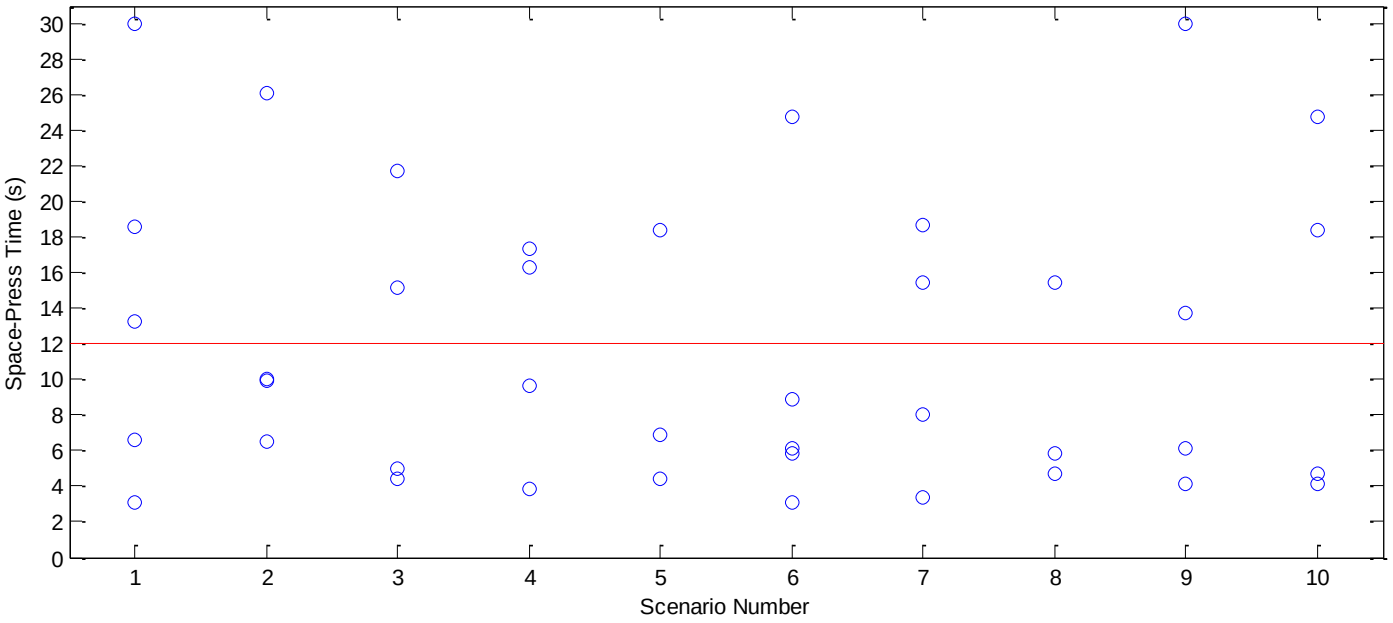


Figure 3: Space-Press time per experimental trial for all participants. The red line at the twelve second mark indicates a gap around which no participant stopped a video. The first five scenarios contain six cars of surrounding traffic, the last five scenarios

Discussion

No clear difference in space-press times for the two traffic densities was found. Although on space-press times, on average, were lower for the four car traffic density, the size of the group of participants, and the number of trials, were not sufficient for statistically testing if such a difference exists. A larger dataset is therefore needed to see if a different preparation time is needed for different traffic densities.

From post-hoc interviews with participants, it turned out participants, in some cases, tended to wait for an easy part of the scenario (e.g., two cars lining up next to each other), for easier memorization, even if they actually felt ready to stop. Participant seven indicated he did not do this for any of the trials, as shows from his data.

From the results it seems that this behaviour, where was waited for an easy part of the scenario, created a gap in stop-times, that lies around the twelve second mark. This could indicate that twelve seconds might be a suited maximum preparation time for ToCs for this type of traffic scenarios. Three seconds seems to be a minimal time for people to prepare for a ToC. Three and twelve seconds video lengths therefore seem appropriate for use in the full-scale experiment.

To be able to see if three seconds can be long enough, a shorter preparation time is also needed. According to VanRullen and Thorpe (2001) [3], the first processing stage in high-level visual categorization tasks can take as little as 75-80 msec. Taking into account that pictures were used in that study instead of videos, using a video length of one second seems appropriate.

In order to see if twelve seconds is long enough to completely prepare, a longer preparation time is needed. Although 30 seconds should surely be long enough, this video length would be so different from twelve seconds, that it would become almost impossible to draw conclusions from them. In a search for what transition times are currently used for ToCs from automated driving to manual control, it was found that Mercedes Benz uses 20 seconds in their concept truck [4], which is a time that seems to be sufficient for preparation, and still close enough to a time of twelve seconds to be able to compare performances.

Except for possible minimum and maximum preparation times, it is also important to know what is happening at times in-between. An advantage is that performance at these times can then be compared with previous findings. Looking at preparation times used in earlier research on ToCs [5 6 7], it seems wise to use video lengths of seven and nine seconds as in-between times.

References

1. L.J. Gugerty (1997), *"Situation Awareness During Driving: Explicit and Implicit Knowledge in Dynamic Spatial Memory,"* in Journal of Experimental Psychology: Applied 1997, Vol. 3, No. 1, pp. 42-66.
2. http://www.videolan.org/vlc/index.html
3. S. VanRullen, J. Thorpe (2001), *"The Time Course of Visual Processing: From Early Perception to Decision-Making"*, in Journal of Cognitive Neuroscience 13:4, pp 454-461.
4. https://www.youtube.com/watch?v=RIe9gSRBQTk
5. D. Damböck (2013), *"Automationseffekte im Fahrzeuf – von der Reaktion zur Übernahme"*
6. Gold, D. Damböck, L. Lorenz, K. Bengler (2013), *""Take over!" How long does it take to get the driver back into the loop?"*, in Proceedings of the Human Factors and Ergonomic Society 57th Annual Meeting, 2013, pp. 1938-1942
7. I. Petermann-Stock, L. Hackenberg, T. Muhr, Ch. Mergl (2013), *"Wie lange braucht der Fahrer? – Eine Analyse zu Übernahmezeiten aus verschiedenen Nebentätigkeiten während einer hochautomatisierten Staufahrt"*

Appendix B2: Forms

Appendix B2-1: Informed consent form

Research title: The time needed to (re)gain a sufficient level of Situation Awareness in order to do a safe and comfortable Transition of Control from automated to manual driving.

Researchers:
Xander Coster – MSc student
Email: x.a.coster@student.tudelft.nl

Dr.ir. Joost C.F. de Winter – Supervisor
Email: j.c.f.dewinter@tudelft.nl

Location of the experiment:
Driving simulator lab; room 34 G-0-210
Faculty of Mechanical, Maritime and Materials Engineering
Delft University of Technology
Mekelweg 2, 2628 CD Delft

Introduction: Before agreeing to participate in this study, you are asked to read this document carefully. This document describes the purpose, procedures, risks, and discomforts of this study. Your signature is required for participation. You have the right to withdraw at any time. If you desire a copy of this consent form, you may request one and we will provide it.

Purpose of the study: The purpose of this study is to investigate how much time is needed for a driver to gain knowledge of a driving situation. The results of this experiment will be published in a Master's thesis and research paper.

Duration: Your participation in this study will last approximately 30 minutes. You have the right to withdraw from the experiment at any time.

Procedures: Before the experiment starts, you will be asked to fill out a questionnaire about your past driving experiences.

Next, you will watch 12 videos of traffic scenarios on a three-lane highway, 2 of which are used as training, followed by 10 test trials. You will be requested to assess the traffic situation in each video. First, a target is displayed for five seconds, which needs to be focussed upon until the scenario starts.
When you have assessed the situation to such an extent that you feel safe to take over control of the vehicle, you press the space bar to stop the video. When the video is not stopped manually, it will automatically stop after 30 seconds.

After each video, you will be asked to fill out a questionnaire to assess your knowledge about the traffic situation, and your subjective mental demand and safety.

Risks and discomforts: There are no known risks for you in this study. You may feel uncomfortable focussing your eyes on a fixed point on the screen for a period of time.

Confidentiality: All data collected in this study will be kept confidential and will be used for research purposes only.

Right to refuse or withdraw: Your participation in this study is voluntary. You have the right to refuse or withdraw from this experiment at any time, without negative consequences.

Questions: Contact Xander Coster (contact details are included at the top) in case you have any questions or concerns about this study or your rights as a research participant.

I have read and understood the information provided above.
I give permission to process the data for the purposes described above.
I voluntarily agree to participate in this study.

Name:
.....

Signature of participant: Date:
..../ ...../ .....

Appendix B2-2: Instructions

Please read the following instructions carefully

Instructions

You will be asked to watch 12 videos of driving scenarios on a three-lane highway. Each video starts with a target being displayed for five seconds, during which a tone will be played. After hearing the tone, please focus on the displayed target until the scenario starts.

Please try to imagine that you are driving in an automatically driving vehicle, and that you are preparing for taking over control of the vehicle (i.e., start driving by yourself). Try to determine where the surrounding traffic is, and what it is doing.

When you have assessed the situation to such an extent that you feel safe to take over control of the vehicle, press the spacebar. The video will stop and you will be requested to fill out a questionnaire.

Do not try to be as fast as possible.

Appendix B3: Information about scenarios and videos

Appendix B3-1: Complete overview of set up of scenarios

The tables below give complete overviews of the setup of each scenario. Colours indicate cars that were of the same type, and had the same behaviour, in several scenarios. Scenarios are indicated with two letters and three digits. The first letter indicates the scenario number, the second whether the scenario had six surrounding cars (A) or four (B). The digits indicate how the surrounding vehicles were distributed over the three lanes.

Training1	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Ford Focus Wagon	Blue	310	28	1150
M1	BMW Z3	Red	350	28	1190
R1	Audi A8	Yellow	270	28	1110

Training2	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Ford Focus Wagon	Blue	260	28	1100
M1	BMW Z3	Red	280	28	1120
R1	Audi A8	Yellow	330	28	1170

AA_222	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Mazda RX8	Red	200	31	1130
L2	Citroen C3	Yellow	190	31	1120
M1	Audi A8	Green	320	28	1160
M2	BMW X5	Blue	270	28	1110
R1	Ford Focus Wagon	Purple	350	27	1160
R2	Toyota Yaris	Black	320	27	1130

BA_123	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Ford Focus Wagon	Green	260	30	1160
M1	BMW Z3	Yellow	280	28	1120
M2	Toyota Previa	Purple	220	29	1090
R1	Audi A8	Blue	290	30	1190
R2	BMW X5	Red	265	29	1135
R3	Citroen C3	Black	240	28	1080

CA_222	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	BMW Z3	Blue	315	30	1215
L2	Toyota Yaris	Purple	260	28	1100
M1	Fiat Bravo	Yellow	310	29	1180
M2	Citroen C3	Red	260	28	1100
R1	BMW X5	Black	330	27	1140
R2	Mazda RX8	Green	310	26	1090

DA_123	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Audi A8	Blue	290	28	1130

M1	BMW Z3	Red	330	29	1200
M2	Mazda RX8	Yellow	200	29	1070
R1	BMW X5	Green	360	28	1200
R2	Ford Fiesta	Black	340	27	1150
R3	Toyota Previa	Purple	320	26	1100

EA_132	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Toyota Yaris	Red	290	30	1190
M1	Fiat Bravo	Green	315	28	1155
M2	Ford Fiesta	Blue	340	30	1240
M3	Mazda RX8	yellow	200	30	1100
R1	Ford Focus Wagon	Black	310	29	1180
R2	BMW X5	Purple	290	28	1130

AB_211	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	BMW Z3	Yellow	270	30	1170
L2	Toyota Yaris	Blue	250	29	1120
M1	Audi A8	Green	320	28	1160
R1	Citroen C3	Red	340	26	1120

BB_121	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	BMW X5	Red	305	28	1145
M1	Toyota Previa	Yellow	320	30	1220
M2	Mazda RX8	Green	285	26	1065
R1	Audi A8	Blue	290	30	1190

CB_112	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Mazda RX8	Blue	370	26	1150
M1	Citroen C3	Red	260	28	1100
R1	BMW Z3	Green	300	30	1200
R2	Toyota Previa	Black	290	27	1100

DB_211	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	BMW X5	Red	305	28	1145
L2	Toyota Yaris	Blue	250	29	1120
M1	Fiat Bravo	Purple	220	30	1120
R1	Ford Fiesta	Black	340	27	1150

EB_112	Type	Colour	Start30	Speed (m/s)	End:
SELF	Fiat Bravo	Grey	300	28	1140
L1	Toyota Yaris	Red	290	30	1190
M1	BMW Z3	Black	360	28	1200
R1	Audi A8	Purple	360	26	1140
R2	Ford Focus Wagon	Green	250	29	1120

Appendix B3-2: Editing four videos into one using VSDC Free Video Editor

The method of editing the four videos into one is the same as described in

Appendix A2-4: Editing four videos into one using VSDC Free Video Editor. The main differences are the resolutions of the videos that were recorded with PreScan, and the lower resolution of the resulting video. These differences have influences on the placement in the frame, when expressed in pixels. A summary of resolutions, placement, and start and end times can be seen in Table B3-1.

Table B3-1: Information on all elements in a final video, containing resolutions (original and final), placement, and start and end times.

Name	Original Resolution (pixels)	Final Resolution (pixels)	Placement in frame ([left, top], pixels)	Start Time (s)	End Time (s)
Target	1920x1080	1280x720	[0, 0]	0	5
POV View	1248x720	1280x720	[0, 0]	5	5+Length
Inner Mirror	2592x720	470x150	[717.5, 51.25]	5	5+Length
Left Wing Mirror	1472x720	287.5x138.75	[15, 493.75]	5	5+Length
Right Wing Mirror	1472x720	287.5x138.75	[977.5, 493.75]	5	5+Length
'Inner' Rectangle	dna	480x137.5	[712.5, 56.25]	5	5+Length
'Left' Rectangle	dna	291.25x151.25	[15, 487.75]	5	5+Length
'Right' Rectangle	dna	291.25x151.25	[973.75, 487.75]	5	5+Length
End Screen	dna	1280x720	[0, 0]	5+Length	5+Length+0.5
Interior sound	dna	dna	dna	0	5+Length
Beep	dna	dna	dna	2	3.52

Appendix B5: Matlab code (SmallGUI.m)

```
%Made By Xander Coster
%Student nr. 1268546

%in case of Matlab stall/crash: turn on or change following lines before restart
% 188, 193 and 196 (DeletedCars, result and vidcount) for click
% 253, 258 and 261 (DeletedCars, result and vidcount) for keypress

function varargout = SmallGUI(varargin)
% SMALLGUI MATLAB code for SmallGUI.fig
%      SMALLGUI, by itself, creates a new SMALLGUI or raises the existing
%      singleton*.
%
%      H = SMALLGUI returns the handle to a new SMALLGUI or the handle to
%      the existing singleton*.
%
%      SMALLGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SMALLGUI.M with the given input arguments.
%
%      SMALLGUI('Property','Value',...) creates a new SMALLGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SmallGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SmallGUI_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 SmallGUI

% Last Modified by GUIDE v2.5 16-Apr-2015 16:28:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SmallGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @SmallGUI_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 SmallGUI is made visible.
function SmallGUI_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 SmallGUI (see VARARGIN)
```

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

% Update handles structure
guidata(hObject, handles);
uicontrol(handles.Info1)
% UIWAIT makes SmallGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SmallGUI_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;
set(hObject, 'units','pixels','outerposition',[0 1080 1920 1200]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% First page / Panel1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --------------------------------------------------------------------
function Panel1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to Panel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Info1_Callback(hObject, eventdata, handles)
% hObject    handle to Info1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Info1 as text
%        str2double(get(hObject,'String')) returns contents of Info1 as a double

PN = str2double(get(hObject,'string'));     %get participant nr from input
assignin('base', 'PN', PN)                  %save participant nr
uicontrol(handles.Info2)

% --- Executes during object creation, after setting all properties.
function Info1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Info1 (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 Info2_Callback(hObject, eventdata, handles)
% hObject    handle to Info2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Info2 as text
%        str2double(get(hObject,'String')) returns contents of Info2 as a double
Age = str2double(get(hObject,'string'));        %get age from input
```

```
assignin('base', 'Age', Age)                    %save age
set(handles.PBSave,'visible','on')
uicontrol(handles.PBSave)

% --- Executes during object creation, after setting all properties.
function Info2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Info2 (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 PBSave.
function PBSave_Callback(hObject, eventdata, handles)
% hObject    handle to PBSave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

PN = evalin('base','PN');
Age = evalin('base','Age');

if mod(PN,2)
    %PN is odd
    Groupnr = 1;
    seq = [1 2 8 3 9 6 11 7 10 5 12 4];
    assignin('base', 'Groupnr', Groupnr)
    assignin('base','seq',seq)
else
    %PN is even
    Groupnr = 2;
    seq = [1 2 4 12 5 10 7 11 6 9 3 8];
    assignin('base', 'Groupnr', Groupnr)
    assignin('base','seq',seq)
end
    load('AnswersSmall.mat');
    assignin('base','Answers',Answers);
    DeletedCars = zeros(12,8);
    DelFile = ['DeletedCars' num2str(PN) '.mat'];
%%%     load((DelFile)); % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%
    assignin('base','DeletedCars',DeletedCars);
    filename = ['Resultsp' int2str(PN) '.mat'];
    assignin('base','filename',filename);
    result = zeros(12,35); %create empty result matrix
%%%    load((filename)); % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%
    result(1,1:3)=[PN Age Groupnr];
    assignin('base','result',result);
    vidcount = 1;       % Change this after Matlab stall/crach%%%%%%%%%%%%%%%
    assignin('base', 'vidcount', vidcount)

    set(handles.Panel1,'visible','off')
    set(handles.PBStart,'visible','on')

    uicontrol(handles.PBStart)

% --- Executes on key press with focus on PBSave and none of its controls.
function PBSave_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to PBSave (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
```

```
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
a = eventdata.Key;

if strcmp(a,'return') == 1
    PN = evalin('base','PN');
    Age = evalin('base','Age');

    if mod(PN,2)
    %PN is odd
    Groupnr = 1;
    seq = [1 2 8 3 9 6 11 7 10 5 12 4];
    assignin('base', 'Groupnr', Groupnr)
    assignin('base','seq',seq)
else
    %PN is even
    Groupnr = 2;
    seq = [1 2 4 12 5 10 7 11 6 9 3 8];
    assignin('base', 'Groupnr', Groupnr)
    assignin('base','seq',seq)
end
    load('AnswersSmall.mat');
    assignin('base','Answers',Answers);

    DeletedCars = zeros(12,8);
    DelFile = ['DeletedCars' num2str(PN) '.mat'];
%%%     load((DelFile)); % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%%
    assignin('base','DeletedCars',DeletedCars);
    filename = ['Resultsp' int2str(PN) '.mat'];
    assignin('base','filename',filename);
    result = zeros(12,35); %create empty result matrix
%%%     load((filename)) % turn this on after Matlab stall/crash%%%%%%%%%%%%%%%
    result(1,1:3)=[PN Age Groupnr];
    assignin('base','result',result);
    vidcount = 1;       % Change this after Matlab stall/crach%%%%%%%%%%%%%%%%
    assignin('base', 'vidcount', vidcount)

    set(handles.Panel1,'visible','off')
    set(handles.PBStart,'visible','on')

    uicontrol(handles.PBStart)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Video display %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in PBStart.
function PBStart_Callback(hObject, eventdata, handles)
% hObject    handle to PBStart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f=figure('units','pixels','outerposition',[0 1040 1920 1280],'Toolbar','none',...
            'Menubar','none','numbertitle','off','Tool','none','Tag','KeyCatch');
vlc=actxcontrol('VideoLAN.VlcPlugin.2',[0 0 1920 1280],f);
Answers = evalin('base','Answers');
vidcount = evalin('base','vidcount');

seq = evalin('base','seq');
result = evalin('base','result');
filepath = 'C:\Users\Coster\Dropbox\Afstuderen\Research\Videos\Small Test\Video';
filepath(filepath=='\') = '/';
filepath = ['file://localhost/' filepath int2str(seq(vidcount)) '.avi'];
```

```
vlc.playlist.add(filepath);
vlc.video.fullscreen();
vlc.playlist.play();
vlc.video.deinterlace.enable('x');

pause(5)
while vlc.playlist.isPlaying == 1
    Key = ' ';
    hf = findall(0,'Tag','KeyCatch');
    figure(hf)
    cc = get(hf,'CurrentCharacter');

    if strcmpi(cc,Key)
%         vlc.playlist.pause()
        t = vlc.input.time();
        pause(0.15)
        vlc.playlist.stop()

%         break
    else
                t = Answers(seq(vidcount),2)*1000;
    end
end
    telapsed = t/1000-5;
    assignin('base','telapsed',telapsed);
    result(vidcount,[4,7]) = [seq(vidcount) telapsed];
    assignin('base','result',result);
close Figure 1

set(handles.PBStart,'visible','off')
set(handles.PosSlider,'Value', 0,'visible','on');
set(handles.LaneSlider,'Value',0,'visible','on');
set(handles.text3,'visible','on');
set(handles.text4,'visible','on');
set(handles.TxtSpeedInstr,'visible','on');
set(handles.text68,'visible','on');
set(handles.text69,'visible','on');
set(handles.text70,'visible','on');

set(handles.SAPanel,'visible','on')
set(handles.PBNewCar,'visible','on')
set(handles.PBDeleteCar,'visible','off')
set(handles.PBResetCar,'visible','off')
set(handles.AxAnswer,'visible','off')

axes(handles.AxTopview)
Tv = imread('Topview_2.PNG');
imshow(Tv,'parent',handles.AxTopview);
uistack(handles.AxTopview,'bottom')
hold(handles.AxTopview);

CarCnt = 0;
CarDel = 0;
assignin('base','CarCnt',CarCnt);
assignin('base','CarDel',CarDel);

% --- Executes on key press with focus on PBStart and none of its controls.
function PBStart_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to PBStart (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
```

```
b = eventdata.Key;
if strcmp(b,'return') == 1

f=figure('units','pixels','outerposition',[0 1040 1920 1280],'Toolbar','none',...
            'Menubar','none','numbertitle','off','Tool','none','Tag','KeyCatch');
vlc=actxcontrol('VideoLAN.VlcPlugin.2',[0 0 1920 1280],f);
Answers = evalin('base','Answers');
vidcount = evalin('base','vidcount');
seq = evalin('base','seq');
result = evalin('base','result');
filepath = 'C:\Users\Coster\Dropbox\Afstuderen\Research\Videos\Small Test\Video';
filepath(filepath=='\') = '/';
filepath = ['file://localhost/' filepath int2str(seq(vidcount)) '.avi'];
vlc.playlist.add(filepath);
vlc.video.fullscreen();
vlc.playlist.play();
vlc.video.deinterlace.enable('x');

set(f,'ButtonDownFcn',@mybttnfcn)
while vlc.playlist.isPlaying == 1
        Key = ' ';
        hf = findall(0,'Tag','KeyCatch');

        figure(hf)

        cc = get(hf,'CurrentCharacter');
        mb = get(hf,'Selectiontype');
        if strcmpi(mb,'normal')~= 1;
            disp('mouse works')
        end
    if strcmpi(cc,Key)
        vlc.playlist.pause()
        t = vlc.input.time();
        pause(0.15)
        vlc.playlist.stop()
%         break
    else
%         t = 35000;
        t = Answers(seq(vidcount),2)*1000;
    end
end

telapsed = t/1000-5;
assignin('base','telapsed',telapsed);
result(vidcount,[4,7]) = [seq(vidcount) telapsed];
assignin('base','result',result);
close Figure 1

set(handles.PBStart,'visible','off')
set(handles.PosSlider,'Value', 0,'visible','on');
set(handles.LaneSlider,'Value',0,'visible','on');
set(handles.text3,'visible','on');
set(handles.text4,'visible','on');
set(handles.SAPanel,'visible','on')
set(handles.PBNewCar,'visible','on')
set(handles.PBDeleteCar,'visible','off')
set(handles.PBResetCar,'visible','off')
set(handles.AxAnswer,'visible','off')
set(handles.TxtSpeedInstr,'visible','on');
set(handles.text68,'visible','on');
set(handles.text69,'visible','on');
set(handles.text70,'visible','on');

axes(handles.AxTopview)
```

```
Tv = imread('Topview_2.PNG');
imshow(Tv,'parent',handles.AxTopview);
uistack(handles.AxTopview,'bottom')
hold(handles.AxTopview);

CarCnt = 0;
CarDel = 0;
assignin('base','CarCnt',CarCnt);
assignin('base','CarDel',CarDel);
end

function mybttnfcn(hObject, eventdata, handles)
hf = get(hObject,'parent');
b = get(hf,'selectiontype');
if strcmpi(b,'Normal');
    disp('Left click')
elseif strcmpi(b,'alt')
    disp('Right click')
    assignin('base','mb',b)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SAPanel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in PBNewCar.
function PBNewCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBNewCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
CarCnt = evalin('base','CarCnt');
CarCnt = CarCnt+1;
assignin('base','CarCnt',CarCnt);
if CarCnt >= 9
    h = msgbox({'Maximum number' 'of cars' 'reached'}, 'Error','error');
    set(h,'units','normalized','outerposition',[0.3 1.5 .1 .12]);
    CarCnt = 8;
    assignin('base','CarCnt',CarCnt);
end
set(handles.LaneSlider,'Value',0);
set(handles.PosSlider,'Value', 0);

CurrentAx = ['AxCar' num2str(CarCnt)];
CurrentTxt = ['TxtCar' num2str(CarCnt)];
RBSlower = ['RB' num2str(CarCnt) 'Slower'];
RBSame = ['RB' num2str(CarCnt) 'Same'];
RBFaster =['RB' num2str(CarCnt) 'Faster'];

set(handles.(CurrentTxt),'visible','on');
set(handles.(RBSlower),'visible','on');
set(handles.(RBSame),'visible','on');
set(handles.(RBFaster),'visible','on');
set(handles.PBDeleteCar,'visible','on');

vidcount = evalin('base','vidcount');
% if vidcount <= 5
    set(handles.PBDoneSA,'visible','on');
    set(handles.PBFinish,'visible','off');
% else
assignin('base','CurrentAx',CurrentAx);

CarIm = imread('Car2.PNG');
CarFig = imshow(CarIm,'parent',handles.(CurrentAx));

set(CarFig,'Tag',(CurrentAx));
```

```
set(CarFig,'ButtonDownFcn',@ImageClickCallback);

set(handles.(CurrentAx),'position',[2.9 13.77 0.255 0.42]);

uistack(handles.(CurrentAx),'top')
text(-
60,0,num2str(CarCnt),'backgroundcolor','none','parent',handles.(CurrentAx),'color','b')

function ImageClickCallback (hObject, eventdata, handles)
axesHandle = get(hObject,'Parent');
Pos = get(axesHandle,'Position');
handles = guihandles(SmallGUI);
Tag = get(hObject,'Tag');
assignin('base','CurrentAx',Tag);
set(handles.PosSlider,'Value',Pos(2)-13.77);
set(handles.LaneSlider,'Value',Pos(1)-2.9);

% --- Executes on button press in PBDeleteCar.
function PBDeleteCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBDeleteCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
CurrentAx = evalin('base','CurrentAx');
CarNum = str2double(CurrentAx(6));
CurrentTxt = ['TxtCar' num2str(CarNum)];
CurrentRBSlow = ['RB' num2str(CarNum) 'Slower'];
CurrentRBSame = ['RB' num2str(CarNum) 'Same'];
CurrentRBFast = ['RB' num2str(CarNum) 'Faster'];
set(handles.(CurrentTxt),'visible','off');
set(handles.(CurrentRBSlow),'visible','off','Value',0);
set(handles.(CurrentRBSame),'visible','off','Value',0);
set(handles.(CurrentRBFast),'visible','off','Value',0);

RelVel = 5;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,CarNum*3+7) = RelVel;
assignin('base','result',result);

set(handles.PBNewCar,'visible','off');
set(handles.PBResetCar,'visible','on');
CarDel = evalin('base','CarDel');
CarDel = CarDel + 1;
assignin('base','CarDel',CarDel);
DeletedCars = evalin('base','DeletedCars');
vidcount = evalin('base','vidcount');
DeletedCars(vidcount, CarDel) = CarNum;
assignin('base','DeletedCars',DeletedCars);
set(handles.(CurrentAx),'position',[8+0.5*str2double(CurrentAx(6)) 30 0.255 0.42]);

% --- Executes on button press in PBResetCar.
function PBResetCar_Callback(hObject, eventdata, handles)
% hObject    handle to PBResetCar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
DeletedCars = evalin('base','DeletedCars');
vidcount = evalin('base','vidcount');
CarDel = evalin('base','CarDel');
CarNum = DeletedCars(vidcount,CarDel);
Ax = ['AxCar' num2str(CarNum)];
set(handles.(Ax),'position',[2.9 13.77 0.255 0.42]);
CurrentRBSlow = ['RB' num2str(CarNum) 'Slower'];
CurrentRBSame = ['RB' num2str(CarNum) 'Same'];
CurrentRBFast = ['RB' num2str(CarNum) 'Faster'];
```

```
set(handles.(CurrentRBSlow),'visible','on');
set(handles.(CurrentRBSame),'visible','on');
set(handles.(CurrentRBFast),'visible','on');
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
DeletedCars(vidcount,CarDel) = 0;
assignin('base','DeletedCars',DeletedCars);
CarDel = CarDel - 1;
assignin('base','CarDel',CarDel);
set(handles.PosSlider,'Value',0);
set(handles.LaneSlider,'Value',0);
assignin('base','CurrentAx',Ax);
if CarDel == 0
    set(handles.PBNewCar,'visible','on');
    set(hObject,'visible','off');
end

%% Sliders
% --- Executes on slider movement.
function PosSlider_Callback(hObject, eventdata, handles)
% hObject    handle to PosSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
CurrentAx = evalin('base','CurrentAx');
CarNum = CurrentAx(6);
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
hight = get(hObject,'Value');
lane = get(handles.LaneSlider,'Value');

set(handles.(CurrentAx),'position',[(2.9+lane) (13.77+hight) 0.255 0.42]);


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

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function LaneSlider_Callback(hObject, eventdata, handles)
% hObject    handle to LaneSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
CurrentAx = evalin('base','CurrentAx');
CarNum = CurrentAx(6);
CurrentTxt = ['TxtCar' num2str(CarNum)];
set(handles.(CurrentTxt),'visible','on');
hight = get(handles.PosSlider,'Value');
lane = get(hObject,'Value');
if lane >= 0.175
    set(hObject,'Value',0.35);
    lane = 0.35;
```

```
elseif lane <= -0.175
    set(hObject,'Value',-0.35);
    lane = -0.35;
else
    set(hObject,'Value',0);
    lane = 0;
end
set(handles.(CurrentAx),'position',[(2.9+lane) (13.77+hight) 0.255 0.42]);

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

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

%% -----Radiobuttons

% --- Executes on button press in RB1Slower.
function RB1Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Slower (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.RB1Same,'Value',0);
    set(handles.RB1Faster,'Value',0);
    RelVel1 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,10) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Slower

% --- Executes on button press in RB1Same.
function RB1Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Same (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.RB1Slower,'Value',0);
    set(handles.RB1Faster,'Value',0);
    RelVel1 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,10) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Same

% --- Executes on button press in RB1Faster.
function RB1Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB1Faster (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.RB1Slower,'Value',0);
    set(handles.RB1Same,'Value',0);
    RelVel1 = 1;
result = evalin('base','result');
```

```
vidcount = evalin('base','vidcount');
result(vidcount,10) = RelVel1;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB1Faster


% --- Executes on button press in RB2Slower.
function RB2Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Slower (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.RB2Same,'Value',0);
    set(handles.RB2Faster,'Value',0);
    RelVel2 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,13) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Slower

% --- Executes on button press in RB2Same.
function RB2Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Same (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.RB2Slower,'Value',0);
    set(handles.RB2Faster,'Value',0);
    RelVel2 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,13) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Same

% --- Executes on button press in RB2Faster.
function RB2Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB2Faster (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.RB2Slower,'Value',0);
    set(handles.RB2Same,'Value',0);
    RelVel2 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,13) = RelVel2;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB2Faster


% --- Executes on button press in RB3Slower.
function RB3Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Slower (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.RB3Same,'Value',0);
    set(handles.RB3Faster,'Value',0);
```

```
    RelVel3 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,16) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Slower

% --- Executes on button press in RB3Same.
function RB3Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Same (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.RB3Slower,'Value',0);
    set(handles.RB3Faster,'Value',0);
    RelVel3 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,16) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Same

% --- Executes on button press in RB3Faster.
function RB3Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB3Faster (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.RB3Slower,'Value',0);
    set(handles.RB3Same,'Value',0);
    RelVel3 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,16) = RelVel3;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB3Faster


% --- Executes on button press in RB4Slower.
function RB4Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Slower (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.RB4Same,'Value',0);
    set(handles.RB4Faster,'Value',0);
    RelVel4 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,19) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Slower

% --- Executes on button press in RB4Same.
function RB4Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Same (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.RB4Slower,'Value',0);
```

```
    set(handles.RB4Faster,'Value',0);
    RelVel4 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,19) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Same

% --- Executes on button press in RB4Faster.
function RB4Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB4Faster (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.RB4Slower,'Value',0);
    set(handles.RB4Same,'Value',0);
    RelVel4 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,19) = RelVel4;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB4Faster

% --- Executes on button press in RB5Slower.
function RB5Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Slower (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.RB5Same,'Value',0);
    set(handles.RB5Faster,'Value',0);
    RelVel5 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,22) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Slower

% --- Executes on button press in RB5Same.
function RB5Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Same (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.RB5Slower,'Value',0);
    set(handles.RB5Faster,'Value',0);
    RelVel5 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,22) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Same

% --- Executes on button press in RB5Faster.
function RB5Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB5Faster (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.RB5Slower,'Value',0);
    set(handles.RB5Same,'Value',0);
```

```
    RelVel5 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,22) = RelVel5;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB5Faster


% --- Executes on button press in RB6Slower.
function RB6Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Slower (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.RB6Same,'Value',0);
    set(handles.RB6Faster,'Value',0);
    RelVel6 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,25) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Slower

% --- Executes on button press in RB6Same.
function RB6Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Same (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.RB6Slower,'Value',0);
    set(handles.RB6Faster,'Value',0);
    RelVel6 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,25) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Same

% --- Executes on button press in RB6Faster.
function RB6Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB6Faster (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.RB6Slower,'Value',0);
    set(handles.RB6Same,'Value',0);
    RelVel6 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,25) = RelVel6;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB6Faster


% --- Executes on button press in RB7Slower.
function RB7Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Slower (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.RB7Same,'Value',0);
    set(handles.RB7Faster,'Value',0);
    RelVel7 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,28) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Slower

% --- Executes on button press in RB7Same.
function RB7Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Same (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.RB7Slower,'Value',0);
    set(handles.RB7Faster,'Value',0);
    RelVel7 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,28) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Same

% --- Executes on button press in RB7Faster.
function RB7Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB7Faster (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.RB7Slower,'Value',0);
    set(handles.RB7Same,'Value',0);
    RelVel7 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,28) = RelVel7;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB7Faster


% --- Executes on button press in RB8Slower.
function RB8Slower_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Slower (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.RB8Same,'Value',0);
    set(handles.RB8Faster,'Value',0);
    RelVel8 = -1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,31) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Slower


% --- Executes on button press in RB8Same.
function RB8Same_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Same (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.RB8Slower,'Value',0);
    set(handles.RB8Faster,'Value',0);
    RelVel8 = 0;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,31) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Same

% --- Executes on button press in RB8Faster.
function RB8Faster_Callback(hObject, eventdata, handles)
% hObject    handle to RB8Faster (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.RB8Slower,'Value',0);
    set(handles.RB8Same,'Value',0);
    RelVel8 = 1;
result = evalin('base','result');
vidcount = evalin('base','vidcount');
result(vidcount,31) = RelVel8;
assignin('base','result',result);
end
% Hint: get(hObject,'Value') returns toggle state of RB8Faster

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

% if nothing is filled in or if filled in incomplete, give warning message
% and get out
vidcount = evalin('base','vidcount');
result = evalin('base','result');
CarCnt = evalin('base','CarCnt');
CarDel = evalin('base','CarDel');
Answers = evalin('base','Answers');
% Answers = load('AnswersSmall.mat','Answers');
telapsed = evalin('base','telapsed');
PosSelf = 300 + 28*telapsed;
seq = evalin('base','seq');
Answer = Answers(seq(vidcount),:);
NumOCars = Answer(1);
assignin('base','NumOCars',NumOCars);
if CarCnt == 0 || CarCnt-CarDel == 0
    h = msgbox({'No cars were placed'}, 'Error','error');
    set(h,'units','normalized','outerposition',[0.45 1.6 .1 .12]);
    return
end

for i = 1:CarCnt
    CurrentTxt = ['TxtCar' num2str(i)];
    CurrentRBSlow = ['RB' num2str(i) 'Slower'];
    CurrentRBSame = ['RB' num2str(i) 'Same'];
    CurrentRBFast = ['RB' num2str(i) 'Faster'];
    if strcmp(get(handles.(CurrentTxt),'visible'),'on') && ...
            get(handles.(CurrentRBSlow),'Value') == 0 && ...
            get(handles.(CurrentRBSame),'Value') == 0 && ...
            get(handles.(CurrentRBFast),'Value') == 0
```

```
        h = msgbox({'Please indicate' 'the relative speed' ['of car' num2str(i)]},
'Error','error');
        set(h,'units','normalized','outerposition',[0.45 1.6 .1 .12]);
        return
    end
end
for i = 1:CarCnt
    CurrentTxt = ['TxtCar' num2str(i)];
    CurrentRBSlow = ['RB' num2str(i) 'Slower'];
    CurrentRBSame = ['RB' num2str(i) 'Same'];
    CurrentRBFast = ['RB' num2str(i) 'Faster'];
    RelVel = result(vidcount,i*3+7);
    CurrentAx = ['AxCar' num2str(i)];
    CarIm = imread('Car2.PNG');
    CarFig = imshow(CarIm,'parent',handles.(CurrentAx));
%     axes(handles.(CurrentAx));
    if RelVel == -1
    text(-80,0,{'<'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
    elseif RelVel == 0
    text(-80,0,{'='
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
    elseif RelVel == 1
        text(-80,0,{'>'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
    end
    set(handles.(CurrentTxt),'visible','off');
    set(handles.TxtSpeedInstr,'visible','off');
    set(handles.text68,'visible','off');
    set(handles.text69,'visible','off');
    set(handles.text70,'visible','off');
    set(handles.(CurrentRBSlow),'visible','off','value',0);
    set(handles.(CurrentRBSame),'visible','off','value',0);
    set(handles.(CurrentRBFast),'visible','off','value',0);
end
% Show answer
    set(handles.AxAnswer,'visible','on');
    axes(handles.AxAnswer);
    Tv = imread('Topview_2.PNG');
    imshow(Tv,'parent',handles.AxAnswer);
    uistack(handles.AxAnswer,'bottom')
    hold(handles.AxAnswer);

        for i = 1:NumOCars
            CurrentAx = ['AxAns' num2str(i)];
            CarIm = imread('Car2.PNG');
            CarFig = imshow(CarIm,'parent',handles.(CurrentAx));
            Lanepos = (Answer(3*i) + Answer(3*i+2)*telapsed - PosSelf)/10;
            Lane = Answer(3*i+1)*0.35;
            set(handles.(CurrentAx),'position',[(17.95+Lane) Lanepos+13.75 0.255
0.42]);
            if Answer(3*i+2) > 28
                text(-80,0,{'>'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
            elseif Answer (3*i+2) == 28
                text(-80,0,{'='
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
            elseif Answer (3*i+2) < 28
```

```
                text(-80,0,{'<'
num2str(i)},'backgroundcolor','none','parent',handles.(CurrentAx),'color','b','FontSize
',14)
            end
        end
    set(hObject,'visible','off');
    set(handles.PBNewCar,'visible','off');
    set(handles.PBDeleteCar,'visible','off');
    set(handles.PBResetCar,'visible','off');
    set(handles.PosSlider,'visible','off');
    set(handles.LaneSlider,'visible','off');
    set(handles.text3,'visible','off');
    set(handles.text4,'visible','off');
    set(handles.TxtYourAnswer,'visible','on');
    set(handles.TxtActual,'visible','on');
    set(handles.PBFinish,'visible','on');
    uicontrol(handles.PBFinish)

% --- Executes on button press in PBFinish.
function PBFinish_Callback(hObject, eventdata, handles)
% hObject    handle to PBFinish (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
result = evalin('base','result');
vidcount = evalin('base','vidcount');
NumOCars = evalin('base','NumOCars');
CarCnt = evalin('base','CarCnt');
CarDel = evalin('base','CarDel');
NumP = CarCnt - CarDel; %number of placed cars - deleted cars
allNumP = CarCnt; %just placed cars
result = evalin('base','result');
result(vidcount,5) = allNumP;
result(vidcount,6) = NumP;
assignin('base','result',result);

for j = 1:8
    AX = ['AxCar' num2str(j)];
    CurrentTxt = ['TxtCar' num2str(j)];
    set(handles.(CurrentTxt),'visible','off');
    CurrentRBSlow = ['RB' num2str(j) 'Slower'];
    CurrentRBSame = ['RB' num2str(j) 'Same'];
    CurrentRBFast = ['RB' num2str(j) 'Faster'];
    set(handles.AxAnswer,'visible','off');

    Pos = get(handles.(AX),'position');
    Pos(1) = Pos(1)-2.9;
    if Pos(1) == -0.35
        Pos(1) = -1;
    elseif Pos(1) == 0.35
        Pos(1) = 1;
    end

    Pos(2) = Pos(2)-13.75;

    result(vidcount,[j*3+5 j*3+6]) = [Pos(1) Pos(2)];
    assignin('base','result',result)
    set(handles.(AX),'position',[8+0.5*j 30 0.252 0.42]);
    axes(handles.(AX));
    cla reset
    set(handles.(AX),'visible','off');
end

for k = 1:NumOCars
    CurrentAx = ['AxAns' num2str(k)];
```

```
    set(handles.(CurrentAx),'position',[12+0.5*k 30 0.252 0.42]);
    axes(handles.(CurrentAx));
    cla reset
    set(handles.(CurrentAx),'visible','off');
end
    set(handles.SAPanel,'visible','off');
    axes(handles.AxAnswer);
    cla reset
    set(handles.AxAnswer,'visible','off');
    set(handles.MentSlid,'Value',50);
    set(handles.SafeSlid,'Value',50);
    set(handles.TLXPanel,'visible','on');
    set(handles.TxtYourAnswer,'visible','off');
    set(handles.TxtActual,'visible','off');
    set(hObject,'visible','off');

% --- Executes on key press with focus on PBFinish and none of its controls.
function PBFinish_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to PBFinish (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
c = eventdata.Key;
if strcmp(c,'return') == 1

vidcount = evalin('base','vidcount');
CarCnt = evalin('base','CarCnt');
for j = 1:8
    AX = ['AxCar' num2str(j)];
    CurrentTxt = ['TxtCar' num2str(j)];
    CurrentRBSlow = ['RB' num2str(j) 'Slower'];
    CurrentRBSame = ['RB' num2str(j) 'Same'];
    CurrentRBFast = ['RB' num2str(j) 'Faster'];
    set(handles.(CurrentTxt),'visible','off');
    set(handles.(CurrentRBSlow),'visible','off');
    set(handles.(CurrentRBSame),'visible','off');
    set(handles.(CurrentRBFast),'visible','off');
    Pos = get(handles.(AX),'position');
    Pos(1) = Pos(1)-2.9;
    Pos(2) = Pos(2)-10.1;
    result = evalin('base','result');
    result(vidcount,[j*3+3 j*3+4]) = [Pos(1) Pos(2)];
    assignin('base','result',result)
    set(handles.(AX),'position',[8+0.5*j 30 0.252 0.42]);
    axes(handles.(AX));
    cla reset
    set(handles.(AX),'visible','off');
end

    for k = 1:NumOCars
    CurrentAx = ['AxAns' num2str(k)];
    set(handles.(CurrentAx),'position',[12+0.5*k 30 0.252 0.42]);
    axes(handles.(CurrentAx));
    cla reset
    set(handles.(CurrentAx),'visible','off');
    end
    set(handles.SAPanel,'visible','off');
    set(handles.AxAnswer,'visible','off');

    set(handles.MentSlid,'Value',50);
    set(handles.SafeSlid,'Value',50);
    set(handles.TLXPanel,'visible','on');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TLX Panel %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on slider movement.
function MentSlid_Callback(hObject, eventdata, handles)
% hObject    handle to MentSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Mental = get(hObject,'Value');
vidcount = evalin('base','vidcount');
result = evalin('base','result');
result(vidcount,32) = Mental;
assignin('base','result',result);

% --- Executes during object creation, after setting all properties.
function MentSlid_CreateFcn(hObject, eventdata, ~)
% hObject    handle to MentSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function TimeSlid_Callback(hObject, eventdata, handles)
% hObject    handle to TimeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Temporal = get(hObject,'Value');
Temporal = roundn(Temporal,-1);
set(hObject,'Value',Temporal)
vidcount = evalin('base','vidcount');
result = evalin('base','result');
result(vidcount,33) = Temporal;
assignin('base','result',result);
% --- Executes during object creation, after setting all properties.
function TimeSlid_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TimeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function SafeSlid_Callback(hObject, eventdata, handles)
% hObject    handle to SafeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
Safety = get(hObject,'Value');
```

```
Safety = roundn(Safety,-1);
set(hObject,'Value',Safety)
vidcount = evalin('base','vidcount');
result = evalin('base','result');
result(vidcount,34) = Safety;
assignin('base','result',result);
% --- Executes during object creation, after setting all properties.
function SafeSlid_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SafeSlid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
set(hObject,'Value',50);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in PBDoneTLX.
function PBDoneTLX_Callback(hObject, eventdata, handles)
% hObject    handle to PBDoneTLX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.TLXPanel,'visible','off');
vidcount=evalin('base','vidcount');
vidcount = vidcount+1;
assignin('base','vidcount',vidcount);
if vidcount == 2
    set(handles.PBStart,'String','Start Training 2');
else
    Num = vidcount - 2;
    set(handles.PBStart,'String',['Start Video ' num2str(Num)]);
end

    filename = evalin('base','filename');
    result = evalin('base','result');
    save((filename),'result');
    PN = evalin('base','PN');
    save('PN.mat','PN')
    DeletedCars = evalin('base','DeletedCars');
    DelFile = ['DeletedCars' num2str(PN) '.mat'];
    save(DelFile,'DeletedCars');
    if vidcount >= 13
    set(handles.text71,'visible','on');
    pause(5)
    close all
else
    set(handles.PBStart,'visible','on');
    uicontrol(handles.PBStart)
end
```

Appendix C: Literature Report

Literature Review

The time needed to (re)gain a sufficient level of Situation Awareness in order to do a safe and comfortable Transition of Control from automated to manual driving.

26-1-2015

Xander Coster
1268546

DAVI project

Picture taken from: http://wardsauto.com/site-files/wardsaut 1

Abstract

In the past few decades, a lot of research efforts have been spent on furthering the automation of cars. It is expected that in the coming decades more and more cars will become highly automated, and that these automated modes will become available for increasingly difficult traffic scenarios. Highly automated vehicles do, however, imply transitions of control (ToC) from the automated vehicle towards the driver and vice versa.

In my literature review, I describe which factors are important for ToCs. It was found that a ToC involves several human factors questions that still need to be answered. One of these questions is how much time is needed to do a safe and comfortable ToC from automated to manual driving. Four experimental studies were found that tried to determine this time. Although the safety of a ToC depends greatly on a person's situation awareness (SA) level, only one of the four reviewed studies actually measured SA. Thus, a clear answer to the question 'how much time is needed for a ToC' was not found.

Next, in my literature review, I investigate what SA actually is and how SA can be measured. Different theories and models of SA, together with the different strengths and weaknesses of measurement techniques, are reviewed. Possible enhancements of the theories and models of SA are discussed.

Finally, I propose an experimental design for measuring the time needed to (re)gain a sufficient level of SA in case of a ToC.

1. Automated driving – general background information

In the past few decades, a lot of research efforts have been spent on furthering the automation of cars. More and more systems come onto the market that take over driving tasks, or assist the driver with a particular activity. The benefits of automating certain driving tasks lies in the fact that technology can outperform humans in both precision and speed and that such performance will consequently ease congestion on the roads [1 2 3 4 5]. Additionally, an economic benefit can be achieved by both lower fuel consumption and shorter transit times [4]. If the penetration rate of automated cars is high enough, platooning becomes a possibility and the fuel consumption and time benefits can become even greater. An environmental benefit is also expected due to the reduction in fuel consumption.

In addition to large-scale benefits, automation can also directly benefit the driver and other occupants of each individual car. The automation taking over (part of) the driver's tasks makes other tasks easier for the driver [4 5] and for highly automated modes even enables him to engage in other activities, making the ride more useful or more fun. The faster reaction time and unbiased detection capabilities of automation may yield a safety benefit that could be substantial, because according to Treat et al. (1979), and Stanton and Young (2010) human error is the major cause of car accidents [6 7]. Hoeger et al. (2011) and Stanton and Marsden (1996) even reported that 95% of car accidents are driver related or partially caused by human error or violation of traffic rules [8 9].

According to Stanton and Marsden (1996) it is expected that car sales will increase when highly automated cars are introduced [9]. This is due to the 'novelty factor' of such cars as well as the other expected benefits, as were described above.

Technological developments towards more automation in the automotive industry are already well underway. In 2014, BMW, among others, brought cars to the market that will be equipped with a partially automated mode (where the 'driver' only needs to monitor the system and the surroundings1), in the form of a traffic jam assistant [10]. They expect that in 2020 partial automated modes will become available for more traffic scenarios and that 2025 will yield cars that have a highly automated mode (where the 'driver' can be completely out of the loop1) for certain situations [11]. Although the technical possibilities for highly

[1] According to the definitions taken from Tom M. Gasser (BASt) (as accepted by the iMobility WG on Automation), see Chapter 2.

automated cars are already here (see the Google Driverless Car [12]), some important steps still need to be taken in order to make it commercially viable in terms of costs and appearance of the car regarding its sensors.

Furthermore, although there is a tendency to increase the level of automation of ground vehicles, these systems will probably not be able to cope with all traffic scenarios in the coming few decades. This fact implies that, depending on the situation, control has to be handed over between the driver and the automation.

Next to the technological issues stated above, there are also human factors issues that need to be resolved before highly automated systems can become commercially available. The issues that might occur can be predicted, to some extent from experiences from automating aircraft [8 9 12 13 14]. These issues generally relate to the fact that with a higher level of automation, the human becomes less involved in the operation of the system, or can even be placed completely out of the control loop (being 'just' a supervisor). This causes different human factors issues [15 16 17]. Situation awareness (SA), one of these human factors issues, has become "a core theme within the human factors (HF) research community" [18].

Especially for vehicles that have both a manual and an automated mode, SA is an important factor due to the needed transitions of control (ToC) between the vehicle and the driver. In case of automated driving, the driver is 'out of the loop' and has to be prepared to take over control. One of the specific questions that needs to be answered regarding ToC is: what is the time a driver needs to be able to safely and comfortably take over control of the vehicle after driving in a highly automated mode?

Aim of this literature report

This report reviews the definitions and human factors issues regarding ToC in automated driving. Next, we review the empirical evidence on one of these HF topics: the effect of 'take-over request time' on SA. Furthermore, it is investigated what SA actually is and how it can be measured in order to find out how the effect of 'take-over request time' after driving in a fully automatic mode could best be researched.

2. Transitions of control in automated driving

Transitions of control are needed when a vehicle has multiple driving modes (i.e., levels of automation) ranging from manual to fully automated driving. Apart from transitions being between different levels of automation, there can be different reasons for initiating a ToC. This in turn determines the way the ToC is done.

2.1. Levels of automation

In order to be able to classify the different driving modes, Flemisch et al. (2008) ([19], [20]) proposed the view that automation can be seen as a continuum ranging from completely manual control to fully automated control, see Figure 1. According to Flemisch et al. (2008) this continuum can be described in two different, but related ways: "The level of involvement of human and machine in the control of a human-machine system for a specific moment" and "The automation capabilities of a specific vehicle" [19].

Flemisch et al. (2008) [20], and the HAVEit [8] and CityMobil projects [21] use the automation capabilities to define the different levels of automation, which can be summarized as follows:

Manual/ Driver only: The driver manually controls the car completely.
Driver Assisted: One or several driving tasks are accommodated with haptic shared control and/or warnings. Only in case of emergency, the automation can temporarily take full control of one or several tasks.
Partial-/semi-automated: Either longitudinal or lateral control is automated. The other tasks will be done manually, or are assisted.
Highly automated: Both longitudinal and lateral control are automated. However, the driver still has to stay focussed. The system cannot navigate by itself and/or the environment may contain elements the automation cannot cope with.
Fully Automated/autonomous: Both longitudinal and lateral control are fully automated.

There are definitions based upon the level of involvement of the human as well, which are the ones used in the remainder of this report, from Tom M. Gasser (BASt) (as accepted by the iMobility WG on Automation) [22]:

Driver only: "The human driver executes the driving task manually."
Driver Assistance: "The driver permanently controls either longitudinal or lateral control. The other task (i.e., lateral or longitudinal control) can be automated to a certain extent by the assistance system."
Partial automation: "The system takes over longitudinal and lateral control. The driver shall permanently monitor the system and shall be prepared to take over control at any time."
High automation: "The system takes over longitudinal and lateral control. The driver must no longer permanently monitor the system. In case of a take-over request, the driver must take-over control with a certain time buffer."
Full Automation: "The system takes over longitudinal and lateral control completely and permanently. In case of a take-over request that is not carried out, the system will return to the minimal risk condition by itself."

When using both (somewhat related) sets of definitions together, different systems can be placed in a graph that has 'degree of automation' on one axis and 'degree of cooperation/autonomy' on the other [23], see Figure 2**Error! Reference source not found.**. ToC can occur between every level, as depicted in Figure 3. The degree of automation can shift either towards more automatic or more manual control.

2.2. Why and how to do transitions of control

There can be different reasons for performing a transition of control. The driver can set the automation to a specific mode (on purpose or accidental), but the automation can initiate a ToC as well. This can be due to the automated mode reaching its working boundaries or by malfunction. In these cases, the situation will have a certain degree of time-criticality, which influences task demands for the human operator.

Some projects like HAVEit [8], CityMobil [24] and InteractIVe [25] have derived guidelines either for distinguishing the different transitions or for designing a highly automated system (incorporating transitions) and have derived differences between transitions. In short, the different possible types of transitions can be found by combining the answers to the following questions:

- Does the transition go to, or come from, a higher automation mode?
- Who initiated the transition: driver or automation?
- Is the transition initiated voluntarily or not?
- Is the transition time-critical (i.e., emergency condition) or not?
- Was the transition successful or not?

Hoeger et al. (2011) [8] devised a short notation for types of transitions of control. This work does not, however, take time-criticality into account (see Table 1).

The HAVEit [8], CityMobil [24] and InteractIVe [25] projects also state that, depending on the type of transition and technology used, different transition protocols need to be devised. These protocols state which steps need to be taken by the driver and/or the automation, what feedback is needed and how much time is needed for each step during the process of ToC. Specifically, answers to the following questions were sought:

- What interactions should there be between the human and the automation, and how much time is needed for these interactions, before, during, and after the ToC?
- How to provide enough (but not too much) and clear feedback about the ToC?
- How to keep/get the driver sufficiently in the loop and/or how to let the driver (re)gain situation awareness to be able to do a ToC?
- How much time is available and how much time is needed for the ToC?

Three main types of transition protocols were found, which can be combined in part to get other transition protocols: 1) 'instantaneous', where the driver gets full control when he pushes a certain button, brakes or

accelerates[2], or turns the steering wheel.[2] 2) 'stepwise', where the automation hands over control in steps (i.e., first steering and then the foot pedals or vice versa) or gradually (i.e., by gradually decreasing the strength of the haptic guidance/feedback). 3) 'time based', where the driver is presented with a countdown (or something of the like) of the time he/she has got left to take over control. When the driver fails to take over control in time, the automation will go into a safety mode[3].

Researchers of the HAVEit project [8] have done a driving simulator study using different protocols for transitions and have studied which type of transition was subjectively assessed as 'best' or most natural. The findings were also compared to expectations of participants prior to using the automation. From this, it was found that expectations of the drivers on transition protocols are dependent on transition protocols for existing automation (i.e., when the same transition protocol is used as when cruise control is turned on, less mistakes are made and the transition is experienced as more comfortable). Furthermore, differences in expectations and the transition protocols that were used caused some confusion, but were adapted to quite fast and without major problems. The 'instantaneous' protocol was assessed as 'best'.

Although there have been experiments that investigated the effects of different types of protocols [21 26 27], almost none took the needed preparation time of the driver into account or they simply lacked detailed specifications of such protocols. One of the most important variables, *time* (both available and needed time), is often overlooked. When there is less time available than is actually needed for the ToC, the ToC has a high chance of failing.

3. The effect of 'take-over request time' on Situation Awareness: a review of experimental research

For cases where ToC is done from the automation towards the driver, it is of importance that the driver regains appropriate SA before control is handed back to him, to assure a safe and comfortable ToC. Research has shown that, when a person has been out-of-the-loop, reaction times are high, and a high workload is experienced at the moment of transition [15 17 28]. In spite of these findings, not much research is available on the time needed to regain SA and/or to take over control after a period of (distracted) automated driving. Following is a short review of the research currently available.

3.1. Damböck (2013)

Damböck (2013) [29] tried to determine in which order of magnitude the take-over request (TOR) time should be to ensure a successful and comfortable ToC. From Schmidtke (1993) and Schweigert (2003) [30 31] it had been found that, based on handling times (i.e., reaction on audible signal, gaze movement towards the scene, grabbing the steering wheel and placing feet at pedals, and doing safety checks), taking over control and doing a lane change would take between 3.07 and 3.52 seconds, not taking into account the following four factors: 1) How long does it take a driver to take in the necessary information (scanning)? 2) How long does it take to make a mental representation of the situation, based on the needed information? 3) How long does it take to choose an action? 4) How long does it take to do a lane change?

The authors carried out a fixed-base driving simulator study where participants (n=32) drove in a highly automated car and, during automated driving, were fully out-of-the-loop due to a highly demanding secondary task. In two trials of about 30 minutes, manual and automatic driving was alternated. For ToCs from automation to manual, audible TORs were provided, 4, 6 and 8 seconds prior to a situation where automation limits were reached. There were three different scenarios in which the automation limits were reached: 1) road markings ended and the driver had to keep the car in the lane, 2) the road narrowed and a lane change had to be done, and 3) the road split up in three ways, the driver had to check the navigation to see to which lane he/she had to take and do a lane change. These scenarios were also encountered during manual driving, where no TOR was provided, the outcomes of which were used as a baseline test.

[2] Certain thresholds are used to ensure that the input from the driver was voluntary.
[3] Details of this mode are still debated, one possibility is that the automation will steer the car onto the hard shoulder.

It was found that TOR-time had a significant positive influence on subjective (NASA-TLX) comfort ratings (i.e., more time yielded higher comfort ratings), where the scenario had a significant negative influence on these ratings (i.e., more difficult scenarios got lower comfort ratings), although for the lane-change scenario no significant difference was found between the 6 and 8 second TOR-time ratings.

All participants successfully finished the 'no road markings' scenario. For the 'narrowing road' scenario, however, significantly more errors (making the lane-change too late) were made with the 4 and 6 s TOR-times than in the baseline test. The 8 s TOR-time did yield more errors than the baseline test, but the difference was not significant.

In the 'splitting road' scenario the same results were found as in the 'narrowing road' scenario when looking at the total number of errors compared to the baseline test. In this scenario, however, different types of errors could be made (no lane change, change to wrong lane, and change too late). If only the right lane-changes were taken into account, only the 4 s TOR-time showed a significant difference with the baseline test.

Next to performance measures, several time-related measures were assessed as well. The time it took until the first reaction to the audible TOR, measured as the first saccade away from the secondary task, and the time it took from the TOR until the first fixation on the road, on average showed no large differences to the values found in [30] for all scenarios. The differences that were found could be explained through participants' physical and mental reaction capabilities and through the different instructions that were given (either to take over as fast as possible or as comfortable as possible), but not through different TOR-time or scenario. The time-related measurements furthermore showed that participants took more time to grab the steering wheel when TOR-time increased (from 2.4 to 2.6 s on average for the 4 and 8 s TOR-times, respectively). These values differed from the values found in [30 31]. Due to the lack of, or not on time, taking over control by a great part of participants, the time it took from the TOR until successfully taking over control for the different scenarios was not subjected to statistical analysis, because in this case only the fastest subjects would be assessed, and the slower participants and the participants who did not deem it necessary to take over control before the system boundary was reached, were not. The values that were found did, however, imply that differences in time needed to take over exist for different TOR-times, where more time was taken when more time was available.

The use of safety checks (using mirrors and doing a shoulder check) and indicators for the two lane change scenarios was assessed as well. It was found that in the 4s cases the inner mirror check was skipped significantly more than in other cases. Furthermore, the shoulder check was done more in the narrowing road test than in the baseline test, possibly because in the baseline test the drivers already had a dynamic picture of the situation and knew that the lane-change was safe.

3.2. Gold et al. (2013)

In Gold et al. (2013) [32], the study of Damböck (2013) [29] was elaborated upon. The study was done in a high-fidelity simulator, where the participants (n=32) drove in a highly automated mode with a demanding secondary task (hence, out-of-the-loop) on a three-lane freeway at 120 km/h. An audible and visible TOR was provided 5 or 7 seconds before a collision would occur if the driver would not react. The TOR was provided at moments when the lead vehicle suddenly changed lanes and a slow driving car was revealed. In a baseline study, the drivers (n=17) drove manually in the same situation, but without TOR and without a secondary task.

The different situations were compared on the use of security checks (i.e., using mirrors, looking over the shoulder, and using the indicator) and the type of intervention: braking, steering, or both, where making an evasive manoeuvre (just steering) was regarded as the 'best' option. The quality of an intervention was determined by the developed accelerations. Furthermore, times were measured from the TOR until the first gaze reaction (i.e., the first saccade away from the secondary task), until the first fixation on the road, until hands were on the steering wheel, until the first intervention (i.e., steering wheel or brake pedal input), until the use of the left side mirror, and until the use of the turn signal. For the baseline test, however, only the time until intervention was measured.

It was found that the different TOR times had a strong influence on the type of reaction for both the main-study groups as well as for the baseline-study group. For shorter TOR times, the brake was used more often. When the TOR-time is kept constant and SA is increased by manual driving, the same effect

appears. This implies that the brake is used to gain time for decision making (i.e., gaining better SA) or for performing the lane change. As Gold et al. (2013) [32] puts it: “It can be stated that the usage of the brake suggests an insufficient timeframe for the reaction”, which was the case for the majority of both the main-study groups. Furthermore, the braking inputs, on average, were done about a second prior to the steering input.

Although for the shorter TOR times the decision making was done quicker and thus the interventions done earlier (2.06 s vs. 3.10 s for braking, and 2.27 s vs. 3.65 s for steering, for the 5 and 7 s TOR-times, respectively), the quality of the reactions was lower (i.e., higher accelerations) and less security checks (i.e., looking over the shoulder and/or checking mirrors) were done before the evasive manoeuvres. In the baseline study, the same effects could be found, although the accelerations were two to three times lower (and thus more controlled) than in the automated case. Even for the 7 s TOR-time, the automation effects were still observable compared to the manual case.

3.3. Petermann-Stock et al. (2013)

Petermann-Stock, Hackenberg, Muhr and Mergl (2013) [33] did a fixed-base driving simulator study, which focussed on reaction times and driving behaviour after an audible and visible (on a special instrument panel on the dashboard) TOR was provided during highly automated driving in a traffic jam on a two-lane highway. Participants (n=72) were divided into two age groups (25-35 and 50-70 years old) and differences between these two groups and three different secondary tasks were compared. The basis for the workload modulation of three tasks was an expert evaluation (n=31) which classified potentially conceivable secondary tasks for automated driving concerning the controllability and the stimulus value.

Two TORs were done, both at 35 km/h. In the first case the lead vehicle drove so slow a collision had to be avoided. In the second case the steering wheel got a sudden steering impulse from the automation and a collision with traffic next to the vehicle had to be avoided.

It was found that reaction times increase with a more demanding secondary task, but fixation times on the special instrument panel (depicting the mode the car was in or ‘Übernehmen!’ (Take over!) when a take-over request was done) decreased. The type of secondary task, and specifically its visual demand, had significant influence on SA, which was determined by asking questions about the situation. Reaction times (from signal to actual take-over) were on average 3.2 seconds, with a maximum of 8.8 seconds.

The authors suggested that a time window for taking over control should be more than 10 seconds long. Questionnaires were also used, where 76% of the participants said they would have liked an earlier warning and the general consensus was that the time to take over control should be somewhere between 5 and 10 seconds in order to have time to take attention away from a secondary task.

3.4 Beukel and van der Voort (2013)

Van den Beukel and van der Voort (2013) [34] “assessed the influence of criticality (available time) on SA drivers obtain when retrieving control after automation” in a fixed-base driving simulator equipped with a congestion assistant and a demanding secondary task.

Two methods of measuring SA were used: the Situation Awareness Global Assessment Technique (SAGAT, where the simulation is blanked and questions are asked about the situation)[4] and the Situation Awareness Rating Technique (SART, a self-rating technique of SA)[4], in three different levels of criticality. The critical situations were formed by the lead car suddenly braking with 8 m/s^2 from a velocity of 60 km/h at different time-headways (0.5, 1.0 and 1.5 seconds, resulting in available times (without reaction) of 1.5, 2.2 and 2.8 seconds, respectively). The TOR was provided by an alarming sound at the beginning of the sudden braking of the lead car. Participants (n=34) were instructed to take over control as soon as the situation required it and were advised to swerve out if possible in case of emergency, but any kind of accident avoidance was allowed.

[4] See Chapter 5 for a more in-depth explanation of these techniques.

It was found that the percentage of accidents decreased, and that the self-rated (SART) scores were higher, with more available time. The SAGAT scores showed no particular relation to time-criticality, possibly because the SAGAT method is not suited for too time-critical situations, or because the questions asked were not specific enough for the dynamicity of the situation.

4. What is Situation Awareness: An overview of three common theories

Although SA has become a main topic of research since the late 1980s, initiated by the (military) aviation industry, researchers seem to be unable to agree on a single definition or theory of SA. In general, they all encompass "knowing what's going on" [35]. The three dominating theories and definitions, according to Stanton, Chambers and Piggot (2001) [36], are (1) the three-level model by Endsley (1988, 1995) [37 38], (2) Activity Theory by Bedny and Meister (1999) [39], and (3) the Perceptual Cycle Model by Smith and Hancock (1995) [40], which are described below. The main difference between them being the extent to which SA is regarded as product or as a process, that is, a state of knowledge (product) or the process of acquiring such knowledge (process).

4.1. Three-level model

The most commonly used definition was proposed by Endsley (1988) [37], who defined SA as follows: "Situational awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and a projection of their status in the near future." Later, in 1995, she proposed the three-level model [38], describing SA as a (mostly conscious) state of knowledge (product) containing three hierarchical levels of cognitive processing of data (prior to decision making) in dynamic situations, as can be seen in Figure 4, which also depicts some (individual-, task-, and system) factors that can influence SA. In addition, SA is based on different information-processing components: short-term sensory memory, working memory, long-term memory, schema and scripts, as can be seen in Figure 5. SA is the prerequisite for decision making and acting. Having a good SA does not, however, guarantee good decisions, nor does having bad SA guarantee bad decisions [41].

Level 1 SA: Perception of the Elements in the Environment

In Endsley (1995), the first level of acquiring SA (coincidently focused on car driving) is described as follows: "The first step in achieving SA is perceiving the status, attributes, and dynamics of relevant elements in the environment. […] An automobile driver needs to know where other vehicles and obstacles are, their dynamics and the status and dynamics of one's own vehicle." [38]. All the errors in SA associated with the first level are some form of failure to perceive certain information. By this definition, what constitutes misperception is obscured information or failures in information sampling (often accompanied with stress).

Level 2 SA: Comprehension of the Current Situation

Comprehension of the current situation is achieved by combining the knowledge of the elements obtained in level 1, forming a holistic picture of the environment. In addition to just being aware of the elements, the significance of these elements is determined, depending on the operator's goals [41]. Prior experience with the level 1 SA elements at hand could yield better level 2 SA, from having a more advanced/correct mental model of the situation, which makes the integration of the data easier.
Errors in the second level of SA are most often caused by a person's inability to properly integrate information or interpret the data in light of his/her goals. Lack of skill and/or a sufficient mental model of the operator makes the operator incapable of discerning the most relevant cues for the most important goals. Another possibility is that the operator has chosen a certain model from memory, which is not the correct one. Yet another possibility is that salience of certain cues is not sufficient (error in level 1), causing a mismatch with the (correct) model [41].

Level 3 SA: Projection of Future Status

The ability to project the future state of the elements together with changes in the situation itself is defined as level 3 SA [41]. Projecting the future status "is achieved through knowledge of the status and elements and comprehension of the situation (both Level 1 and Level 2 SA)." [38]. The anticipated future state of elements is thus determined by a person's knowledge about -or understanding of- dynamic properties of elements and their current dynamical state in light of the current situation and the projected future situation. How far ahead in time these projections go depends on how fast the situation and the elements are

changing and how significant the elements were deemed in light of the situation and the person's goals. Highly dynamic systems and elements are more difficult to predict accurately due to more possible outcomes in the same time span. The better the knowledge about (general) dynamical properties and capabilities of elements and the current situation is, the better their future states can be predicted. Errors in this level are due to lacking or incorrect SA of the current situation (errors made in assessing dynamical properties or –status of elements), which thus are errors made in the previous two levels, or by lack of skills in predicting the future state.

4.1.1. Pros and cons of the Three-level model

As stated earlier, the Three-level model is the most commonly used model for SA measurements. It presents SA in a simple manner and is applicable in many domains. Including the factors that influence SA [41], this theory also provides guidelines for designing different working environments and training strategies.

One of the main weaknesses of the Three-level model is that it seems definitive when the third level is reached, failing to cope with the dynamic nature of SA. Endsley defines SA as a product, but it can be argued that the three levels in both her model and definition of SA actually describe processes to obtain SA. Furthermore, researchers have criticised Endsley for the constructs she uses to base her theory on, because they are themselves poorly understood or ill-defined [36 42].

For a short description of the main strengths and weaknesses, see Table 2.

4.2. Activity theory

In the late nineties, Activity Theory (AT) came to the Western world from Russia, where it had been developed since the 1930s as a psychological way to study work behaviour. Where in the Western world work behaviour was approached from different, separate theories, in Russia it was tried to combine all separate theories. Although it has been quite an unfamiliar cognitive theory for Western psychologists and ergonomists, AT is rapidly winning terrain as a versatile theory that combines different, both physical and mental, constructs of work behaviour in one model.

In AT 'activities' are used to describe both physical and psychological human behaviour. According to Kuutti (1996), activities can be described as "the minimal meaningful context to understand individual actions." [43]. This context is important for analysing real-life situations in laboratory settings. Activities are always goal-directed and driven by a motive, organized by the goal and mechanisms of self-regulation (reflection on actions, performance and goal) [44]. A goal is an ideal image or desired end state of activity [36 39 42].

An absolute definition of activity does not exist, because it is not a constant or static entity. Not only is it dependent on the situation, which is not static in general, but it is also always developing while a person is working towards a goal and reflecting on his actions. [43]. Bedny Karwowski and Jeng (2003) however note that the major principles of activity are "unity of different internal, mental and motivational processes and their connection to external behaviour." [45]

AT is modelled by a series of logically ordered function blocks, see Figure 6. The whole framework can be utilized both with conscious and unconscious processes, except for the goal, which is always conscious. A great difference with other models of behaviour is that a person's reaction to specific stimuli is not necessarily an extrinsic/deterministic reaction, but "she or he updates the input information, forms different goals according to certain personal motives, and organizes her or his behaviour to achieve these goals" [46].

Activity can be split into three stages: orientation, execution, and evaluation [39 43 44], which all influence, and are influenced by, a person's desired goals. The stages are not a specific part of the model from Figure 6, but for each stage the interactions between the blocks differ.

1) In the orientation stage, explorative actions (both mental and motor) are done in order to understand the situation. form a subjective model of reality -in light of the desired goals- from which distinct representations are actively extracted, eventually forming a dynamic picture of the world. This picture is used for the interpretation of reality and anticipation of the future state of the situation. The oriental stage encompasses SA (in the subjectively relevant task conditions block, see Figure 6), in broad lines

similar to Endsley's model, but other than in Endsley's model, also the operative thinking process (next to memory and attention) is of importance.
2) In the executive stage, the person moves towards his/her desired goal. These movements consist of decision making and performance of (mental or physical) actions, which are directed to achieve the desired conscious goal.
3) In the evaluative stage, the result of the activity is assessed through information feedback. This assessment involves corrective influence on both the orientation and the executive stage, by altering, for instance, the subjective model or decisions/actions, respectively. The corrections depend upon the goals, level of motivation, and the results of the subjective evaluation itself.

4.2.1. Activity Theory and Situation Awareness

Gaining SA (process) and having SA (product) is part of the orientation stage. When the main goal is the dynamic reflection of the situation and when only those components that precede the performance of an executive action are considered[5], then the AT model, see Figure 6, can be simplified to a functional model of orientation activity consisting of eight function blocks, which encompass SA as part of Subjectively Relevant Task Conditions (block 3), see Figure 7. [36 39]

In the Subjectively Relevant Task Conditions block it is determined which features of the world are meaningful, based upon the perceived significance to the task goal , a person's motivation to find these features (block 4a and 4b, respectively) and the outcome of the orienting and explorative actions (block 5). In block 3, an operative image is formed which is in part conscious, which is named situation awareness, and in part not conscious. It influences how a person (decides to) interact with the world (block 5) and it is evaluated if enough effort is put in gaining the operative image (through block 4). Bedny and Meister (1999) define SA in that context as follows: "Situational awareness is the conscious dynamic reflection on the situation by an individual. It provides dynamic orientation to the situation, the opportunity to reflect not only the past, present and future, but the potential features of the situation. The dynamic reflection contains logical-conceptual, imaginative, conscious and unconscious components which enables individuals to develop mental models of external events." [39]

4.2.2. Pros and cons of reflective-orientational activity model

The reflective-orientational activity model offers a clear description of the roles of the different functional blocks for acquiring and maintaining SA, and handles SA as a separate entity. It is therefore possible to use this model to describe SA both as process and as product. Furthermore, it offers a more dynamic description of SA than the Three-level model, especially how SA dynamically modifies interaction with the world and how interaction with the world dynamically modifies SA.

A downside of the model, as depicted in Figure 7, is that it seems to miss a feed-forward link from block 2 (image-goal) to block 4 (Evaluative and inducing components of motivation) and a link from block 5 (Performance and explorative actions) to the world. The utility of the model is hindered through lack of a definition of activity itself. Furthermore, the model lacks empirical evidence. This is partly caused by that it has not been fully embraced by psychologists, but also due to the fact that no measurement techniques have been suggested, which makes it a difficult theory to use.

For a short description of the main strengths and weaknesses, see Table 2.

4.3. Perceptual cycle model

Smith and Hancock (1995) [40] describe SA as both process and product by expressing it in terms of Neisser's perceptual cycle model [47]. They suggest that SA resides through a person's interaction with the world rather than exclusively being part of either the person or the world itself and they see it as "purposeful behavior that is directed toward achieving a goal in a specific task environment." [40] They define SA as: "the invariant in the agent-environment system that generates the momentary knowledge and behaviour required to attain the goals specified by an arbiter of performance in the environment." [40], where the

[5] "Because each function block includes in different was the same psychological processes, the use of concepts such as memory, thinking, and so forth is not useful at this stage of analysis" [39]

arbiter of performance encompasses the goals and criteria for assessing performance variables adherent to the task, thus being part of the task environment.[6]

The interactional behaviour with the world (also called 'explorations') is directed by internally held schemata and a person's own goals. These schemata, however, are updated by the outcome of the interaction, influencing further explorational actions, thus creating an infinite cycle [42], see Figure 8. Adams, Tenney and Pew (1995) [48] modified the model shown in Figure 8 using the theory from Sanford and Garrod (1981) [49], which states that long-term memory can be split up into episodic and semantic memory, and that short-term (active or focal) memory can be split up into explicit focus and implicit focus, see Figure 9. Long-term semantic memory contains everything that has been learnt so far. Long-term episodic memory is the knowledge gained during the current task. Implicit focus is the complete schema for the current situation, where explicit focus (conventionally labelled as working memory) contains a limited number of features that are attended to at that moment, which contain tokens of (or pointers to) knowledge structures in long-term memory [48]. Information that is relevant to knowledge in the implicit focus domain takes longer to be sampled (noticed) and modified (interpreted) than information relevant to explicit focus. An example, from aviation, of the different types of long- and short-term memories can be seen in Figure 10.

4.3.1. Pros and cons of the Perceptual cycle model

The Perceptual cycle model has as great advantages that it has sound underpinning theory from Neisser's work [47] and that it describes both the process (as the continuous sampling of the environment) and the product (as a schema that is continually updated) of SA. The descriptions of both Smith and Hancock (1995) [40], and Adams et al. (1995) [35] provide for the dynamic nature of SA, possibly making the Perceptual cycle model the most complete description of SA.
The completeness of the description of SA is also somewhat the downfall of the Perceptual cycle model, because it makes it difficult to shortly describe what SA is in a manner that it can be measured. So far, Task Performance measurements, of which the link with SA is not direct, are the only forms of measurement of the cycle [36 42].

For a short description of the main strengths and weaknesses, see Table 2.

4.4. Comparison of the three SA theories

Although the three models/theories above may seem to be quite different, they actually show quite similar aspects. All see SA, at least partially, as a state of knowledge, where the process of gaining SA is consciously goal-directed, based on previous experiences and mental models, which are all updated by what is currently experienced (reflection). Stanton, Chambers & Piggot (2001) [36] found a way to integrate the three SA definitions. They present three main elements in a system: The person, the world, and the interaction between them. The person is divided into three main sub-systems: Working memory, mental models, and reflection, see Figure 11. Each of the three theories uses these blocks, albeit in a somewhat different manner. All theories have their strengths and weaknesses: The Three level model is easy to measure and very practical, but lacks the dynamic description of SA and underpinning theory; Activity theory describes SA, both as process and product, well, but is hard to understand and to measure; Perceptual cycle has a great underpinning theory, but also lacks direct measurement methods. In short, all theories and models have their merits when it comes either to underlying theories or to applicability.

For a short overview of the main strengths and weaknesses of the different models, see Table 2.

5. How can Situation Awareness be measured?

Situation awareness can be measured in several different ways. The different measurement techniques are based upon the different models described earlier. The techniques can be subdivided into six categories described below, as was done by Salmon, Stanton, Walker et al. (2009) [50]: 1) Freeze probe techniques, 2) Real-time probe techniques, 3) Self-rating techniques, 4) Observer rating techniques, 5) Performance measures, and 6) Process indices.

[6] Goals can be split up in task specific goals, having their own inherent criteria for performance, and personal (sub)goals, defining, among others, a person's desired level of performance.

5.1. Freeze probe technique

Measures based on the freeze probe technique are done during virtual simulations of a real task, where at certain moments the simulation is frozen (i.e., the screen goes black), during which SA related queries are done. Several queries (related to perception, comprehension, and prediction) need to be answered based upon the participant's knowledge and understanding of the situation at the point of the freeze. When the queries are finished, the simulation continues where it was paused. After the experiment, the investigator compares the participants' answers to the actual state of the simulation at the freeze moments, and calculates a corresponding SA score.

The strength of this technique lies in its direct and objective nature, circumventing issues with collecting SA data post-trial.[7] The technique is, however, also heavily criticised due to its intrusive nature on task performance. Furthermore, SAGAT (generally) cannot be applied to real-life scenarios. The validity of the method is questioned regarding if it the technique actually measures SA or memory instead, and how it can be made sure that appropriate probing questions are used.

Endsley's (1995) [51] Situation Awareness Global Assessment Technique (SAGAT) is probably the most popular freeze probe technique according to [50].

5.2. Real-time probe techniques

Real-time probes are queries that are taken during the performance of the task, without freezes. The technique was developed to limit the level of intrusion experienced with the freeze probe technique. The queries are usually developed prior to the task by Subject Matter Experts (SMEs), but in some cases an on-the-spot approach is used, where the questions are produced during the task by the investigator(s). SA is scored by the answer content and the response time for correctly answered questions.

The strength of the real-time probe technique lies in its lower level of intrusiveness than freeze probe techniques. They still, however, have the same other downsides as freeze probe techniques have.

One example of a real-time probe technique is the Situation Present Assessment Method (SPAM), which is described by Durso et al. (1998) [52].

5.3. Self-rating techniques

With self-rating techniques, participants rate their own perceived level of SA on some rating scale. Self-rating techniques are often used because of their ease of use and their lack of intrusiveness, because they are usually administered post-trial. They are, however, heavily criticised. There might be problems with recalling certain (parts of) situations after a trial, even though the participant might have had good SA. Furthermore, subjective rating is subjective, meaning that it can be influenced by several personal factors (i.e. mood, fatigue etc.) and biases.

The most popular self-rating technique is the Situation Awareness Rating Technique (SART), introduced by Taylor (1990) [53].

5.4. Observer rating techniques

With observer-rating techniques one or several experts rate participants' SA during task performance, based on pre-defined, observable, SA-related behaviours. The main advantage of these techniques is that it has no impact on the task and can be used in non-simulator tests. The main disadvantage regards validity issues, where it is unknown if an observer can actually assess someone else's SA.

[7] A post-trial query would not have the benefits of the freeze probe- or the real-time probe technique and would still have the downsides.

5.5. Performance measures

Performance measures are used as indirect measures of SA. Depending on the task at hand, certain aspects of performance are recorded. Different aspect measures can be either intrusive or non-intrusive, making performance measures applicable to many situations. There is, however, a problem with the validity of these measured, because the level of A does not necessarily have an effect on task performance.

Gugerty (1997) [54] used different performance measures during a simulated driving task. Hazard detection, blocking car detection and crash avoidance were assessed.

5.6. Process indices

Process indices are recordings of the processes that participants go through in order to develop SA during the task at hand. Eye-tracking, combined with verbal protocol analysis, is one of these measures, where assessments of eye movements, fixation locations, and fixation durations can be used to determine how participants' attention was allocated during the task. A transcript of operator behaviour, based upon the operator 'thinking aloud' gives insight into the cognitive aspects of his/her behaviour and can be used as indicator of operator's SA during task performance. Eye-tracking cannot, however, determine with 100% certainty that something that was looked at was actually or vice versa. That is, sometimes things that were focussed on cannot be recalled, whereas things in a person's peripheral view are recalled.

6. Discussion and recommendations for future research

In this literature review, we reviewed the definitions and human factors issues regarding transition of control (ToC) in highly automated driving. Furthermore, the current empirical evidence regarding the effects of 'take-over request time' (TOR-time) on situation awareness (SA) was reviewed. We also investigated what SA is and how it could be measured. Here it is discussed how this knowledge could be used to set up more research into the effects of TOR-time on safety and comfort of ToCs, regarding SA. Furthermore, it is discussed how the different theories of SA could be improved/elaborated upon.

6.1. Effects of TOR-time on safety and comfort of ToCs, regarding SA

Although researchers are well on their way to make highly automated driving possible, there are still a number of human factors issues to be solved. Lack of situation awareness prior to/during a transition of control from a highly automated towards a manual mode, could especially become an issue for the developers of automated driving systems. The empirical research reviewed in Chapter 3 did not find a basic value for time needed to retake control, although it was found that automation effects are still observable with a TOR-time of 7 s. Although [29], [32] and [33] measured times for different phases of a take-over, they do not measure how much time the driver needs to (re)gain SA. Only [34] actually tried to determine SA, but did not compare the results to SA scores for manual driving. Clearly, more research is needed in order to find out how SA builds up over time. In order to acquire such knowledge for a specific situation, SA measurements should be done after a specific length of time in that situation. Repeating this for several lengths of time (in comparable situations) should provide some insight into the buildup of SA. For a ToC situation (from automated to manual), participants could be placed out-of-the-loop in an automatically driving car[8], where they are provided with a TOR. The time from the first gaze at the scene until a set time when the simulation is stopped, is the time the participant had to build up SA. Comparing SA scores achieved at different time lengths will provide information about how SA builds up over time, when the participant has been out-of-the-loop.

In order to determine how much time is needed for a safe and comfortable ToC, it needs to be determined what level of SA is *required* and how much time it takes to reach that level. One possible way to measure this is to place participants out-of-the-loop in an automatically driving car, provide them with a TOR, and stop the simulation at the moment that the participant claims control of the car. The time from the first gaze at the scene until the stopping of the simulation is the time the participant deems safe and comfortable, and is the time he/she had to build up SA. An SA and a comfort measure directly after the simulation will

[8] Simulated, of course.

provide a baseline value of what should be a comfortable SA level and transition time for a ToC. The safety level, however, should be checked by an objective (performance) test. This test should be comparable to the other test, but now the simulation is not stopped and driving performance is measured during and after the ToC.

When the time needed to (1) gain a specific level of SA, (2) safely and comfortably take over, and (3) handling time measurements [29 32 33], are combined, this could lead to a better insight into the actual time needed for a *complete* ToC.

6.2. SA theories

As was described Chapter 4.4, all three theories that were discussed contained several corresponding aspects: 1)SA is, at least partially, a state of knowledge, 2) the process of gaining SA is consciously goal-directed, based on previous experiences and mental models, 3) The mental models are updated by what is currently experienced (reflection). Furthermore, all three theories seemed to have their own strengths and weaknesses. The Three level model has weak theoretical underpinnings, but has been proven useful in practice. The Perceptual cycle and the Activity theory have sound theoretical underpinnings, but need to be validated. It might be possible to combine the theories in order to overcome the theoretical weaknesses of the Three level model in particular. Another possibility would be to specify measurement techniques for both the Perceptual cycle model and Activity theory. Future research should clarify which theory/model is most feasible in practice, depending on the situation and/or task.

6.3. Measurement techniques

Regarding measurement techniques, it can generally be recommended to use several SA measures in order to overcome some of the weaknesses. The ideal SA measure (combination) is direct, non-intrusive, objective, and applicable in all scenarios. Furthermore, it is recommended that SA theories and measures get an additional ‘time’ scale, where certain levels or achievements are linked to the time it takes to complete them. Doing so could increase our understanding of what SA is which (psychological) processes are involved.

Figures

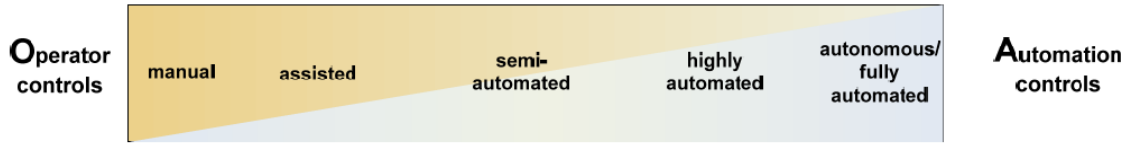


Figure 1: Continuous automation spectrum with different levels. Taken from Flemisch et al. (2008) [19], adapted by Martens, Pauwelussen et al. (2011) [21]

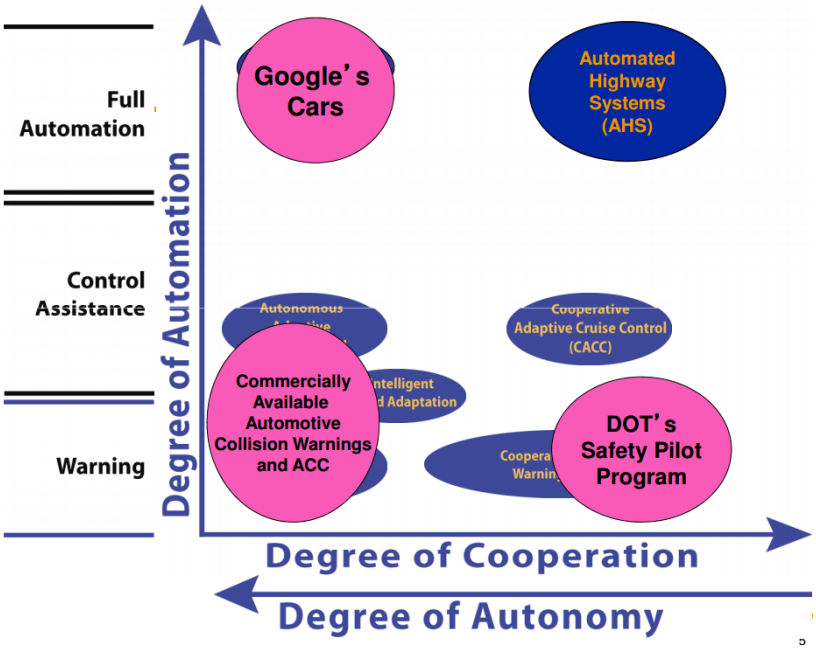


Figure 2: Classifying automation systems. Taken from iMobility WG on automation in road transport [23]

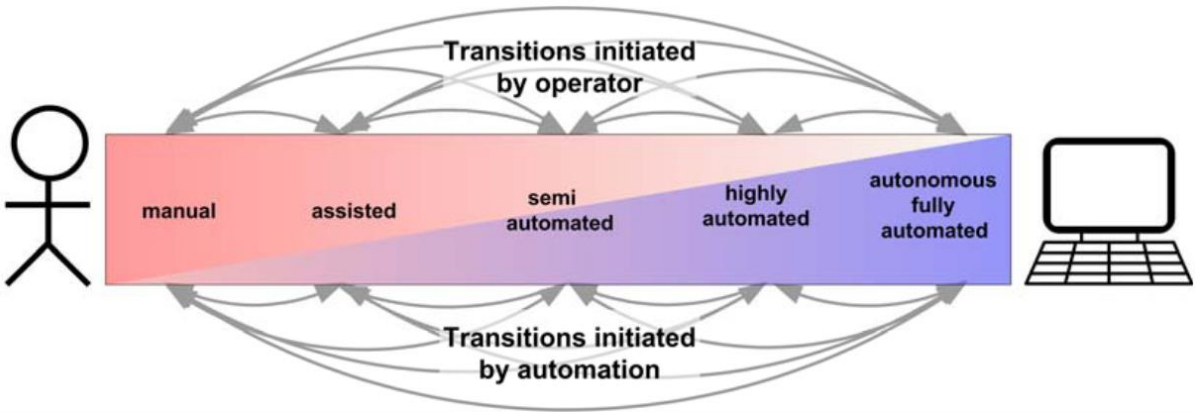


Figure 3: Transitions between different levels of automation. Taken from Flemisch et al. (2008) [20]

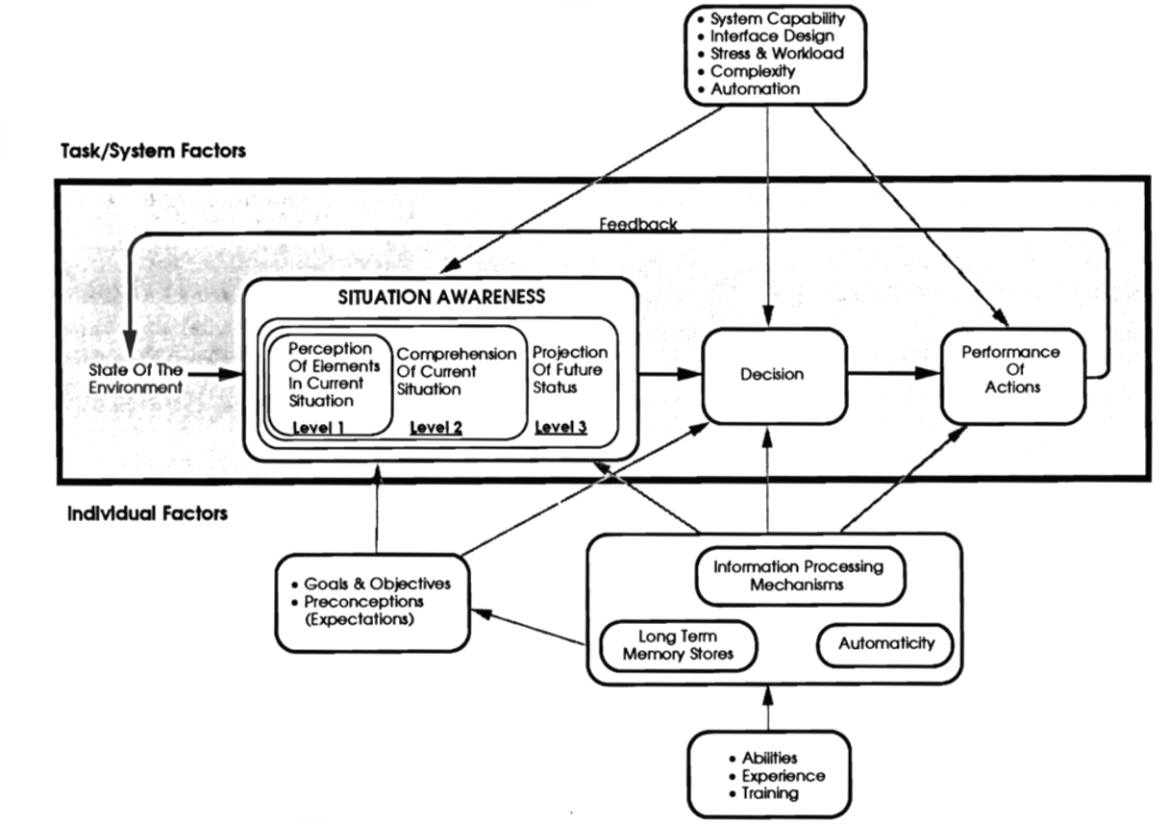


Figure 4: Model of Situation Awareness in dynamic decision making. Taken from Endsley (1995) [38]

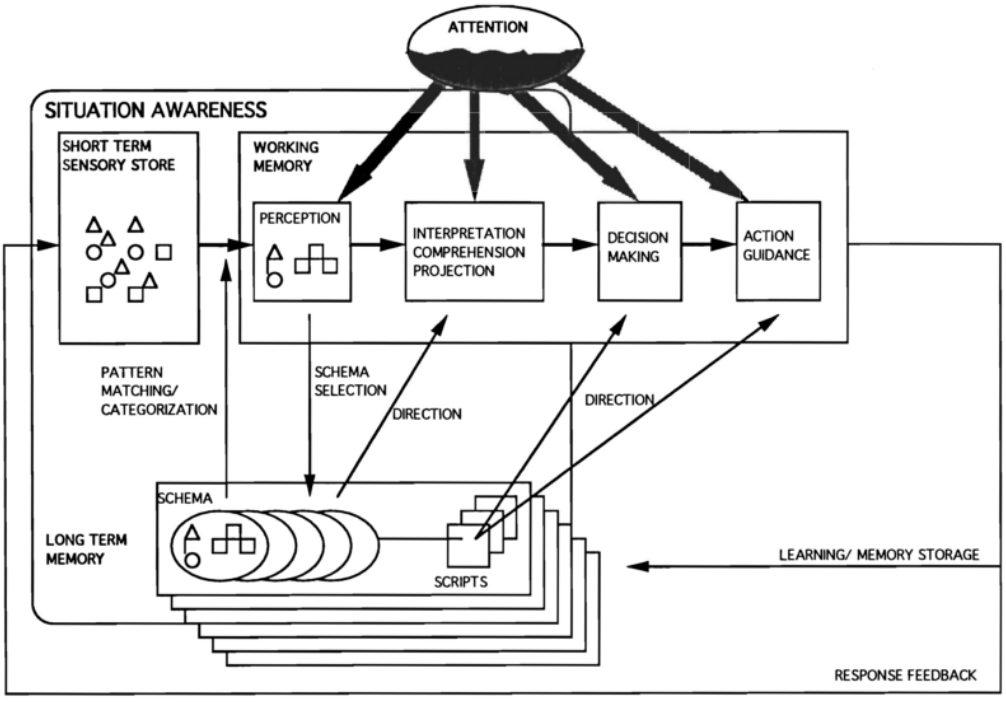


Figure 5: Information-processing mechanisms of SA. Taken from Endsley (1995) [38]

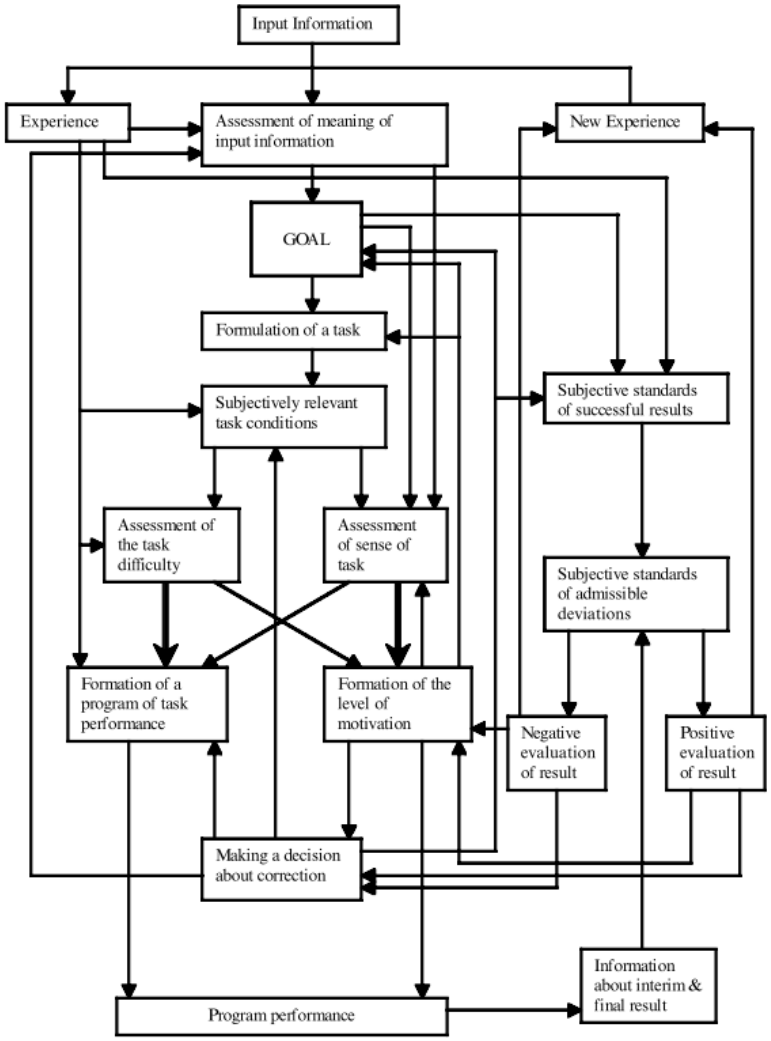


Figure 6: Model of self-regulation of activity. Taken from Bedny et al. (2000) [44]

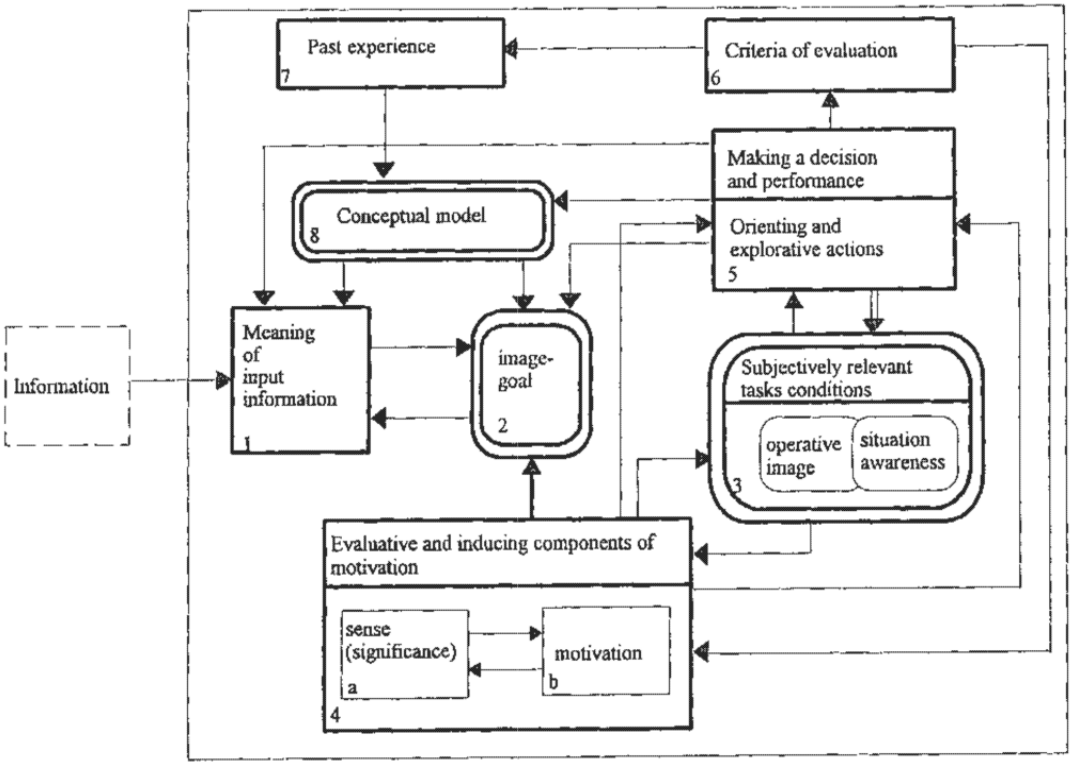


Figure 7: The functional model of orientation activity. Taken from Bedny and Meister (1999) [39]

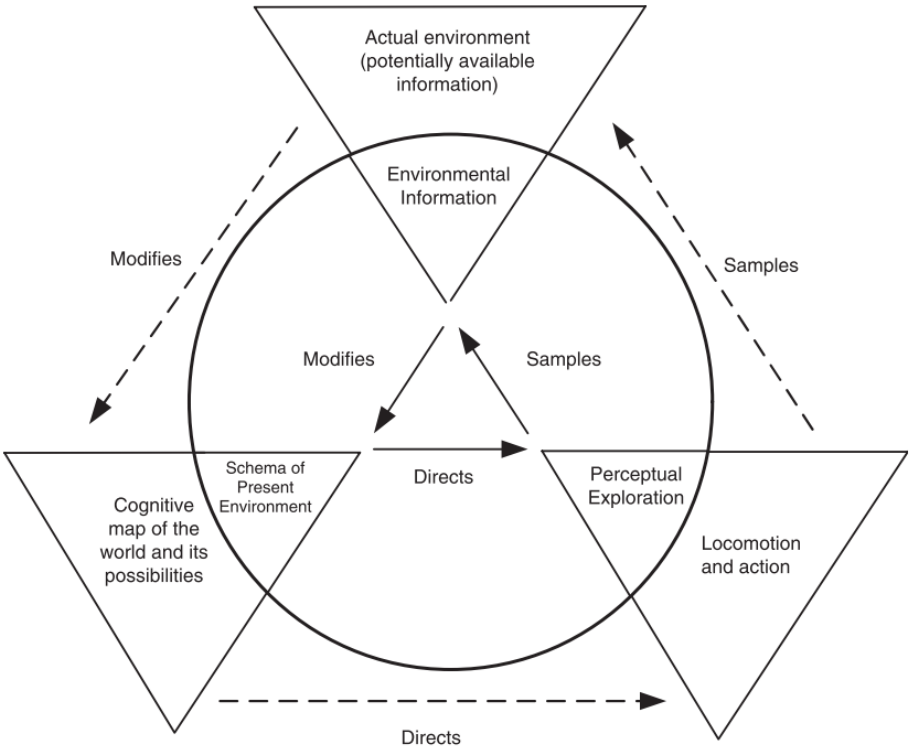


Figure 8: Perceptual Cycle model. Taken from Salmon, Stanton and Walker (2008) [42]

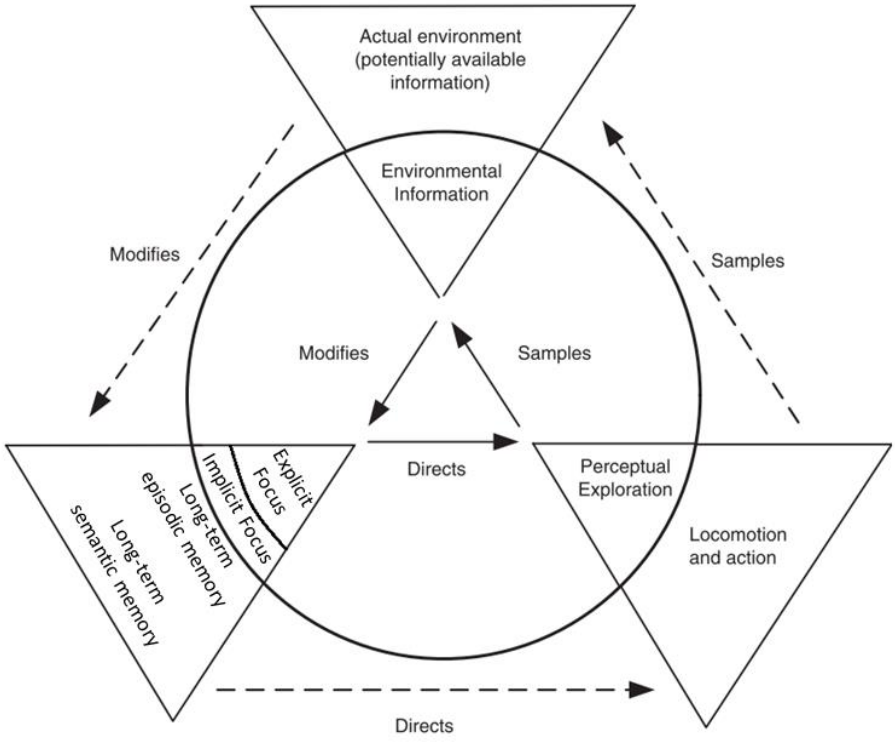


Figure 9: Modified Perceptual Cycle model. Adapted from Salmon, Stanton and Walker (2008) [42]

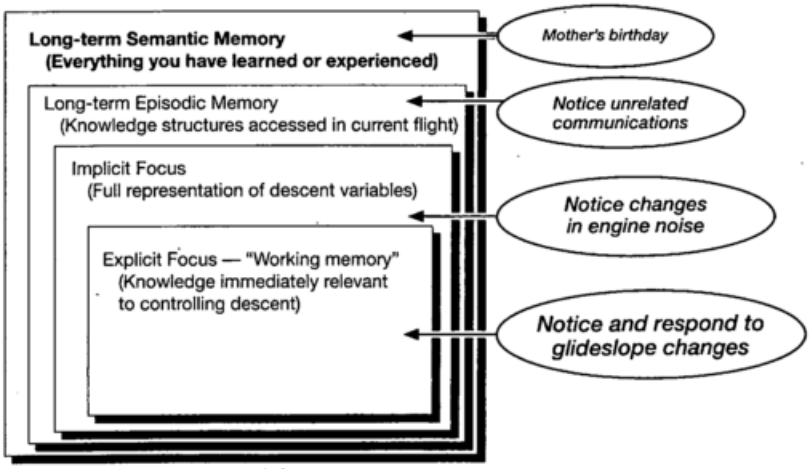


Figure 10: Hypothetical view of the pilot's memory during landing. Taken from Adams, Tenney and Pew (1995) [35]

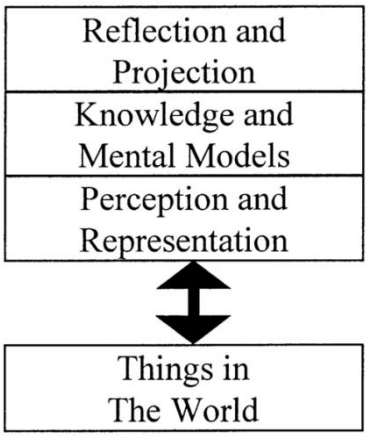


Figure 11: SA definitions integrated into a systems approach. Taken from Stanton, Chambers and Piggot (2001) [36]

Tables

Table 1: General classification of transitions between driver and automation, without taking time-criticality into account. Adapted from Hoeger et al. (2011) [8].

Short Notation	$D_i \rightarrow A$	$D_i \dashrightarrow A$	$D \rightarrow A_i$	$D_i \leftarrow A$	$D \leftarrow A_i$	$D \mid \dashleftarrow A_i$
Direction of transition	Towards automation	Towards automation	Towards automation	Towards driver	Towards driver	Towards driver
Initiation	By driver	By driver	By automation	By driver	By automation	By automation
Status	Successful	Failed / Refused	Successful	Successful	Successful	Failed
Occurrence	Normal driving situation	Situation at automation limits	Normal driving situation	Normal driving situation Situation at automation limits Situation with automation failure	Situation at automation limits	Situation at automation limits Situation with automation failure

Table 2: Situation Awareness comparison table. Adapted from Salmon, Stanton, Walker et al. (2008) [42]

Theory	Main Strengths	Main Weaknesses
Three-level model	1. Simple intuitive description of SA 2. Division of SA into levels is neat and permits measurement 3. Holistic approach that considers factors such as system & interface design, workload and training	1. Fails to cater for the dynamic nature of SA 2. SA process oriented definition is contradictory to the description of SA as a 'product' comprising three levels 3. Based on ill-defined and poorly understood psychological models
Theory of Activity model	1. Model offers a more complete, dynamic description of SA than the three-level model 2. Clear description of each functional block's role in SA acquisition and maintenance is useful 3. SA described as a distinct and separate entity	1. Limited application and the model lacks supporting empirical evidence 2. Underpinning activity theory remains unclear 3. No measurement approach suggested
Perceptual cycle model	1. Dynamic description of SA acquisition, maintenance and update of schema 2. Sound theoretical underpinning 3. Completeness of model is attractive i.e. it describes both the process of acquiring SA and the product of SA	1. Does not translate easily to SA description and measurement 2. Limited applications 3. The actual correlation between SA and performance is complex and not yet fully understood

References

1. B. van Arem, et al. (2007),*"Design and evaluation of an Integrated Full-Range Speed Assistant"*
2. J. Vergeest, B. van Arem (2012), *"The Effect of Vehicle Acceleration near Traffic Congestion Fronts"*
3. A. Kesting et al. (2008), *"Adaptive cruise control design for active congestion avoidance"*.
4. T. Benz et al. (2003), *"Traffic Effects of Driver Assistance Systems – The Approach within INVENT"*
5. C.J.G. van Driel (2007), *"Driver Support in Congestion – An Assessment of User Needs and Impacts on Driver and Traffic Flow"*
6. J.R. Treat et al. (1979), *"Tri-level Study of the Causes of Traffic Accidents. Executive Summary"*
7. N.A. Stanton M.S. Young (2010), *"Vehicle automation and driving performance"*
8. R. Hoeger et al. (2011), *"Highly Automated Vehicles for Intelligent Transport (HAVEit) – The Future of Driving"*
9. N.A. Stanton, P. Marsden (1996), *"From Fly-By-Wire to Drive-By-Wire: Safety Implicatios of Automation in Vehicles"*
10. www.**bmw**.com/com/en/newvehicles/x/x5/.../**traffic_jam_assistant**.html
11. http://singularityhub.com/2013/03/21/bmw-cars-will-be-highly-automated-by-2020-driverless -by-2025-behind-the-curve/
12. http://www.youtube.com/watch?v=HUU7GgBMmY
13. S. M. Young, N. A. Stanton and D. Harris (2007), *"Driving Automation: Learning from Aviation about Design Philosophies"*
14. N. B. Sarter and D. D. Woods (1995), *"How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control"*
15. M. Saffarian, J.C.F. de Winter, R. Happee (2012), *"Automated Driving: Human-Factors Issues and Design Solutions"*, in Proceedings of the Human Factors and Ergonomics Society Annual Meeting 2012, 2012, pp. 2296-2300.
16. D.B. Kaber, M.R. Endsley (1997), *"Out-of-the-Loop Performance Problems and the Use of Intermediate Levels of Automation for Improved Control System Functioning and Safety"*
17. R. Parasaruman, T.B. Sheridan, C.D.A. Wickens (2000), *A Model for Types and Levels of Human Interaction with Automation"*, in IEEE.
18. P. Salmon, N. Stanton, G. Walker and D. Green (2006), *"Situation Awareness Measurement: A review of Applicability for C4i environments"*
19. F. Flemisch, J. Kelsch, C. Löper, A. Schieben and J. Schindler (2008), *"Automation spectrum, inner / outer compatibility and other potentially useful human factors concepts for assistance and automation"*
20. F. Flemisch, J. Kelsch, C. Löper, A. Schieben, J. Schindler and M. Heesen (2008), *"Cooperative Control and Active Interfaces for Vehicle Assistance and Automation"*, FISITA World Automotive Congress, Munich (2008).
21. Martens, Pauwelussen, Schieben, Flemisch, Merat, Jamson and Caci (2011), *"CityMobil: Human Factors' Aspects in Automated and Semi-Automatic Transport Systems: State of the Art."*
22. http://www.imobilitysupport.eu/library/imobility-forum/working-groups/active/automation/ meetings-6/2012-3/1st-meeting-4/1575-auto-wg-presentation-07-mar-2012/file
23. http://www.imobilitysupport.eu/library/imobility-forum/working-groups/active/automation/ workshops-3/07-mar-2013/2015-auto-ws-1-b-trb-joint-subcommittee-07-mar-2013/file
24. A. Toffetti, E.S. Wilschut, M.A. Martens et al. (2009) *"CityMobil: Human Factor Issues Regarding Highly-automated Vehicles on an eLane."*, in Transportation Research Record: Journal of the Transportation Research Board, 2110. 1 - 8.
25. T. Hesse, J. Engström, E. Johansson et al (2011), *"Towards User-Centered Development of Integrated Information, Warning, and Intervention Strategies for Multiple ADAS in the EU Project InteractIVe."*
26. A.P. De Vos, W. Hoekstra (1997), *"Behavioural aspects of Automatic Vehicle Guidance (AVG); Leaving the automated lane"*
27. A. L. Levitan, M. Bunus, W. L. Dewing et al. (1998), *"Preliminary Human Factors Guidelines for Automated Highway Systems Designers (second edition). Volume II: User-System Transactions."*
28. de Winter, J. C. F., et al. Effects of adaptive cruise control and highly automated driving on workload and situation awareness: A review of the empirical evidence. TransportationResearch Part F (2014), http://dx.doi.org/10.1016/j.trf.2014.06.016C.
29. D. Damböck (2013), *"Automationseffekte im Fahrzeuf – von der Reaktion zur Übernahme"*
30. H. Schmidtke (1993), *"Der Leistungsbegriff in der Ergonomie"*, in Ergonomie. Passau: Carl Hanser Verlag München Wien; ISBN 3-446-16440-5.

31. M. Schweigert (2003), *“Fahrerblickverhalten und Nebenaufgaben Dissertation an der Technischen Unversität München.*
32. Gold, D. Damböck, L. Lorenz, K. Bengler (2013), *“”Take over!” How long does it take to get the driver back into the loop?”*, in Proceedings of the Human Factors and Ergonomic Society 57th Annual Meeting, 2013, pp. 1938-1942.
33. I. Petermann-Stock, L. Hackenberg, T. Muhr, Ch. Mergl (2013), *“Wie lange braucht der Fahrer? – Eine Analyse zu Übernahmezeiten aus verschiedenen Nebentätigkeiten während einer hochautomatisierten Staufahrt”*
34. A.P. van den Beukel and M.C. van der Voort (2013), *“The Influence of Time-criticality on Situation Awareness when Retrieving Human Control after Automated Driving”*
35. M.J. Adams, Y.J. Tenney, R.W. Pew (1995), *“Situation awareness and the cognitive management of complex-systems”*, in Human Factors 37 (1), pp. 85-104.
36. N.A. Stanton, P.R.G. Chambers, J. Piggott (2001), *“Situational awareness and safety”*, in Safety Science 39 (2001), pp. 189-204.
37. M. Endsley (1988), *“Design and evaluation for situational awareness enhancement”*, in Proceedings of the Human Factors Society 32nd Annual Meeting. HFES, Santa Monica, pp. 97-101.
38. M.R. Endsley (1995), *“Toward a theory of situation awareness in dynamic-systems”*, in Human Factors 37 (1), pp. 32-64.
39. G.Z. Bedny, D. Meister (1999), *"Theory of Activity and Situation Awareness"*, in International Journal of Cognitive Ergonomics 1999, 3(1), pp. 63-72.
40. K. Smith, P.A. Hancock (1995), *“Situaion awareness is adaptive, externally directed consciousness”*, in Human Factors 37 (1), pp. 137-148.
41. M.R. Endsley (2000), *"Theoretical Underpinnings of Situation Awareness: A Critical Review"*
42. P.M. Salmon, N.A. Stanton, G.H. Walker et al. (2008), *"What really is going on? Review of situation awareness models for individuals and teams"*, in Theoretical Issues in Ergonomics Science Vol. 9, No4, July-August 2008, pp. 297-323.
43. K. Kuutti (1996), "Activity Theory as a potential framework for human-computer interaction research"
44. G.Z. Bedny, M.H. Seglin, D. Meister (2000), *"Activity theory: History, research and application"*, in Theoretical Issues in Ergonomics Science, 1:2, pp. 168-206, DOI: 10.1080/14639220050171324.
45. G.Z. Bedny, W. Karwowski, O.J. Jeng (2003), *"Concept of Orienting Activity and Situation Awareness"*
46. G.Z. Bedny, W. Karwowski, M. Bedny (2001), *"The Principle of Cognition and Behavior: Implications of Activity Theory for the Study of Human Work"*
47. U. Neisser (1976), *“Cognition and reality: Principles and implications of cognitive psychology”*
48. M.J. Adams, U.J. Tenney, R.W. Pew (1995), *“Situation Awareness and the Cognitive Management of Complex Systems”*, in Human Factors 37 (1), pp. 85-104.
49. A.J. Sanford, S.C. Garrod (1981), *“Understanding written language”*
50. P.M. Salmon, N.A. Stanton, G.H. Walker, D. Jenkins, D. Ladva, L. Rafferty, M. Young (2009), *“Measuring Situation Awareness in complex systems: Comparison of measures study”*, in International Journal of Industial Ergonomics 39 (2009), pp. 490-500.
51. M.R. Endsley (1995), *“Measurement of situation awareness in dynamic systems”*, in Human Factor 37 (1), pp. 65-84.
52. F.T. Durso, C.A. Hackworth, T. Truitt, J. Crutchfield, C.A. Manning (1998), *“Situation Awareness as a predictor of performance in en route air traffic controllers”*, in Air Traffic Quarterly 6, pp. 1-20.
53. R.M. Taylor (1990), *“Situational Awareness Rating Technique (SART): the development of a tool for aircrew systems design (AGARD-CP-478) pp3/1 -3/17”*, in Situational Awareness in Aerospace Operations. NATO-AGARD, Neuilly Sur Seine, France.
54. L.J. Gugerty (1997), *“Situation Awareness During Driving: Explicit and Implicit Knowledge in Dynamic Spatial Memory,”* in Journal of Experimental Psychology: Applied 1997, Vol. 3, No. 1, pp. 42-66.