Communication between Automated Vehicles and Pedestrians:
A standalone external Human Machine Interface

Bachelor Final Project 2019 - Mechanical Engineering TU Delft

Carolina van Oeveren 4591046
Justine Kroese 4547659
Laudy Brockhoff 4492528
Rick van Genderen 4445783

November 19, 2020

Abstract

One of the major challenges that automated vehicles (AVs) are facing today is driving in an urban environment. In such circumstances AVs must be able to communicate with vulnerable road users like pedestrians. To solve this problem, external Human Machine Interfaces (eHMIs) are being developed. Previous research has failed to make a standalone device applicable for every car brand.

The main question of this work is: How accurately can a standalone device communicate the speed and acceleration of an AV towards a pedestrian in comparison to an integrated car system? To answer this research question a prototype is made. Thereafter several experimental are executed to calculate the accuracy and reliability of the sensors used in the device.

The results show that the accuracy and reliability of the total device are satisfactory when driving at constant speed or when accelerating. By combining a GPS and an accelerometer the system can give an output that the car is going to stop within 0.15 seconds after starting to decelerate. Moreover, the eHMI is able to give an output within 0.1 seconds after it starts to accelerate. However, when standing still, the delay created by the GPS is too large which results in delayed output that the car has stopped of 0.688 ± 0.282 seconds when using both sensors.

This work is a contribution to the development of a standalone device to support future research in the field of human factors in the autonomous driving world and it facilitates the development of a universal AV-pedestrian communication.

This report is commissioned by the Cognitive Robotics department of Technical University Delft.

TU Delft

1 Introduction

At the moment, there are about 1.4 billion cars on the road [11]. In the course of time manually-driven cars will be replaced by self-driving vehicles [31]. Hyundai and Renault-Nissan, for example expect that, between 2025 and 2030, fully autonomous driving will be possible in an urban environment [9].

One of the major challenges of automated vehicles (AVs) is driving in such an environment. Then AVs have to be able to communicate not only with other (automated) vehicles, but also with vulnerable road users (VRUs) such as cyclists and pedestrians [19],[15]. Right now, the communication between VRUs and human drivers of non-automated vehicles often relies on hand gestures and other signals [26],[23] such as waving a hand or making eye contact [18]. Future AVs might not contain a human driver and even if they do so the driver might be occupied with non-driving tasks and not pay attention to the road. Human-human interaction will not be possible [22]. Consequently, it is important to find alternative ways for AVs to communicate with VRUs.

A possible solution for AV-VRU communication could be an external human-machine interface (eHMI) [3]. In the case of an eHMI, the AV measures its own state and the state of the environment (input) and communicates advice or its own intention to the VRU (output). Several eHMIs are currently being developed worldwide. However, these typically rely on the instrumentation and software interfaces within the car for inputting and outputting information. Therefore, they are often limited to a specific car brand [29],[5]. The usage of such car-internal systems makes the software usually complicated [2]. Moreover car-specific eHMIs are also inconvenient for investigating human behaviour in response to AVs, because such eHMIs cannot be easily shared between research groups using different cars.

To solve the problems mentioned above, there is need for eHMIs that are generic and independent of the car. Accordingly, the aim of this project is to create a standalone eHMI that is compatible with all AVs. The corresponding software will be shared open source. In this way, the eHMI could be implementable in different car brands and thus utilised in future experiments in the field of human factors as well as allow AVs on the road to operate using this same software. Hereby AVs will be able to communicate with pedestrians in a consistent manner, which could prevent confusion, and thus increase road safety [28].

The main question that is aimed to be answered in this work is: How accurately and reliably can a standalone device communicate the speed and acceleration of an AV towards a pedestrian in comparison to an integrated car system?

The first part of the paper provides a brief literature review on the possibilities of the eHMI output and corresponding input needed to generate these outputs. Next, design requirements for the standalone eHMI are formulated and used to create a series of conceptual designs. By using a decision matrix, the most promising concept is chosen and worked out into a detailed design. A prototype is built and quantitatively evaluated. The paper ends with a reflection on the performance of the prototype and recommendations for future improvements.

2 Literature research

2.1 Input

Many different sensors are utilised to provide input to an eHMI. Some of these sensors provide information about the state and actions of the AV itself and others provide information about the environment.

2.1.1 Input from the AV itself

A wide variety of sensors have been developed to provide input to eHMIs regarding the speed and location of an AV. The WEpod [4], for example, uses two GPS sensors on the roof of the vehicle to determine its absolute position. Furthermore, the WEpod is equipped with an Internal Navigation sensor, which measures the velocity of the vehicle in all directions, and odometers that register the rotations of the wheels that help determining the location of the vehicle. Kato et al.[16] used Javad RTK sensors to receive global positioning information from satellites. These sensors are often coupled with gyro-sensors and odometers to fix the positioning information. Sukkarieh et al. [30] developed a navigation system combining a GPS and an inertial measurement unit (IMU).

2.1.2 Input from environment

Data from the environment can be obtained through different sensors which can be optionally combined. Sandt and Owens [28], for example, proposed the use of a lidar, a machine vision system that uses cameras or a V2X beacon, which is a system that connects pedestrians, cyclists and other vehicles wirelessly to the AV. Likewise, Kotseruba [17] stated that it is possible to use pedestrians phones to broadcast their position, warning the AV that a pedestrian is about to cross the street. Alternatively, Pennycooke [27] proposed using the Microsoft Kinect sensor to detect pedestrians. This sensor uses a RGB camera, an infrared dot-pattern blaster and an infrared camera to see its surroundings. Furthermore, Pennycooke used sonar proximity sensors to define the distance to an object. Moreover, the WEpod [4] uses among a wide array of other sensors, automotive radars as primary detection system and ultrasonic and lbeo lidars for additional input.

2.2 Output

Humans are able to estimate whether a car is going to stop based on the deceleration of the car, but the accuracy of this assessment can vary depending on various parameters such as the speed of the car and the light intensity [1]. Previous research has shown that the majority of pedestrians find external interfaces useful on top of solely observing the vehicle movement [23], [20]. Three types of output can be distinguished: output announcing the intentions of the AV, output that advises the pedestrian what to do and output that shows the AV's recognition of the pedestrian by the AV [10].

2.2.1 Output announcing the intentions of the AV

One way for an eHMI to interact with pedestrians is to communicate the intentions of the AV, instead of telling to pedestrians what to do. Habibovic and Klingegard [13] designed a projection of a line on the road where the car is going to stop and different LED outputs, such as LEDs in the grill of the car simulating the airflow when the car is moving. Similarly, Ford [29] designed a LED strip that shows slow pulses of white lights moving back and forth if the AV is yielding and rapidly blinking if it is accelerating. Clamann et al. [7] also designed a LED display, mounted on the front of the AV, which displays the speed of the vehicle. Alternatively, De Clercq [8] proposed a front brake light. Next to visual output, auditory output has also been previously considered. Mahadevan [21], for example, tested a speaker mounted on the vehicle that played the messages "about to stop" and "about to start".

2.2.2 Output advising pedestrians

Several ways have been proposed in which an AV can advise pedestrians what to do when the vehicle is approaching. Drive.ai [25] and De Clercq [8] suggested using a display that shows "Walk" when the pedestrian is safe to cross or "Don't walk" if the pedestrian should wait. Next to textual messages, Drive.ai also utilised an icon of a crossing pedestrian. Likewise Clamann et al. [7] made use of an Ampelmann icon. Another approach of communicating a recommendation is by displaying a projection on the road. Mercedes-Benz [24] has chosen to project a crosswalk on the street to indicate to the pedestrian that it is safe to cross. The company has also presented eHMIs that convey advisory auditory messages.

2.2.3 Output showing recognition of pedestrians

Apart from communicating a message to a group of pedestrians as a whole, eHMIs targeting a single person have also been considered. Here the aspect of pedestrian recognition comes into play. Chang et al. [6] proposed an eHMI with eyes on a screen that follow the pedestrian while crossing. In a similar way, Pennycooke [27] used the headlights to follow the pedestrian similar to how the eyes of a human driver would. Graziano [12] presented a LED strip all around the AV, with a light indication pointing towards the pedestrian and following the pedestrian across the road. There is also the possibility of providing haptic feedback to the pedestrians through wearable devices, such as the vibration of phones or smartwatches. The wearable device would vibrate to indicate that it is safe to cross or a spoken message or sound could be sent to the phone of the pedestrian [21].

3 Design

This section describes the design process of the eHMI prototype. First, the requirements are presented and motivated. Thereafter, by using a decision matrix, a detailed design for the prototype is developed.

3.1 List of requirements

The design requirements are drafted based on the main question of the paper as stated in the introduction: How accurately and reliably can a standalone device communicate the speed and acceleration of an AV towards a pedestrian in comparison to an integrated system? The requirements are divided into two categories: design and performance. The design requirements are fulfilled by making particular assembly choices in the prototype. The performance requirements will be experimentally validated (see section 4).

Design

- *The system is a standalone device.* As explained in the introduction, the benefit of a standalone device is that it can be utilised for human factor research and that it can provide a universal and clear message to pedestrians when necessary regardless of the car brand.
- *The system should be applicable for every car.* This requirement is of great essence because there is not much added value from being standalone if the system is implementable only to one car brand.

Performance

- *The system should give an output that the car is going to stop (e.g. "Decelerating" or a front brake light) within 0.15 seconds after starting to decelerate.* The choice of a maximum 0.15 seconds delay is associated with gap acceptance. Matthias Begiato et al. (2018) defined gap acceptance as: "...the decision of persons on the last moment at which they would safely cross a street before an oncoming vehicle." The time gap before the car reaches the pedestrian

at 30 km/h (i.e., the maximum speed for residential areas), is about 1.53 seconds. The chosen maximum delay of 0.15 s corresponds to one-tenth of the smallest time gap.

- *The system should give a different output indicating that the car has stopped (e.g. "I've stopped" or "Walk") within 0.5 seconds after stopping.* We allowed the accepted delay for the change of the output when the car has stopped to be larger than 0.15 seconds. This is because this output is less safety critical than the output concerning the future action of the AV described in the first requirement.
- *The system should stop giving an output within 0.1 seconds after it started to accelerate.* It is important that the car does not give an output when it is accelerating, especially if it is driving off at a pedestrian crossing. We calculated the maximum delay in displaying the output by using the average zero-to-sixty miles per hour acceleration of a car, which corresponds to 3-4 m/s^2 [14]. Assuming that a car stops 2 meters in front of a zebra crossing and that the acceleration is constant, the minimum time until the car is at the zebra crossing is 1 second. We set the maximum acceptable delay to 0.1 seconds, which is one-tenth of the minimal time.

3.2 Conceptual Design

To make a choice for the conceptual design, we considered several different solutions. First, possible inputs were set side-by-side. Thereafter, we looked at advised outputs according to literature. Lastly, various standalone systems were compared. A decision matrix was applied to choose a final design.

3.2.1 Input

To measure the speed and acceleration without using car data, two types of sensor are often mentioned in literature: a GPS-module ([4],[30]) and an accelerometer or IMU [30]. therefore, we set the requirement that these sensors should be present in the device. Whether the GPS-module, the accelerometer or a combination of the two is the most accurate detector will be tested and evaluated in section 4 and section 5.

To detect whether the AV is stopping for a pedestrian, and therefore should give an output, a detection sensor is needed. According to Sandt and Owens [28] lidar has very precise data mapping, but the current lidar hardware is expensive. therefore the issue has been looked at from a different angle. Instead of detecting a person, it can be detected whether a car is in front of you or not by using three ultrasonic sensors. These three sensors are placed at the front side of the car: one on the left, one on the right and one in the middle, all facing forward. When there is a car in front of all three sensors, they will measure about the same value (using a margin, to compensate for the fact that the cars might not be perfectly aligned). On the road the system would work as following: when the vehicle is decelerating and stopping and there is no car in front of it, there is a fair chance that the AV is stopping for a pedestrian. On the other hand, if there is a car in front of the AV when the AV decelerates, it is at least not the first car to stop for a pedestrian, so there is no use to give an output.

3.2.2 Output

Based on the output options presented in literature (see subsection 2.2), we selected several possible outputs for the eHMI. The options that we came across most often were either the usage of a screen to present an icon [25],[7] or text [25],[8], or LEDs that simulate the movements of the car [13],[29], therefore we decided that these needed to be present or attachable in the final prototype.

3.2.3 Hardware

To be able to process these inputs and outputs, hardware is needed. For a standalone device a few possibilities exist. Here we considered three types of tablets and an Arduino. These devices were examined, because they are affordable and standalone devices, which incorporate a GPS and an IMU. In case of the tablets they also have a display. The tablets considered are from Apple, Microsoft and Samsung (IOS, Windows and Android). From these brands, we chose the tablets with the largest screen size, because the eHMI should give an output that is visible from some distance [8]. The Arduino was considered, because many different sensors and output possibilities can be added to the board which makes the system easily adjustable and versatile.

3.2.4 Decision matrix

In this section, hardware solutions are compared on the ground of the requirements stated in subsection 3.1 as well as on the selected inputs and outputs stated in subsubsection 3.2.1 and subsubsection 3.2.2. This analysis is shown in Figure 1.

Each hardware solution is rated based on how easily the the design requirements can be fulfilled with 0, 1 or 2.

0 corresponds to not available, 1 correlates to compatible with the specific system and 2 means easily applicable for the system. Moreover, we set a weight factor for each requirement depending on its importance for our project.

		Weighting	Tablet			Arduino	ROS
			IOS IPad Pro	Windows Microsoft	Android Samsung		
Design	Standalone device	2	2	2	2	2	0
	Universally applicable	1	2	2	2	2	0
Input	Accelerometer	2	2	2	2	2	2
	GPS	2	2	2	2	2	2
	Ultrasonic sensor	1	0	0	0	2	2
Output	LEDs	2	0	0	0	2	2
	Screen	2	1	1	2	1	0
Total		100%	67%	67%	75%	92%	58%

Figure 1: Decision matrix

The matrix, shown in Figure 1, starts off with design necessities. These factors are essential to have a result that can be implementable in different car brands and thus usable in future experiments in the field of human factors. We have taken a look at whether the hardware could be a standalone device and whether the system is applicable to every car brand.

After the design goals, the necessary input sensors are being evaluated for each system. The weight factor is based on the essentiality of the sensor to build a working device while living up to the performance requirements.

In the last section of the matrix, the possibility to display the chosen outputs are assessed. As mentioned earlier the options for visual feedback, that were taken into consideration for the conceptual design are conveying a message using LED's or a screen displaying an icon or a message. The hardware solutions concerning the output are scored based on the following elements: the size, the visibility in all weather conditions and the flexibility in output options.

Based on the decision matrix shown in Figure 1 the Arduino was chosen as the hardware solution, because it scored the highest for both the design requirements and the requirements set in subsubsection 3.2.1 and subsubsection 3.2.2.

3.3 Prototype

A prototype of an eHMI is developed and experimentally evaluated. The input, output and hardware will be elaborately described in the following section.

3.3.1 Input

In the prototype the sensors specified in Figure 2 are used.

		GPS module **Gp-20u7**	Accelerometer **LSM6DS3**	Water sensor **Grove water sensor**	Ultrasonic sensor **HC-SR04**	Light sensor **Grove Sunlight Sensor**
Manufacturer		ADH-tech	STMicroelectronics	SEEED	OSEPP electronics	SEEED
Accuracy	Update rate	1 Hz	104 Hz	*	40 Hz	*
	Location	50% in 2.5m	*	*	3 mm	*
Sensitivity/range		*	1/16284 mg/LSB	*	Range: 2cm-4m	*

Figure 2: Specifications of GPS modules, accelerometer and extra sensors

The wiring of the GPS-module and accelerometer are displayed in Figure 3 and Figure 4 respectively.

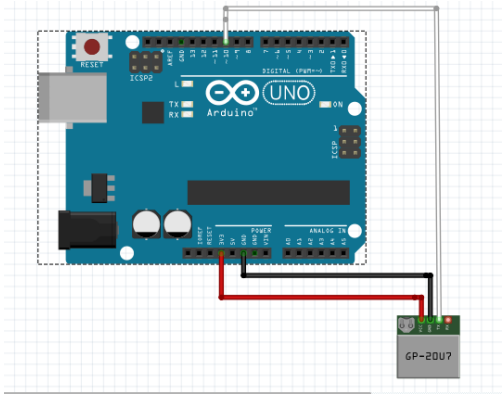

Figure 3: Circuit GPS

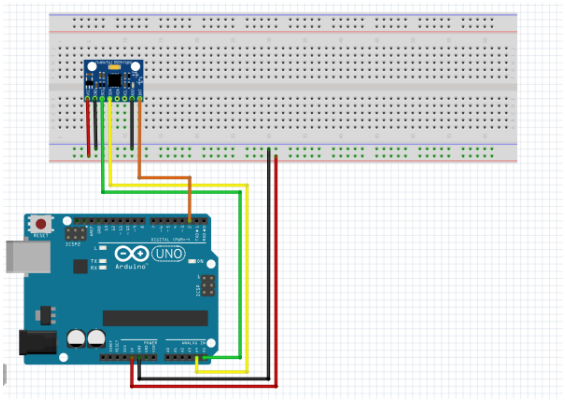

Figure 4: Circuit Accelerometer

The three ultrasonic sensors that are used (Figure 6) are not waterproof, so we designed cases that were 3D printed (Figure 5). So they can still be used on a rainy day.

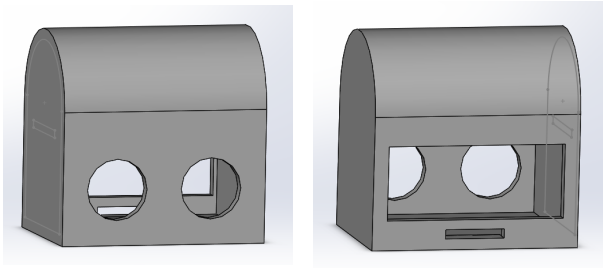

Figure 5: Solidworks design ultrasonic case

Additionally, a water and light sensor are implemented (Figure 6). These are not necessary to achieve a working system, but they do have an effect on the amount of energy the LED strip uses. Both sensors are used to alter the brightness of the output so that it is not disturbing at night and is visible in various weather circumstances.

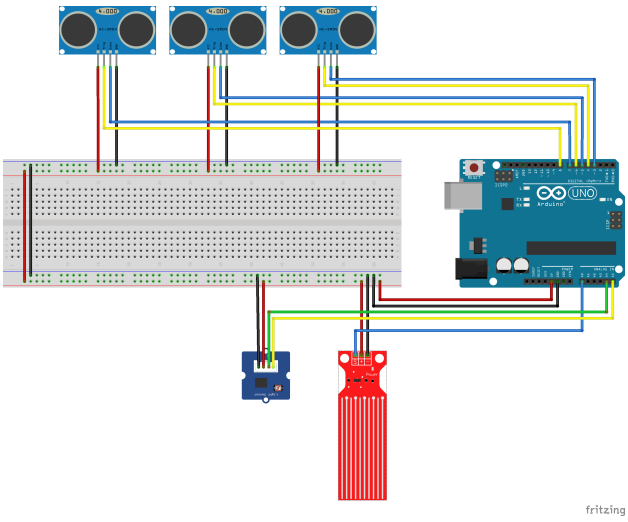

Figure 6: Circuit Combination Ultrasonic sensors and Water and Light sensor

3.3.2 Output

At the Cognitive Robotics department, a LED strip with Arduino has already been built and tested by MSc students M. Barendse and R. Agarwal (Figure 7).

The entire strip consists of 12 LED strips of 1 meter mounted below each other. It makes use of WS2812 digital RGB LED strips with 60 LEDs p/m. Moreover, a LED strip can draw a lot of current (60 mA per LED unit, $12 strips x 60 LEDs x 60 mA = 43.2 A$) [32] from an Arduino, therefore an Arduino Mega is used in combination with an external power supply of 5V, which is connected to the electricity grid. The strips are programmed with the Neopixel library from Adafruit. Herewith, the LED strip can be programmed to give different outputs, such as text, colour and symbols/icons.

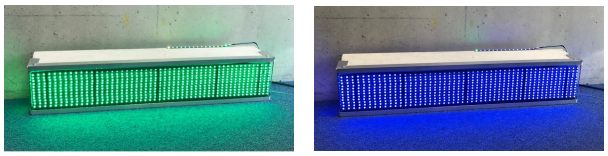

Figure 7: LED strip with green and blue output

The messages we want to convey are the following three: 1. No output 2. Decelerating output 3. Stopping output. These three different outputs follow from the performance requirements.

3.3.3 Hardware

The hardware that is part of the prototype consists of multiple Arduino Uno boards. These are used to control the input sensors. The GPS-module and the accelerometer are mounted on two different Arduino Unos, because the sampling rate for the GPS-module (1 Hz) and the accelerometer (104 Hz) is different. The Arduino Uno boards are connected to a power bank to have a wireless power supply. To protect the sensors and boards and to make sure the wires will not loosen by accident, we designed a case shown in Figure 8.

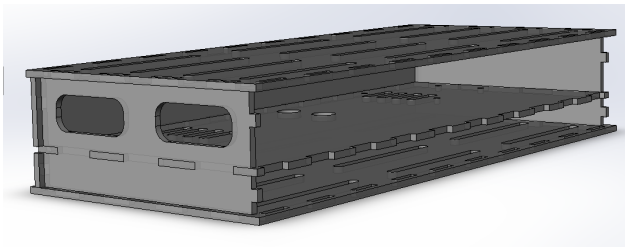

Figure 8: Designed case

One of the sides has been removed to give more insight in the design within. In the bottom layer the power bank can be stored. In the top layer the Arduino boards and breadboard can be secured. The holes in the layer in between are for the cable from the power bank to the Arduino board. The holes on the sides are to be used for external cables that have to be connected, for example, to the ultrasonic sensors on the car. The case fits at the back of the LED strip, so it will automatically have extra protection against unfavourable weather conditions, such as rain.

The LED strip is controlled by an Arduino MEGA already installed in construction of the LED strip.

4 Experimental evaluation

An experiment was conducted to evaluate the prototype. In the experiment, an instrumented car was used in order to determine the accuracy and reliability of the GPS sensor and accelerometer of the prototype. Specifically, we investigated the performance of these sensors in measuring the speed, acceleration and deceleration of the AV, as well as in identifying whether the AV is standing still.

4.1 Methods

4.1.1 Experiment: Investigating the accuracy of the sensory

In this experiment the accelerometer and GPS of the prototype were compared with the highly accurate and precise sensors of an instrumented car, the Toyota Prius from the 3mE faculty (Figure 9). The data from the ROS (Robot Operating System) of this car will be considered as the ground truth.

Figure 9: Toyota testing car

The first test was conducted with the aim of assessing the performance of the eHMI sensors when driving at constant speeds. In this test, the car started from rest and then accelerated up to 10 km/h. Then the driver tried to keep driving at this speed for 30 seconds. Thereafter, the car accelerated to a speed around 20 km/h and kept this pace for about 30 seconds. At last, the car accelerated to around 30 km/h and stayed at this speed for approximately 30 seconds as well. Then the car stopped, turned around and accelerated to 30 km/h to drive back to the starting point. This test was conducted seven times.

A second test was conducted to assess the performance of the eHMI sensors in measuring accelerations. This test consists of two parts. In the first part the car started from rest, then it accelerated until 10km/h and stayed at this speed for a few seconds. Next, the car accelerated to 20km/h and again stayed at this speed for a few seconds. Similarly the car accelerated to 30km/h. Hereafter the car decelerated likewise until coming to a standstill. This test was repeated twice. For the second part of the acceleration test, the car accelerated to 10 km/h and then decelerated to a standstill once, to 20 km/h and to a standstill four times and four times to 30 km/h and back to a standstill.

The data of the GPS and the accelerometer of the eHMI were compared to the sensor data of the ROS of the car. With these calculations the accuracy and reliability of the GPS module was estimated.

4.1.2 Assessment criteria

The performance of the sensors can be assessed by computing the errors. Before computing, the data is distinguished in two different situations: driving at a constant speed and accelerating/decelerating. This is done to get a clear impression if the accuracy and reliability of the sensors differ in the various situations. All intervals for the constant speed are added together to get the maximum available data. From this data the *mean error, root mean squared error, mean squared error* and *standard deviation* are calculated for both the GPS-module and the accelerometer. The same is done for the accelerating/decelerating intervals. The *Root Mean Squared Error* is a measurement of the errors, where the square root of the squared mean is taken. The *mean squared error* makes the large outliers more significant in comparison with smaller outliers. This is computed, because large outliers might mess with giving a clear and non-flickering output. Next to these computations, the *response times* or *delays* of the GPS-module is analysed, which is of great importance to determine the reliability of the sensor. This is done by determining 10 peaks from the car data and comparing these with their associated peaks from the data of the sensor. In this way their average delay is calculated. At last, the unitless *mean absolute percentage error* is used, for comparing the two sensors, the GPS-module and the accelerometer, which produce data with different units. In Equation 1, the MAPE is calculated.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{1}$$

Where,
- A_t = Data from the car
- F_t = Data from a sensor

4.1.3 Data processing

The GPS data from the car and from the GPS module can be easily compared due to the time stamp that comes with every time step. To be able to compare the data, the data is first saved in arrays and are made equal in length, using interpolation. When both arrays are of equal length, the differences between the measured GPS speed and the real car speed are calculated

and with these differences the various errors are computed.

The data obtained from the accelerometer is not precisely in the same time frame as the data from the accelerometer of the car, so the data has to be shifted to the same time frame. This can be done by comparing a fixed point that is identifiable in all sensors, for example a bump in the road that gives an identifiable peak in the z direction of both data plots.

As displayed in Figure 12, the raw data of the accelerometer contains a significant noise. Three types of often used filters were implemented and compared. The first filter used, is a Butterworth filter. The cutoff frequency for this filter is determined by plotting the magnitude bode plot of the Fourier transform of the raw data and determine where it stays around zero, from that point on, the data is not useful anymore, because it only contains noise. We found that this point was around 7 Hz. The sampling frequency (f_s) is 104 Hz. The normalised cutoff frequency is than calculated by:

$$W_n = \frac{f_c}{f_s/2}$$

A third order Butterworth filter is used, since the acceleration data of the car is a moving mass, which is a third order system. The second filter implemented is a moving average filter with a sliding window. To determine what window length should be used the data is filtered with all window lengths from 2 to 20 in steps of 2. The mean absolute errors of these filters are calculated and plotted in Figure 10, a clear trend is visible. The optimal window length is the smallest window length with the lowest mean absolute error, since using a larger window size does not reduce the error any further and a larger window generates a larger delay. As can be seen in Figure 10, the optimal window length is 12.

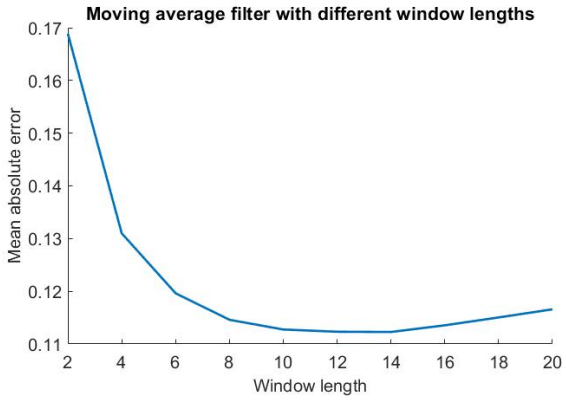

Figure 10: Mean absolute error after filtering with different window lengths for a moving average filter

The last filter tested is also a moving average filter, but based on exponential weighting. Then the optimal forgetting factor has to be determined. The forgetting factor causes the value that has to be determined to rely on the past samples, but with a decreasing factor, so that older samples are slowly "forgotten". To get the optimal factor for this exponential weighting, the same method is used as for the window length. The root mean squared errors of moving average filters with forgetting factors between 0.5 and 1 (maximum value) are shown in Figure 11.

Filter	Butterworth	Moving average sliding	Moving average exponential
RMSE	0.1818	0.1795	0.1780

Table 1: Root mean squared errors for different filters

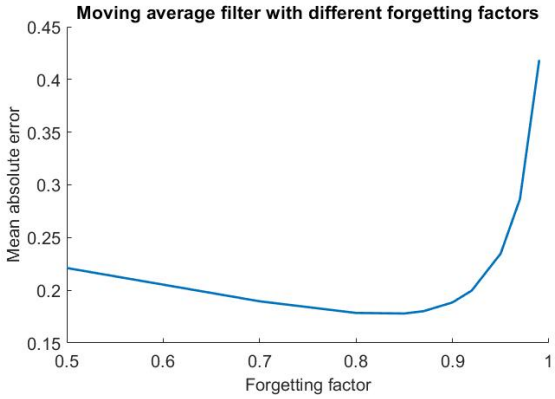

Figure 11: Mean absolute error after filtering with different forgetting factors for a moving average filter

Figure 12 shows the data of the accelerometer after filtering with the third order Butterworth filter and the two moving average filters. After testing and comparing different window lengths and forgetting factors, the sliding window length has been set to 12 and the forgetting factor to 0.85, since these resulted in the most accurate filtered data. The root mean squared errors (RMSE's) of the filters have been calculated and are shown in Table 1. From these RMSEs it is derived that the moving average filter based on exponential weighting with a forgetting factor of 0.85 is best suited to filter the accelerometer data. This filter will therefore be used for processing the data.

The next step in data processing is about making a decision whether or not to give output. In order to determine the output for the system, we designed three different outputs: standing still, decelerating and a case with no output. For the different outputs to take place, several conditions have to be met. For standing still there are two conditions: the speed of the GPS has to be close to zero m/s and the accelerometer should measure an acceleration that is close to zero as well. The limits were not set precisely at zero to include for the errors in the GPS-module and accelerometer.

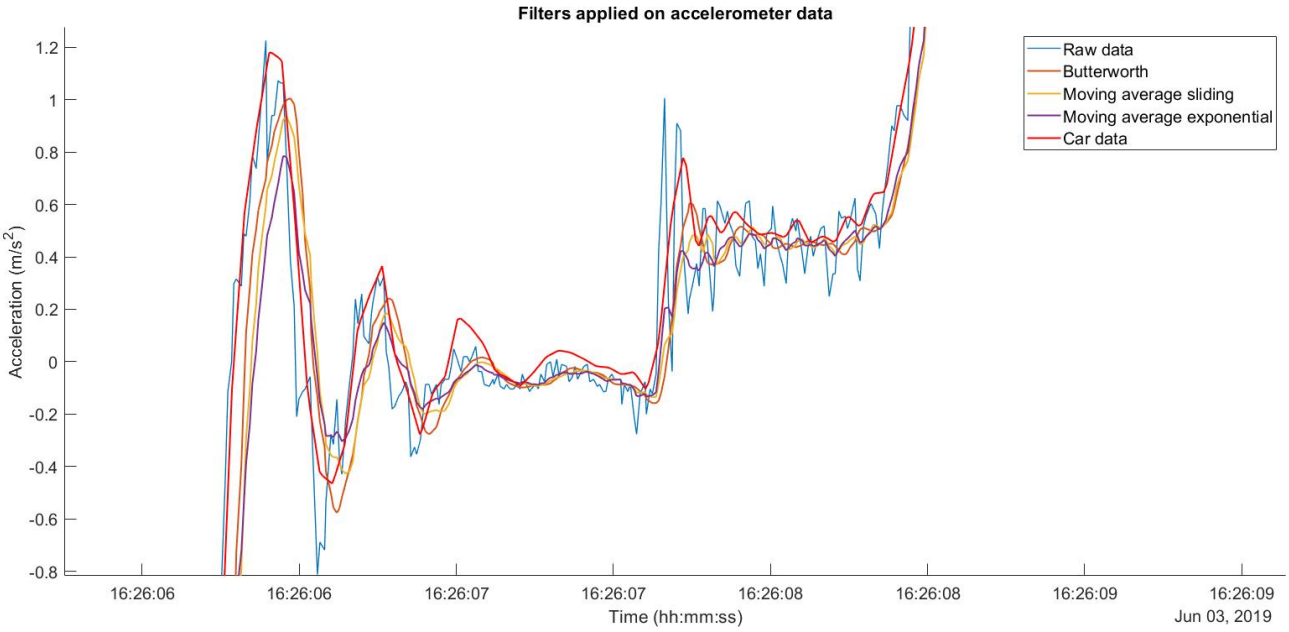

Figure 12: Accelerometer data after applying different filters

For the device to give the second output, that the car is decelerating, there are several criteria that have to be met. In the following pseudo code the first steps in the process are described algorithm 1:

if *the speed > 8.33 m/s* **then**
  Produce No Output;
else
  if *GPS measures deceleration* **then**
    Produce Output;
  else
    Produce No Output;
  end
end

Algorithm 1: Example of decision code for only GPS-module

This code was programmed in MatLab. The results of this programming are displayed in Figure 17. In this figure the speed of the car (blue dashed line) and the measured speed (red line) are displayed. As displayed in the graph, the blue line (car data) is almost equal to the red line (GPS data), but the red line reaches each point a little later. This is the delay of the sensor. In order to get rid of this delay, an accelerometer is added to the process. This is because the update frequency of the accelerometer is higher then the update frequency from the GPS module. Hereby, a deceleration or acceleration can be detected sooner. Including the accelerometer in the decision process brings an extra step in the pseudo code:

if *the speed < 1 m/s and the mean acc < 0.2 m/s^2* **then**
  Produce Stopping Output;
else
  if *the speed > 8.33 m/s* **then**
    Produce No Output;
  else
    if *the device measures < 0 m/s^2* **then**
      Produce Decelerating Output;
    else
      Produce No Output;
    end
  end
end

Algorithm 2: Pseudo code of combination of GPS-module and accelerometer

4.2 Results

4.2.1 Data analysis of the sensors

In this section, the results of the data analysis of the different sensors are presented. By looking at the them separately, an analysis of the total performance of the system as a whole is made.

A. The GPS-module

A graph of a test round is shown below in Figure 13. This round consisted of driving at a constant speed of 10km/h, 20km/h and 30km/h and at the end, one acceleration from 0km/h to 30km/h. Between driving constant at 20km/h and 30km/h, the car had to make some manoeuvres to turn around and stay on the testing grounds (at around 16:17:30).

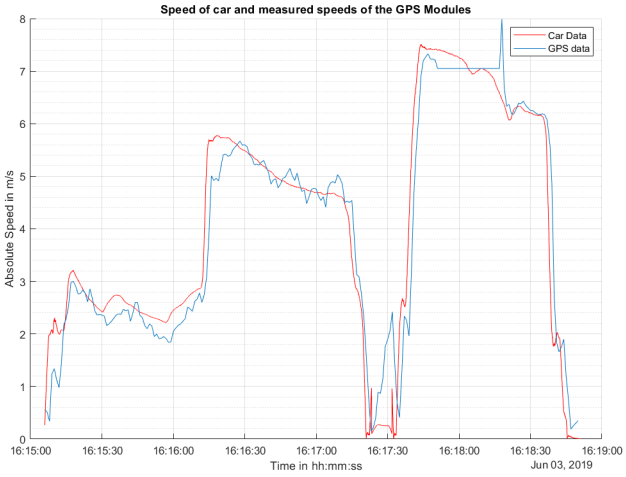

Figure 13: Speed from car and GPS-module

In the graph, the time is plotted against the speed in *m/s* measured by the GPS, and the instrumented car.

B. Accelerometer

In Figure 14 the measured accelerometer data is compared to the data of the vehicle. They are represented in a similar way as the GPS plots. The red line stand for the car data and the blue line shows the data of the accelerometer in our device. Before the data was plotted the raw data of the accelerometer was filtered with the moving average exponential filter from subsubsection 4.1.3 before the comparison with the car data was made.

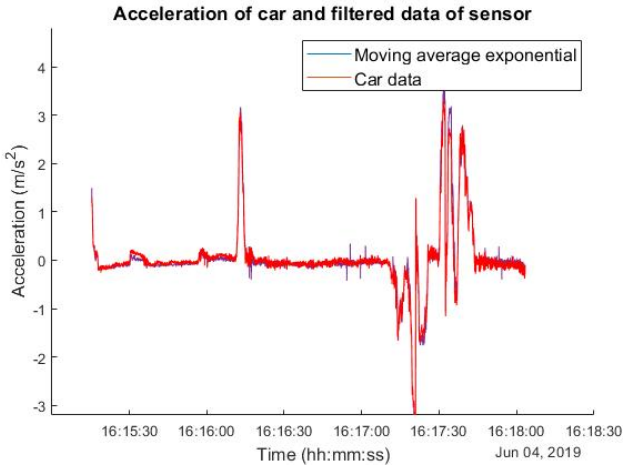

Figure 14: Acceleration from car and accelerometer

4.2.2 Data analysis of the car states

In contrast to the previous results, the data in this section will be analysed according to the state of the car: driving at constant speed or accelerating. To determine the assessment criteria for driving at a constant speed, only specific time frames of the data are analysed.

A. Constant speed

Figure 15 , shown below, is a table presenting the assessment criteria from the two different sensors. In this table only the data when driving at a constant speed has been analysed to see how accurate and reliable the sensory perform in this state

		GPS $[\frac{m}{s}]$	Accelerometer $[\frac{m}{s^2}]$
Constant speed data	Mean error	0.04	-0.02
	Standard deviation	0.24	0.07
	Root Mean Squared error	0.24	0.07
	Mean squared error	0.06	0.005
	MAPE [%]	3.38	443.47

Figure 15: Constant speed data analysis

When comparing the MAPE's of the GPS and the accelerometer,the GPS module comes out on top. Therefore the constant speed can be measured most accurately by a GPS module.

B. Acceleration and deceleration

In Figure 16 , the same errors are calculated in accelerating an decelerating states. In addition the delay has been assessed with this data.

			GPS $[\frac{m}{s}]$	Accelerometer $[\frac{m}{s^2}]$
Accelerating/decellerating data	Mean error		-0.004	0.03
	Standard deviation		2.21	0.39
	Root Mean Squared error		2.21	0.39
	Mean squared error		4.89	0.15
	MAPE [%]		459.49	107.45
	Delay	Mean delay [s]	-1.62	*
		Standard deviation [s]	0.42	*

Figure 16: Acceleration/deceleration data analysis

Because the GPS-module and the accelerometer measure in a different unit, the MAPE is used to compare the sensors. Looking only at these relative values, the accelerometer is more accurate in measuring the movements of the car than the GPS-module. Also, the delay for the GPS-module is 1.62 seconds, which is a lot larger than our performance requirement. Combining these two results, the accelerometer is faster and more accurate in measuring acceleration and deceleration than the GPS-module.

C. Standing still

When the GPS sensor data is looked at separately, it has a large error with detecting that the car is standing still. In Figure 17 the "stopping" output is presented at a value of -3.5. When zooming in on the intervals where a stopping output is generated, one can observe an average delay of 0.688 ± 0.282 seconds, when using both sensors.

In the code that produces this step to -3.5, the stopping output is generated if the GPS speed is below 2 m/s and if the accelerometer gives an acceleration below 0.1 m/s^2. This 2 m/s is of course not a reliable indicator that the car is (almost) standing still, but the delay of the GPS sensor was too large to use a value much closer to zero. Because the stopping moments in this test are short, the GPS would finally detect a value close to zero m/s when the car would have actually started to accelerate again.

4.2.3 Data analysis of the output states

The GPS sensor has the smallest mean error when driving at a constant speed (-0.004± 2.210 m/s) but it does have an average delay of 1.62 seconds (Figure 16). When calculating deceleration out of the speed measured with a GPS-module, two moments in time after each other are required. If $v_2 < v_1$ there is a deceleration. This extra measurement takes 1 second, because the update rate is 1 Hz.

This total delay in displaying the output, generated with the GPS-module, is then equal to the average delay of the GPS-module plus the extra second. This adds up to a total of 2.62 seconds delay before an output is produced by the device. This result can be seen in Figure 17 at the bottom of the graph.

In the figure two lines are plotted: the output based only on the GPS data (in thick red) and the output based on a combination of the GPS and accelerometer (in blue). If these lines are high, i.e. -3 on the right y-axis, the system generates the decelerating output. If the lines are low, i.e. -4 on the right y-axis, the system does not produce any output. If these line are at -3.5, it means the system detects that the car has stopped and the system generates the stopping output. The delay mentioned above can be seen in these lines, when the blue line has gone up to indicate deceleration, the thick red line goes up, with an average of 2.87 seconds later every time. The difference in this delay results from the use of a counter, which will be explained in the next paragraph and in algorithm 3.

The delay in the combination of the GPS and the accelerometer exists of two parts. First, the accelerometer has an update frequency of 104 Hz, meaning there is a delay of 0.0096 seconds. The second part is a result of the decision model. In this model a counter is implemented to get rid of outliers in the raw data. Meaning that a significant number of similar data points must occur after each other in order to change output. This prevents the output to change with every outlier in the data and an example of this can be found in algorithm 3.

```
for i = 1 : length(speedVector) do
    if data is higher than borderValue then
        counter = counter + 1;
        if counter > borderCounter then
            Change state
        end
    end
end
```

Algorithm 3: Example of counter mechanism

In this example the counter principle is displayed. As one can see, the counter value must be higher than a prespecified borderCounter, meaning that the for-loop must have run at least that number of times with values higher than the borderValue. Taking into account that the loop runs every 1/104 seconds (update rate accelerometer), it takes borderValue times 0.0096 seconds for a change of the state to occur.

4.2.4 Summary

In conclusion, to measure the speed and acceleration of the car it is best to use a combination of a GPS module and a accelerometer. The GPS sensor has the smallest mean error when driving at a constant speed (-0.004± 2.210 m/s) but it does have an average delay of 1.62 seconds Figure 16. The accelerometer has the smaller error during acceleration/deceleration (0.03± 0.39 m/s^2) and with a delay of only 0.009 seconds it can compensate for the delay of the GPS-module.

The delay of the GPS-module causes problems in detecting when the car is standing still. When the upper boundary to produce an output for standing still is set at a value close to zero m/s, this GPS-module is unable to detect that the car is standing still before the car starts to accelerate again.

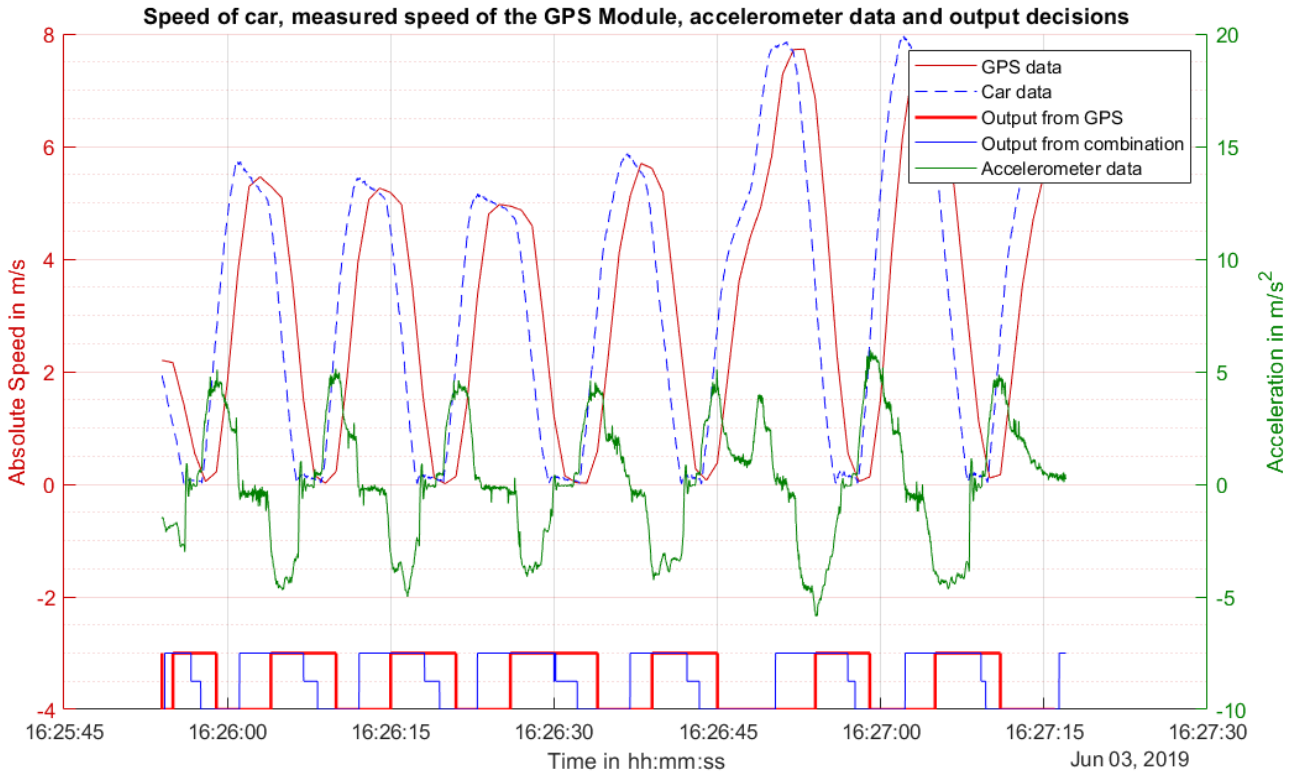

Figure 17: Decision graph for output

5 Discussion & Evaluation

This work aims to set the first steps in making a standalone eHMI for communication between autonomous vehicles and pedestrians. It focuses on examining whether or not it is possible to accurately and reliably measure the speed and acceleration of the vehicle without using the ROS of the car. It is important to gain insight in this subject to facilitate human factor studies with respect to AVs.

In this study, we tested a GPS module and an accelerometer. The obtained data was compared to highly accurate and precise data of an instrumented car. By comparing this data the accuracy and delay of the sensors have been calculated. By analysing these results it is determined how the sensors should be implemented in the eHMI to simulate the movements of the vehicle in the most accurate and reliable way.

The most important result of our study is the founding and validation that a combination of a GPS module and an accelerometer in one standalone device is an accurate and reliable way of measuring the speed and acceleration of a vehicle, looking at the performance requirements. However, the delay of our device when measuring that the car has stopped is too large to be implementable. With the combination of the two sensors, a response time less than 0.1 seconds after accelerating and less than 0.15 seconds after decelerating, as stated in the performance requirements, is acquired. These requirements could not be met using only one of both sensors due to the delay in the GPS system and the noise of the accelerometer output.

Due to limited availability of the instrumented car there has only been one test moment to obtain data. In the first part of the test the speed data from the GPS of the car had not been saved due to a technical error. A shorter test run was performed directly afterwards to still have data, but this did result in a smaller amount of data to analyse.

Secondly, the device did not meet the requirement to give an output within 0.5 seconds after standing still. This might be solved by using a GPS with a higher sample frequency, so this sensor will be able to measure sooner that the car is standing still.

Furthermore, the second design requirement to make the device applicable for every car has not been met completely. This is because the focus has been on the input (which is applicable to every car), instead of also adjusting the LED strip to make a universal mounting system. Moreover, the LED strip that is currently used, has a plug that needs 230 V, which is not easily accessible in every car. To make the device more universally applicable a new LED strip could be designed, which has an easier mounting system and does not require an electricity grid connection, for example by using a rechargeable battery.

In our study, not only the implementation of a GPS module and an accelerometer are examined, several extra features of a standalone device are looked at as well, i.e. ultrasonic, light and water sensors. However, no data has been collected to verify the functioning of these sensors, because we focused our study on the movement of the car. Future studies should focus on testing and analysing these other sensors needed to design a standalone device that functions similar to an integrated eHMI.

Our contribution to the field of human factor research with respect to AVs are the results of the combination of an accelerometer and a GPS-system. The overall conclusion of our work is that the accuracy and reliability of the total device meet our requirements when driving at a constant speed or is accelerating. However, when measuring that the car is standing still, the delay created by our GPS is too large to satisfy that requirement.

References

[1] Claudia Ackermann, Matthias Beggiato, Luka-Franziska Bluhm, and Josef Krems. Vehicle movement and its potential as implicit communication signal for pedestrians and automated vehicles. June 2018. Accessed:14 April 2019.

[2] Michael Aeberhard, Thomas Kühbeck, Bernhard Seidl, Martin Friedl, Julian Thomas, and Oliver Scheickl. Automated driving with ros at bmw. 2015. Accessed:14 April 2019.

[3] Pavlo Bazilinskyy. Assessing the clarity of ehmi concepts via crowdsourcing. unpublished, 2019.

[4] Reanne Boersma, B Arem, and Frank Rieck. Casestudy wepod: een onderzoek naar de inzet van automatisch vervoer in ede/wageningen, March 2018.

[5] Melissa Cefkin. Towards socially acceptable autonomous driving. 2018.

[6] Chia-Ming Chang, Koki Toda, Daisuke Sakamoto, and Takeo Igarashi. Eyes on a car: an interface design for communication between an autonomous car and a pedestrian. pages 65–73, 09 2017.

[7] Michael Clamann, Miles Aubert, and Mary Cummings. Evaluation of vehicle-to-pedestrian communication displays for autonomous vehicles. 01 2017.

[8] Dietrich A. Núñez Velasco J. P. de Winter J. Happee R. de Clercq, K. External human machine interfaces on automated vehicles: Effects on pedestrian crossing decisions. In *Human Factors*, 2019.

[9] Ayn de Jesus. Machine vision for self-driving cars – current applications. https://emerj.com/ai-sector-overviews/machine-vision-for-self-driving-cars-current-applic Accessed:16 May 2019.

[10] Grace Eden. Transforming cars into computers: Interdisciplinary opportunities for hci. 07 2018.

[11] David Galland. 10 million self-driving cars will hit the road by 2020 - here's how to profit. https://www.forbes.com/sites/oliviergarret/2017/03/03/10-million-self-driving-cars-will-hit-the-road-by-2 #7a45d9ba7e50, 2017. Accessed: 17 May 2019.

[12] Leonardo Graziano. Autonomi, ISIA Roma Design Institute, 03 2014.

[13] Azra Habibovic and Maria Klingegard. Communication between pedestrians and automated vehicles. https://www.viktoria.se/projects/AVIP. Accessed:21 Februari 2019.

[14] Kersten Heineke, Philipp Kampshoff, Armen Mkrtchyan, and Emily Shao. Acceleration of a car. https://hypertextbook.com/facts/2001/MeredithBarricella.shtml. Accessed:25 May 2019.

[15] Sureshkumaar Jayaraman, Chandler Creech, Lionel Robert, Dawn M Tilbury, Xi Jessie Yang, Anuj Pradhan, and Katherine M Tsui. Trust in av: An uncertainty reduction model of av-pedestrian interactions. March 2018.

[16] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, and Kazuya Takeda. An open approach to autonomous vehicles. *IEEE Micro*, vol.48, 11 2015.

[17] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. Joint attention in autonomous driving (jaad). *CoRR*, abs/1609.04741, 2016.

[18] Tobias Lagström and Victor Malmsten Lundgren. Avip - autonomous vehicles' interaction with pedestrians - an investigation of pedestrian-driver communication and development of a vehicle external interface. 2016.

[19] Christian Lehsing, Andrea Kracke, and Klaus Bengler. Urban perception - a cross-correlation approach to quantify the social interaction in a multiple simulator setting. September 2015.

[20] Yeti Li, Murat Dikmen, Thana Hussein, Yahui Wang, and Catherine Burns. To cross or not to cross: Urgency-based external warning displays on autonomous vehicles to improve pedestrian crossing safety. 09 2018.

[21] Karthik Mahadevan, Sowmya Somanath, and Ehud Sharlin. Communicating awareness and intent in autonomous vehicle-pedestrian interaction. In *CHI*, 2018.

[22] Victor Malmsten Lundgren, Azra Habibovic, Jonas Andersson, Tobias Lagström, Maria Klingegård, Anna Sirkka, Johan Fagerlönn, Rikard Fredriksson, Claes Edgren, Stas Krupenia, and Dennis Saluäär. *Will There Be New Communication Needs When Introducing Automated Vehicles to the Urban Context?*, volume 484, pages 485–497. 01 2017.

[23] Milecia Matthews, Girish Chowdhary, and Emily Kieson. Intent communication between autonomous vehicles and pedestrians. *CoRR*, abs/1708.07123, 2017.

[24] The Mercedes Benz. The Mercedes-Benz F 015 Luxury in Motion. `https://www.mercedes-benz.com/en/mercedes-benz/innovation/research-vehicle-f-015-luxury-in-motion/`. Accessed:8 May 2019.

[25] Danielle Muoio. A start-up born out of stanford just entered the driverless car race with a radical approach. `http://www.businessinsider.com/driveai-using-deep-learning-for-its-autonomous-cars-2016-8`. Accessed:21 Februari 2019.

[26] Donald Norman. *Turn Signals are the Facial Expressions of Automobiles*, pages 117–134. January 1992.

[27] Nicholas D Pennycooke. Designing biomimetic vehicle-to-pedestrian communication protocols for autonomously operating parking on-road electric vehicles. *Massachusetts Institute of Technology*, 2012.

[28] Laura Sandt and Justin M Owens. *Discussion Guide for Automated and Connected Vehicles, Pedestrians and Bicyclists*, August 2017.

[29] John Shutko. How self-driving cars could communicate with you in the future. `https://medium.com/self-driven/how-self-driving-cars-could-communicate-with-you-in-the-future-e814d276937f`. Accessed:14 April 2019.

[30] Salah Sukkarieh, Eduardo Mario Nebot, and Hugh F. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Trans. Robotics and Automation*, 15:572–578, 1999.

[31] Jon Walker. The self-driving car timeline – predictions from the top 11 global automakers. `https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/`, 2019. Accessed:16 May 2019.

[32] LIMITED WORLDSEMI CO. Ws2812 intelligent control led integrated light source. `https://html.alldatasheet.com/html-pdf/553088/ETC2/WS2812/95/1/WS2812.html`. Accessed:4 April 2019.