UNIVERSITY OF GOTHENBURG

Comparison and visualization of real-time recordings against simulated environment in development of self-driving miniature vehicles

Bachelor of Science Thesis Software Engineering and Management

ALIAKSANDR KARASIOU
LEILA KEZA
TOMASZ RAKALSKI

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2014

Comparison and visualization of real-time recordings against simulated environment in development of self-driving miniature vehicles

Aliaksandr Karasiou
Leila Keza
Tomasz Rakalski

Examiner: Michel Chaudron

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2014

Comparison and visualization of real-time recordings against simulated environment in development of self-driving miniature vehicles

Aliaksandr Karasiou
Dept. of Software Engineering
Göteborgs Universitet
Göteborg, Sweden
aliaksandr12@yahoo.com

Leila Keza
Dept. of Software Engineering
Göteborgs Universitet
Göteborg, Sweden
guskezle@student.gu.se

Tomasz Rakalski
Dept. of Software Engineering
Göteborgs Universitet
Göteborg, Sweden
tomasz.rakalski@hotmail.com

Abstract—**Testing of solutions for embedded systems has repeatedly proven to be an issue. What is a perfectly working algorithm on a testing machine may not necessarily be an optimal fix for a miniature board. When working with miniature vehicle project at the University of Gothenburg, we have experienced these issues first hand. The goal of this work is to evaluate whether we can improve implementing of embedded solutions by enhancing the testing process. The enhancement would be created from enabling to test an embedded solution directly on a real world recording, without engaging the actual hardware. In order to test our hypothesis we conducted an experiment where we devised a number of scenarios, representing different types of lane following, and performed a recording of lane following with the help of a miniature vehicle. These recordings were done within both a simulated environment and a real world physical track as well as compared. We were able to determine the differences between the recordings and whether it is negligible enough, so that the simulated recording can be deemed sufficient for testing. We have concluded that our solution cannot enhance the testing unless improved further.**

Keywords—embedded systems, testing, miniature vehicles, Carolo Cup, simulation, OpenDaVinci

I. Introduction

The teams developing cars for Carolo Cup 2014 [CC2014], at the University of Gothenburg, have been working on creating the solution for miniature autonomous vehicles, which resulted in extensive testing in the field of algorithm implementation. This idea has received partial attention when one of the team members involved in the Carolo Cup, recorded the track and continued to work with image processing based on the data received.

When developing embedded systems, one frequently experiences problems when testing software solutions directly on the hardware [ElS98]. Having to conduct test cases directly on hardware increases the difficulty of testing and requires a lot of time. Compilation and running of code on hardware for testing purposes can be reworked into a three-step approach, where solutions are first applied to a simulated data and then verified whether they can be tested on a software level, with hardware input transferred to more powerful testing machines.

Currently software solutions for embedded systems are being tested in a two-step process. This process involves creation of a software solution, testing in a simulated environment and implementation directly on the hardware [BCHLS2013]. This process requires a certain amount of guesswork and assumption [ElS98] which could be eliminated.

There is a simulation environment, which is used for modeling the track and its surroundings for testing. It is called Hesperia and was used for testing purposes both during Carolo Cup competition and the university course. Hesperia has all the components of the real environment, such as lane markings, intersections, side objects etc. In addition it generates the data of sensors and camera image. However, all the data is very accurate and does not consider the existence of hardware factors and real environment noise. Modeled environment is represented with solid colors and perfect-shaped lines and objects. In addition, sensor data generated on a computer is much more frequent, than on a board, due to the difference in processing power.

We have experienced a testing gap while implementing solutions from the simulated environment to the hardware. There is a basic concept of the solution we want to research, the needed data and resources, which will change and expand during the research process.

The ability to test lane following algorithms effectively without having to engage the hardware is a serious challenge. Currently, we are unaware of an efficient solution to this issue. A common problem is that embedded software testing requires either prototyping or guesswork [ElS98]. We are going to create a test that will compare the data from the Hesperia simulator with the data gathered by recording of the track in a physical environment. Should the results display that the difference between the simulated environment and the physical one is minimal, it will be possible to use the simulated data in order to test possible algorithms, for the autonomous movement of the miniature vehicle.

This study will help us determine whether it is possible to shorten the testing in the development process of a miniature vehicle, by diminishing the gap between the simulated environment and real world data. This study is relevant because it might shorten the amount of time required for validation of a software solution [BCHLS2013].

II. Related work

Our investigation aims to compare the simulation and hardware data for an autonomous, self-driving vehicle. Any research related to hardware-in-the-loop (HIL) simulation testing, where the camera and sensors are considered, is related to our research.

Comparing virtual and real camera images has been explored by Rander, Narayanan, et al. [RNK97]. They found out three main factors that make the virtual and real camera images differ and the impact of those aspects in the image processing research [ACIP2010]. The virtual environment is built by using simulated real life objects and the real environment is uncontrollable. The real environment can be influenced by external objects which were not simulated during the virtualization [BC95]. The simulated environment is a normal computer interface and the virtual camera motion is not affected by the environment since the two are different components and do not interact with each other [BBSP2002]. The camera images taken in the real-world environment can be affected by different unpredictable factors when the virtual camera images do not show any interaction with the scene, with the exception of out of bounds rendering.

Furthermore, they brought out yet another aspect that makes the images from the two cameras differ when only comparing pixels [ACIP2010]. The intensity of the surface of the real camera images varies according to the viewing angle, which is not the case for the virtual camera image. Virtual images can also be inconsistent compared to the real camera images due to movement. The virtual camera operates discontinuous movements, which can make the virtual camera images inconsistent [GMS2009].

Various papers [PvGVVCTK2004] [RNK97] [GMS2009] do not report any significant reasons to stop using simulation for image processing, development and testing. Instead, their models contribute to an improvement of the way the objects are managed in the virtual scenes to optimize a resemblance between the two environments. Rander, Narayanan, et al. [RNK97] combined the virtual and real environment by modeling the scenarios that have impact on the images in the two environments such as shadows, viewpoints, etc. Gilad et al. [GMS2009] proposed solution was about extracting feature points and neighboring correspondence, in conjunction with some algorithms proposed by Pollefeys et al. [PvGVVCTK2004] such as image subtraction pixel-by-pixel, geometric model and view-dependent geometry.

Gietelink et al. [GPdSV2006] conducted an experiment on Vehicle-in-loop (VEHIL), a multi-agent simulator method for design and validation of Advance Driver Assistant Systems (ADAS) envelopment. The ADAS are technologies such as sensors, camera GPS, radar, vehicle mounted laser, and they provide security related information to the driver while the car is driving. The ADAS has shown an important performance in the road security. Surveys have been conducted with ADAS performance in mind, and they have shown impressive results regarding contribution to the reduction of the numbers of accidents. VEHIL simulation environment uses a combination of real and simulated cars for testing of ADAS' performance. The VEHIL principle is the same as a normal HIL sensor based testing for the detection of other objects on the road. The difference is only that VEHIL combines a real car and moving robot instead of a single car on a simulated track. However, the VEHIL testing has been limited on the sensors and reserved the driver testing for the next iteration to be able to validate the results.

Arrichiello et al. [ACIP2010] investigated the Null-Space-based-Behavioral, which consists of controlling a robot by prioritizing the task in such a way, that they are executed hierarchically. They found out that the projection technique used in NSB is not enough to assume that the tasks hierarchy is respected. Further, they proposed a conjunction of the NSB and velocity saturation management. For the results validation, they conducted a number of "numerical simulations" and hardware testing. It was concluded that their solution worked perfectly on the simulation and hardware, while observing a performance increase in obstacle avoidance. Umeda et al. [UOK96] tested the use of multiple ultrasonic sensors' wide-angles for distance measurement to a moving obstacle. They used Kalman filter algorithm to estimate the movement of obstacles, from range of a single ultrasonic sensor, in relation to others installed on the same vehicle. Arrichello et al. [ACIP2010] proposed fusion of multiple ultrasonic sensors for the moving obstacle distance measurements. Their solution is a compliment to the optimization of the NSB behavior as discussed in their study.

Umeda et al. [UOK96] tested their solution on both hardware and simulation and observed a more precise and rapid distance measurement to the mobile obstacles. Moreover it also tested positively while handling changes in movement direction.

Benet et al. [BBSP2002] focused on using infrared sensors for distance measurement in an autonomous system. They studied the case of Yet Another Intelligent Robot's sensor-based distance measurement and focused on the infrared sensors. YAIR robot uses two ultrasonic and sixteen infrared sensors. The infrared sensors can measure the distance between zero and one meters, which are more precise compared to other infrared sensors, normally used for obstacle detection in moving vehicles. In their findings, Benet et al. [BBSP2002] pointed out an error in the distance measurement due to the noise and errors in the distance estimation of YAIR's infrared sensors. To address those inconsistencies, they developed a model based on the infrared reflectivity coefficient of the surface using the ultrasonic sensors. They tested their model on hardware and simulation environments and the results showed a similarity between the simulation and hardware test results.

Gat [G92] developed and conducted an experiment on A Three-Layer Architecture for Navigating through Intricate Situations (ATLANTIS). Gat's architecture is an action-model based architecture developed to solve hardware-software integration problems when a simulation has been used during the development. ATLANTIS is a simulated architecture that models the unpredictable factors observed in the real world, in order to facilitate code integration directly from simulation to the hardware. The results show that ATLANTIS presents a high performance in controlling the robot, while taking into consideration the noisy and unpredictable factors, while running in simulation. The same software was integrated in the hardware and a similarity has been observed.

III. Purpose of the Study

The aim is to determine the difference between the simulated data provided by the Hesperia simulator and real life environment and to implement a testing environment in the development process of miniature smart vehicles. The testing environment is represented by a simulator, which compares the data, gathered both from Hesperia simulation environment and recorded data from the track.

We want to investigate, whether it is possible to shorten and improve the development process, by introducing an approach that would allow developers to directly measure the difference between the data in the simulated environment and the actual physical environment. It is possible that the implementation would shorten the amount of time it takes to test the software, since it eliminates the requirement for the proposed algorithms to run on the hardware. It could enable usage of data captured from the real life recording and remove the delay caused by initial installation on a miniature board. With this solution the lane following implementation can be sent to the board when it is certain that it can handle real life input.

The objectives are:

- To determine whether the difference is small enough to allow usage of the physical data in algorithm testing on the board.
- To examine the data from the camera image generator (Hesperia simulator) and actual video data.
- To examine the data from the simulated sensors (ultrasonic, infrared) and the actual data from the track

A. *Research questions*

- What is the difference in accuracy between the data provided by the simulated environment and the real world data?
- If insignificant, how can the data be utilized with the aim of guiding the future development with the help of an adequate visualization?

IV. Methodology

This is a quantitative study, where an experiment [ZW97] will be carried out, because our intention is to study the impact of the difference between simulation and reality to the autonomous car software and hardware integration. The data will be collected with the help of recordings from multiple sources on both the simulated environment and the physical track. The difference between the gathered data will be determined with the help of statistical standard score [B2010]. Both types of recordings will be done with the help of a miniature vehicle, either a physical construction or a digital creation. Both the physical and digital vehicle will have the same physical parameters, such as size, camera position and sensor layout. To determine the exact car movement patterns, a set of scenarios is used to figure out how to translate the path following into actual data. These scenarios will enable us to create a method to validate the comparison. If, for example, the car is following a straight path, we can translate what it sees on the physical track and redraw that in the simulated environment with the help of its editing tools. Once that is completed a run of the simulated track combined with gathering of numerical and image data will allow us to make the comparison with the help of statistical math.

A. *Track*

The physical track used for recording allows for direct measurement of total size, road width and length as well as possible objects and obstacles. We are going to perform a physical measurement of the track, which we will then use to recreate it in the virtual environment. This will eliminate the problem of having to execute scenarios on different types of track, rendering the comparison pointless.

B. *Recording parameters*

We have determined the types of relevant data required for performing of this study and decided that we are using five main attributes in the comparison. A recording is set up in two ways depending on the source (simulation or physical track).

1) Simulation

In the simulated environment we are gathering data with the help of inbuilt components, which simulate the outcome of the actual devices used on a physical car. These components have been programmed to simulate output based on a feed given from track data created in the OpenDaVinci editing tool, ScUI, and are set to operate within a factor of 10 of the actual values [BCHLS2010]. The video data is provided by the camera image generator, which provides frames based on the virtually generated track and how the simulator perceives it from the point of the car. The sensor values are given from the component IRUS, which generates sensor data based on the simulated distance between the focal point (the simulated miniature vehicle) and the virtual objects on the track. The virtual miniature vehicle contains a method of measuring its travelled path, which determines whether it is following the path it is supposed to. Moreover the virtual car is also receiving data from a basic driver component, which provides information about current heading and speed of the vehicle. All these values are gathered based on the frequency of the feed, normally about ten times per second and can be identified for comparison with the help of a timestamp, which provides a way to index the data based on what time it was recorded and compare it to the same position in the physical recordings.

2) Physical track

The data gathered on the physical track is of the same type as that gathered from the simulation, however the sources vary. In the simulator we can gather camera images from an automatically generated feed, however in the physical car, the camera operates at a set amount of frames per second, which require adjustment to match the amount of data we gather per second on the simulator. Moreover, we also need to perform recording of individual components with multiple devices, instead of using components of the OpenDaVinci framework, like we do in the simulator. The ultrasonic and infrared sensors are operated by their own component which provides numerical data, which translates into positions of objects around the miniature car. The car is also equipped with a device, which generates the travelled path numerical data based on the cars speed and heading. Speed is determined by how many times the wheels have turned in the last second; the heading is determined by an inbuilt gyro. Again the timestamps based on scenario progression are used to determine the location of the data and its comparison place.

3) The values

The values we are gathering from both environments are as follows:

- Camera images in raw format, gathered with a frequency of 10 per second
- Infrared numerical output, providing distance between the car and objects
- Ultrasonic numerical output, providing distance between the car and objects
- Timestamps, gathered to identify the sets of abovementioned data types in milliseconds

Apart from these types of data, we also expect to gather additional information, which might affect the results of this study.

C. Independent variables

The variables we have control over in this study are:

- The tracks, both physical and virtual; considering the virtual track can be rebuilt after the physical one
- The cars, mainly considered recording devices for this study
- The environmental parameters for the virtual testing environment
- The amount of recorded data used for comparison

D. Dependent variables

The variables we do not have direct control over are the values we will receive upon application of statistical algorithms on the recorded data. This data is presented in section B3 and represents all the data the cars are capable of gathering, which is also considered relevant.

We also cannot anticipate the levels of noise, which might be introduced while performing recordings in the physical environment. Recording in a closed location introduces issues with light reflections, possible imperfections on shapes detected by sensors, accuracy problems when determining distance between multiple objects and also power levels of batteries, which can alter the speed of the cars.

E. Scenarios

The scenarios will represent the behavior of the miniature vehicles on both the virtual and physical tracks. In order to ascertain the followed path to be exactly the same in both environments, we are going to use the measurements of followed path combined with a predetermined case for the test itself. For the standard lane following, we are using a set of scenarios that determine straight path line following, left hand turn, right hand turn and intersection handling. As for sensor outputs, we have integrated those with the image comparison and measure them with the help of objects placed around the track during lane following scenarios.

F. Scenario example

Straight line following – the car follows a path set from the point of origin to a hundred units directly forward. The car follows the predetermined path and makes recordings at the set frequency. For every recording there is timestamp which provides means of comparison. The recordings keep being taken until the car reaches end of the defined track. No path following algorithms are required to follow the track, the car is being driven manually.

G. Visualization

Once the data is gathered, it is possible to create a plug-in for the OpenDaVinci library, which will allow for visualization of the data. This visualization will help future users to directly determine any differences and help with testing. The visualization will be a component running the tracks in parallel and it will allow for a clearer representation of tested lane following algorithms.

H. Statistical testing

The statistical test will be performed with the help of a t-test. We have found the t-test to be the best tool when dealing with large samples of paired data such as in this study [W2012]. The data gathered will be represented by a large amount of sets of values. The amount of sets will be the amount of seconds in a scenario multiplied by the number of recordings per second. Since the scenarios will usually run for longer than five seconds, we can determine that the sample size can be considered large (>50). Since we are working with a normal population, the sample is representing the population accurately. The normality of our population can be assumed due to the fact, that any additions to the current stock are controlled and analysis has been performed on the entire available recorded material. Since we will receive data from two main sources, the sets of data will be directly linked to each other with the help of timestamps. These linked pairs will have a statistical score and based on accuracy of 0.05, we can determine whether they are close enough to each other. The 0.05 value is the standard statistical alpha value used for determination of sample size versus its accuracy. Moreover, with the value of 0.05 we only have a 5% chance to perform a Type 1 error. With the statistical testing, the research question one can be rephrased into a hypothesis:

- H_0 There is no significant difference between the data sets from the physical and the simulated track

- H_1 There is a significant difference between the data sets from the physical and the simulated track

With the statistical score we will be able to determine whether we can reject H_0, and thus determine whether the simulator can be more widely used in testing of embedded solutions.

V. Results

This sections describes what we have unearthed once we applied our methodology to the variables presented earlier and preformed the study. In this section we are also presenting issues that have slowed us down and made us pivot certain aspects of the study. For our experiment we focused on straight-line following.

A. Data gathering (simulator)

In order to gather the data, we had to alter the Hesperia framework, so that it not only handles the miniature vehicle, but also records its surroundings provided by the camera and sensors. We have made alterations to the lane detecting portion of the framework, which allows us to capture ten images per second from the camera feed and save them. For storage purposes we have also developed a way to store those images in a binary format, combined with a timestamp gathered in the recording. Recording and storage had to be performed separately due to the sheer amount of numbers the image recording requires. Each image contains 640x480x3 numbers representing the color values for each pixel in a given image. Since the recording captures ten images per second that number is multiplied adequately. This results in a massive amount of numbers that have to be processed and stored in a binary format, resulting in a major slowdown. The timestamp mentioned earlier, provides us with an ability to synchronize the recording of images and that of the sensors.

In order to gather the sensor data we have altered the driving section of the framework, where we capture data from the sensor handling and implement the storage before executing driving commands. The saving is done directly into binary format, since sensors require significantly less memory power. Unlike images the sensors only require 6+1 numbers per iteration. The six represents the amount of sensors on a vehicle and one corresponds to the timestamp displaying when the data was actually gathered.

The gathering process differed a lot in the simulated environment and the real car. In the simulated environment we had to recreate the track using the ScUI editor, an example of which can be seen in Appendix 1. In order to recreate the track, we have measured the start, end and center points of central lines on the physical track available at the university. Based on these central lines we have developed a calculator that provided us with {X,Y} coordinates of left and right lines, which represented borders of the lanes on the track. Due to the limitations of the editor we were unable to exactly recreate the track and had to deal with rendering problems. These limitations were based on the fact that the editor cannot simply draw straight lines, but instead have to work with rectangular elements called point modifiers. These elements take between two and three coordinates, as well as a value representing width of the element and recreate a rectangular object representing a lane segment. This lane segment can have its edges painted to represent either a straight line, which translates to the border of a road, or a segmented line, which translates to the central lines in the middle of a road. The editor does not allow for alterations in the distance between segmented lines, which resulted in us having to draw the track with a point modifier for each segmented line and the distance between them. In the end the track consists of 267 objects. Handling this many objects and the relations between them has proven to be very difficult and has resulted in some issues with rendering the track in the simulator. This issue was partially solved after re-addressing the way with which the lanes were drawn, however upon implementation of static curves, the renderer tended to place coordinates in wrong locations.

Apart from recreating the track, we also had to take into account the way image is displayed in the simulated environment. The camera angling of the virtual car differs quite a lot from the real car and we had to take that into account. The difference between the camera images can be seen on Figures 1 and 2. We measured the angle of the camera installed on the real car and translated it into the simulation with the result looking like Figure 1.

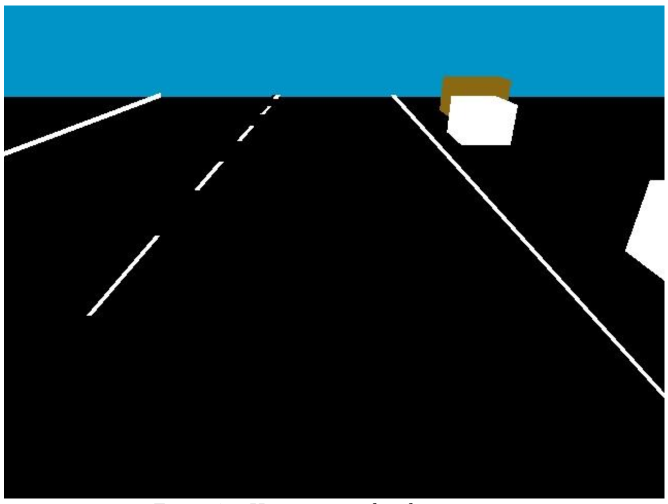


Figure 1. Hesperia simulated car image

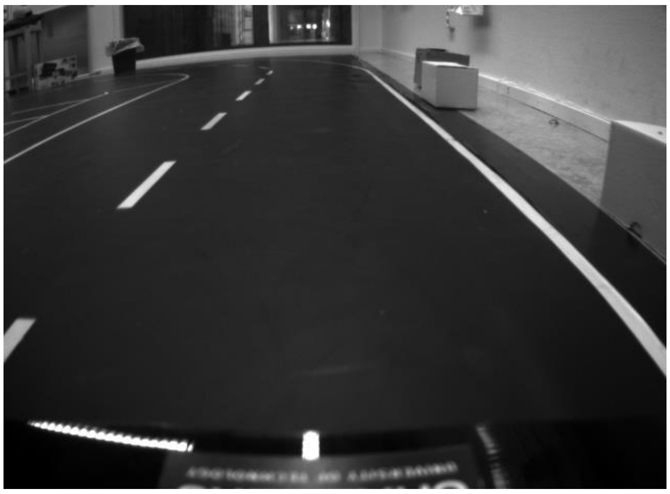


Figure 2. Real car camera image

In order to perform testing on sensors we have placed several rectangular boxes on the physical track and recreated them in the simulator. Sensor positioning in the simulated environment had to be adjusted as well, in order to match the layout of the real car.

Finally, the starting and ending position was determined and translated into the simulator in a similar way with which we drew objects. We placed the real car on the physical track, triangulated its position with the help of the center lines and recreated the starting location within the simulator.

B. Data gathering (physical track)

The way, with which we gathered data, was exactly the same on the physical track as with the simulator. The differences between the two were caused by the different issues the real car has given us. Initially we had to deal with the camera resolution being different, that is 752x480. Because of this difference we had to crop images, excluding regions which we deemed uninteresting for comparison, such as exterior building walls.

Apart from the camera working differently, we have also discovered that the sensor range is different to that of the simulator and had to be adjusted. We wanted to see if the car recognizes objects around it at a similar time and distance, which required us to use the same layout and range in the simulated environment. The only difference in sensor output between simulator and real car was the floating point accuracy, which required an alteration of the recording system.

C. Data analysis (image)

In order to analyze the images we first looked at what sections of each pair were important to us. We discarded sections which were completely empty or showed objects which were irrelevant to the study, such as building walls. What we deemed important on the images were the side and center lines, so we marked them accordingly as regions of interest. Within these regions of interest we have done pixel by pixel color comparison on lines. An example of region of interest division can be seen in Figure 3. This comparison resulted in a percentage value, which shows how accurately the lines are represented. However, the percentage value only shows the difference between the amounts of pixels, not their distribution. We will discuss improvements in the accuracy in the Discussion part.

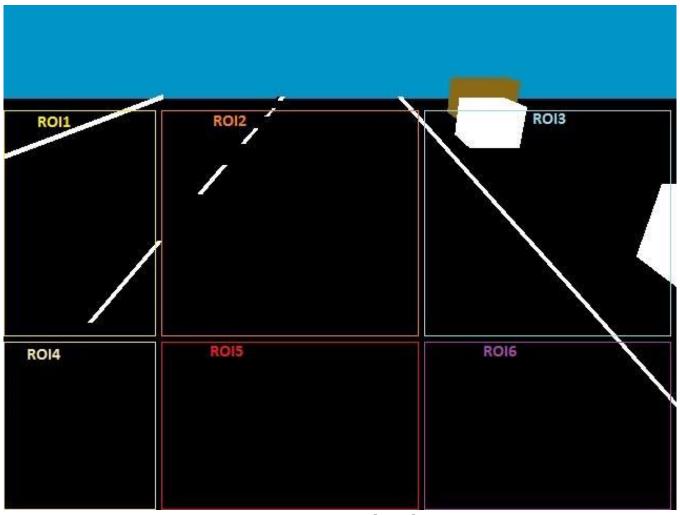
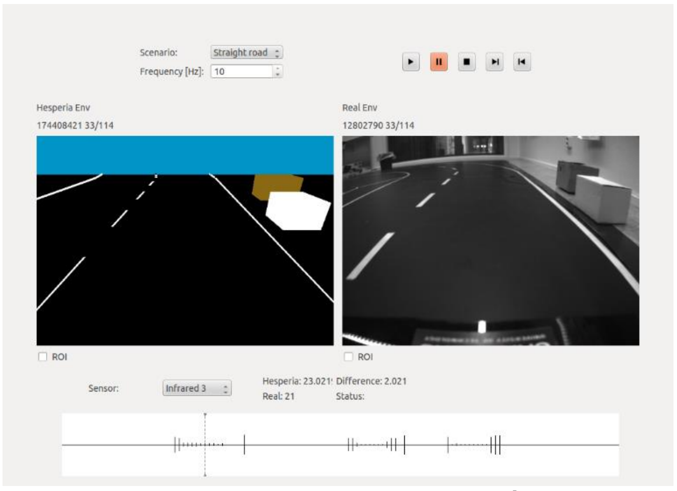


Figure 3. ROI distribution

Once we have established the regions of interest, we represent the pixel values with the help of histograms. Each pair of images, one real and one simulated, has a histogram showing color distribution among pixels, which represents how close they are to each other. The listed results of these comparisons for each recording are shown in Appendix 2.

D. Data analysis (sensor)

In order to compare the sensors, we use the binary files created by the recordings, read them and subtract the values from each other. The differences in the sensor readings can be seen whenever an iteration is not showing zero. However, due to the fact that we are dealing with doubles and integers, there are some floating point values, which require a closer look. Detailed results for sensor recording can be seen in Appendix 3, which show differences between two recordings.


Scenario: Straight road
Frequency [Hz]: 10

Hesperia Env
174408421 33/114

Real Env
12802790 33/114

☐ ROI
☐ ROI

Sensor: Infrared 3
Hesperia: 23.021 Difference: 2.021
Real: 21 Status:

Figure 4. Hesperia Environment Comparator plug-in

E. Hesperia Environment Comparator (HEC)

In order to visualize our solution and actually provide functionality we have created a plug-in for the Hesperia framework. This plug-in allows the user to directly compare recordings from a given scenario and review the differences. Images are displayed side-by-side showing both the simulated recording and physical from the same start and end locations.

The plug-in allows for review of not only images but also sensor data with the help of a graph displaying sensor readings as well as whether they are within the acceptable range difference.

F. *Results vs hypothesis (image)*

With all the results gathered, and the critical value (c) set to 0.05 we finally managed to apply a formula which would allow to check our hypotheses. We followed a standard formula for determining a test statistic with the help of an alpha coefficient, which can be seen in Figure 4.

$$Z = \frac{\overline{X} - \mu_0}{\sigma/\sqrt{n}}$$

Figure 5. Calculation of a test statistic

Since, according to the peak signal-to-noise ratio approach, very similar images have ratings between 30 and 50 [YS2012]. We decided 30 to correspond as the value subtracted from the mean. The standard deviation resulted in 0.87 with the total sample size of 114. Once we replaced the variables with results, we could see our test statistic as shown in Figure 5.

$$-57.81 = \frac{25.29 - 30}{0.87/10.68}$$

Figure 6. Results applied to the equation

A Z score of -57.81 gives a p value < 0.0001.

Using the Structural Similarity (SSIM) algorithm, the values are closer to 100% if the compared images are similar [YS2012]. We decided 99.99 to correspond as the value subtracted from the mean. A z score of -44.85 gives a p value which is less than 0.0001.

$$-44.85 = \frac{93.81 - 99.99}{1.45/10.68}$$

Figure 7. Results applied to calculate p value

G. *Results vs hypothesis (sensor)*

For the comparison of sensor values we are using the same formula as shown in Figure 4 with the same accuracy factor of 0.05. In the case of the sensors, we are working with differences between recordings with the optimal value being 0. Acceptable difference between sensors is up to 5 mm. IR sensors 1 and 2 did not detect any objects during the recordings and have also been ignored. The results for sensor IR3 display a mean of 2.89 and standard deviation of 7.19. If applied to the formula in Figure 4, the test statistic becomes 3.5075. This value give a p value of 0.000452 which is less than 0.05.

IR 4 with the mean of 1.46 and standard deviation of 4.74 produces the test statistic of 2.1818, which gives a p value of 0.029124 which is less than 0.05.

Ultrasonic sensor recording did contain a lot of interference from outside sources, contrary to the simulated sensors which

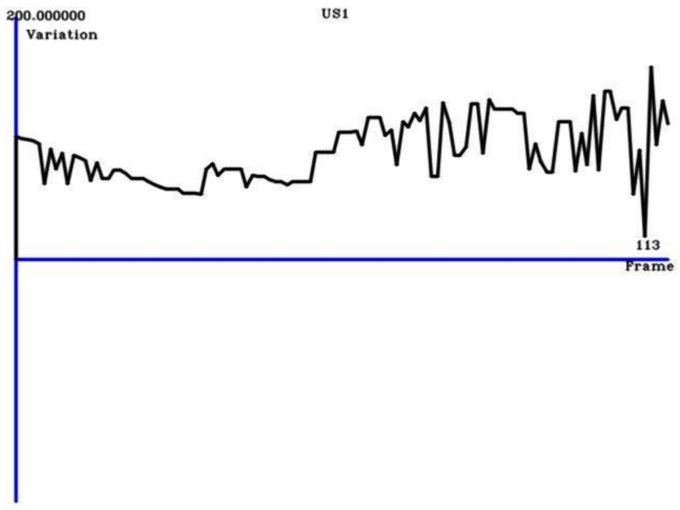


Figure 8. US1 visualization

had little to do, due to lack of any sort of physical boundaries around the track. The recording for US1 with the mean of 89.25 and standard deviation of 25.63 produced a test statistic of 37.7659, which give a p value which is less than 0.00001. Finally US2 with the mean of 87.25 and standard deviation of 19.92 produced a test statistic of 46.5147, which give a p value which is less than 0.00001. Visualizations of ultrasonic sensors can be seen on figures 8 and 9.

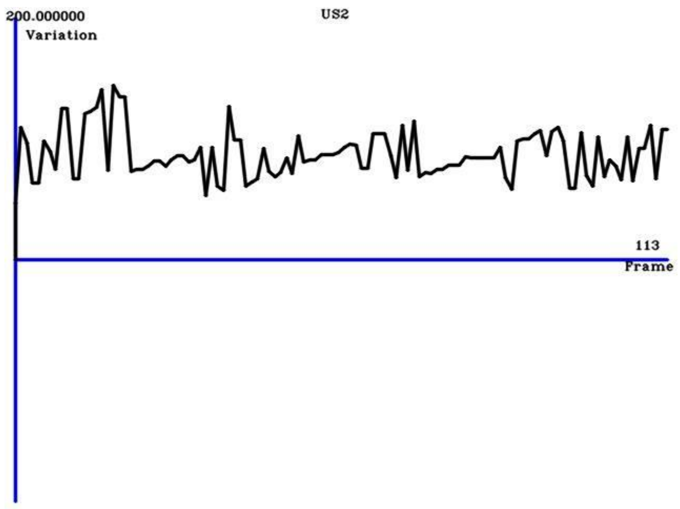


Figure 9. US2 visualization

Visual representations of differences in sensors IR3 and IR4 can be seen in Figures 10 and 11. Visual representation of histogram results in picture comparison can be seen in Figure 11.

VI. DISCUSSION

The data comparison between hardware and Hesperia simulation are discussed. Furthermore, the results will be compared to the other researchers' finding on the hardware and simulation comparison.

A. *Results discussion*

Our image comparison technique has been motivated by the Peak Signal-to-Noise Ratio (PSNR), The Structural Similarity (SSIM), Human Visual System (HVS) and Universal Image Quality Index (UIQI) for the image quality measurement made by Ysura and Soong [YS2012] and also other studies [PvGVVCTK2004] [RNK97] [GMS2009] which consist of feature point extraction and application of different algorithms for optimization of image comparison.

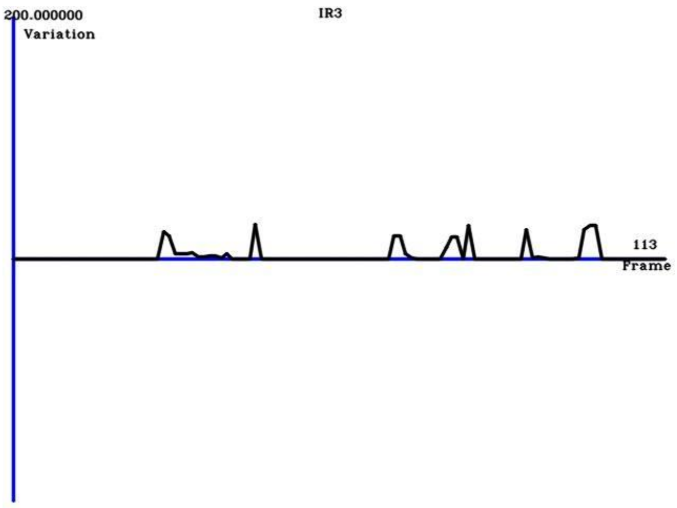


Figure 10. IR3 visualization

We found enough to compare the image comparison data d by using PSNR algorithm's that is more pixel by pixel comparison and add SSIM since it is human based [YS2012]. For our case we will focus on PSNR and SSIM values which are available for review in Appendix 2.

Image	PSNR	SSIM	HVS	UIQI
Hats	18.4	0.78	0.34	0.88
Ship	12.0	0.55	2.68	0.76
Window	13.1	0.57	1.04	0.83
Toys	15.1	0.86	0.35	0.93
Butterfly	20.5	0.9	0.14	0.95
House	13.67	0.59	1.9	0.83

TABLE I.

The PSNR value vary between 30 and 50 if two similar images are compared and the higher is better [YS2012]. For the SSIM algorithm, the values are between 0 and 1 [YS2012] if compared two similar images the values are much closer to 1. In their comparison, the greatest PSNR value is 20.5 that correspond to 0.9 SSIM (Table 1). Our image comparison results show a mean PSNR value of 25.29. If we consider the values in Appendix 2, our PSNR value (25.29) should exceed 0.9 SSIM.

In Appendix 2, 25.29 PSNR corresponds to 93.81 % SSIM. The standard deviation is 87 PSNR and 1.45 SSIM. The median is 25 PSNR and 94 SSIM, which are close to the mean. All those values show that the values are closer to each other.

The identical image subtraction should be 100%, which is not the case for our research. However, [ACIP2010] explained that the intensity of the camera image surface varies according to the viewing angles. This was the case of the car we used. The real camera was mounted on a railing, which made the camera angle downwards at roughly 45 degrees". This caused an irrelevant difference in the lanes. Histograms in figure 11 show the color distribution after the image subtraction. The remaining picture region of interest histograms are computed for pixels values = 0 to 255 where 0 is black and 255 white (figure 11). Those results show a large number of black pixels and a very small number of white pixels. Since the standard deviation is not that much bigger, all histograms look mostly alike and the results should be interpreted in the same way.

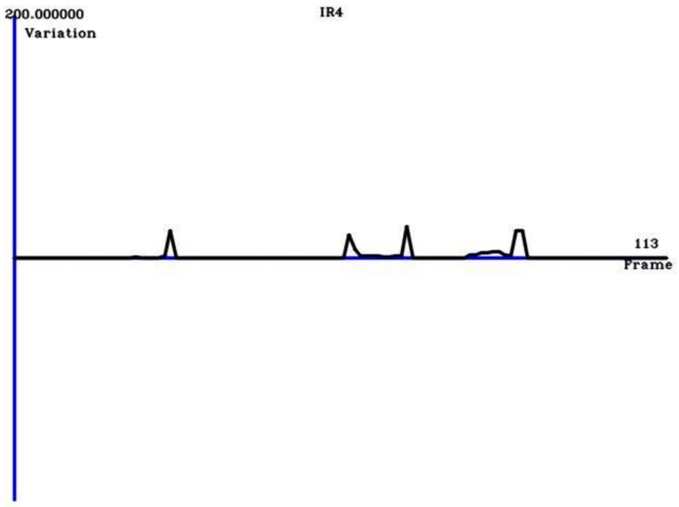


Figure 11. IR4 visualization

The ultrasonic data shows a big difference of 89.2506 and 87.2559 and a standard deviation of 25.6315 and 19.9203. It is easy to consider that as anomalies or error in the recordings or sensors. *Gietelink et al.* [GPdSV2006] conducted experiment on VEHIL simulation combing the real and moving robots and he has got positive results that worked both on the simulation and hardware.

Gat’s [G92] architecture modeled the real-world environments’ “unpredictable” factors in order to facilitate the software-hardware integration directly from the simulation to the hardware-in-it. That architecture worked fine on the simulation and hardware.

Our experiment environment was not appropriate to compare Hesperia sensors data and real car data. The ultrasonic sensors can sense objects from 30 m and the experiment has been operated in a classroom and the wall and other objects placed there influenced the results.

The left infrared sensors did not give any results because of the absence of the objects. For the right sensors, we have a mean of 2.88972 and 1.46064. The standard deviations of 7.19093 and 4.74485. The results are explained by the absence of the object and out of the 114 the median is 0.

B. *Threats to validity*

This subsection reports the four threads of validity described by Runeson and Höst [RH2009]. Those threats are the following: internal, external, construct and conclusion.

1) *Internal validity*

The internal threats of validity are defined as the investigation of the causes and risks that can affect the researched factor [C2009]. An internal threat of validity is identified when the researcher does not have enough information about the factors that can affect the researched factor.

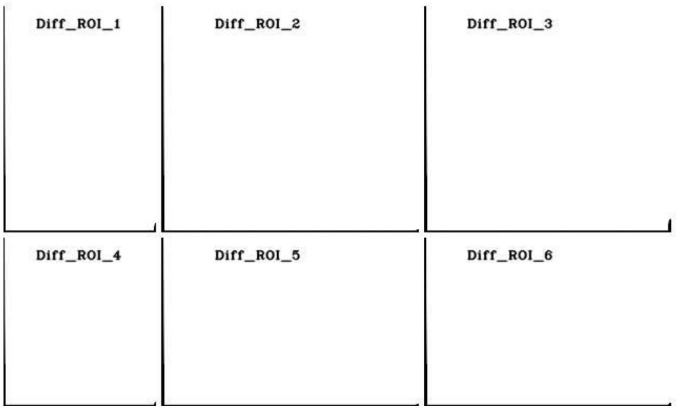


Figure 12. Histogram results for ROIs

In our study, the largest risk comes from instrument change. Different miniature vehicles utilize various types of cameras and post processing, therefore in order to utilize the solutions proposed by this study to their full extent, one would have to adapt to different camera imaging. In this study, however, we have only worked with one source of physical track imaging. Another possible threat is the amount of recordings conducted. The higher the amount, the more likely it becomes that the variation between images will cause inconsistencies. Finally, while recording on the physical track, limitations in battery power may cause the speed of the vehicle to alter in such a way that it is invisible to the human eye, but would cause delays between the recordings.

2) *External validity*

External validity means that the study can be generalized outside the setting, where the study has been conducted [C2009] [RH2009]. Our findings can be extended to other companies working with software development of embedded systems using simulation. In addition to the companies developing software for the Advance Driver Assistant Systems (ADAS) such as camera and sensors for automobiles, our experiment can be also helpful for the Artificial Intelligence (AI) developers.

The only weakness of our findings is that our experiment was based on the simulation and hardware data comparison, based on the miniature autonomous car that has been developed by students for learning purposes. Our findings are prone to be affected by various environmental causes, such as random objects, wall distance and lightning, none of which are present in a simulated environment.

3) *Construct validity*

Construct validity is about knowing whether the study has been done in the way the researcher thought and if the results answer the research questions explicitly [C2009].

For our research, the recording techniques have been developed in such a way that there is next to no delay and data gathering proceeds smoothly in both simulation and real car recording. For image comparison we compare results from two algorithms as well as color distribution histograms. The sensor data analysis has been implemented and analyzed statistically with additional graphical visualization. The plug-in provides space for further enhancements. The only visible threat comes from the simulated track, which has proven problematic and could use further enhancements in order to increase accuracy of the lines.

4) *Conclusion validity*

The conclusion validity is about reliability between the results and the research itself. The conclusion has to be real so that any other researcher can say the same thing if the research should duplicate [C2009]. The threats to conclusion validating arise when the research questions are not clear or if the conclusion is not based on the findings.

In our case, we mitigated the threats to conclusion by designing clear research questions and implemented algorithms for data collection and data analysis. Moreover, we have also saved every possible bit of auxiliary data produced by recordings before applying any sort of post-processing.

For the hardware the recordings were operated in conjunction with the simulation environment. Further research has been done on the related work in order to understand our topic and results.

The weekly reports submission and meeting with the supervisor for review and questions has been important for our results gathering and analysis.

VII. CONCLUSION

In our experiment involving a straight road, our results show that the differences between reality and simulation are rather small (<10%). These differences are considered acceptable due to the fact that the imaging does contain a multitude of artefacts, such as wall sockets, glass walls, noise and the bumper of the real car. Because of all these artefacts, we felt that a difference of <10% is acceptable. We hope that due to this the testing can become more and more straightforward and also improve future solutions for lane following with miniature vehicle project at the University of Gothenburg.

In this experiment we used histograms and color pattern recognition to analyze the differences between images, however it might be more accurate to use pattern recognition instead. This approach would involve scanning for line locations, their start and end points and recreating them in order to match their position, width and color between both recordings. This approach will allow to not only notice patterns with pixel by pixel color distribution but to more thoroughly analyze the layout with which the pixels are aligned.

This experiment leaves behind a way to record and conduct experiments on different types of tracks and lanes, such as curves and intersections, which can be used to further enhance this type of approach and increase the validity of simulated testing.

The plug-in we have provided for the simulated environment will hopefully be an adequate tool for visualization of recordings, which will allow future developers to monitor how their solutions behave in different environments.

It is also a baseline for further enhancements, namely application of algorithmic solutions and visualization of lane following algorithms directly on both types of source material.

ACKNOWLEDGMENT

Aliaksandr Karasiou – Abstract, Introduction, Purpose

Tomasz Rakalski – Methodology, Results, Conclusion

Leila Keza – Related work, Discussion

We would like to thank our supervisor Christian Berger, for not only providing us with access, information and support regarding the Hesperia framework, but also for the feedback we have received throughout the duration of the thesis course.

We would also like to thank Viktor Botev and James Omoya for the support we received when working with their miniature vehicle.

REFERENCES

[ACIP2010] F. Arrichiello, S. Chiaverini, G. Indiveri, P. Pedone. The Null-Space-based Behavioral Control for Mobile Robots with Velocity Actuator Saturations. *The International Journal of Robotics Research, 29(10)*, 1317-1337, 2010.

[BBSP2002] G. Benet, F. Blanes, J. E. Simó, P. Pérez. *Using infrared sensors for distance measurement in mobile robots.* Robotics and autonomous systems, 40(4), 255-266, 2002.

[BS95] H. R. Beom, K. S. Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *Systems, Man and Cybernetics,* IEEE Transactions on, 25(3), 464-477, 1995.

[BCHLS2013] C. Berger, M. Chaudron, R. Heldal, O. Landsiedel, E. M. Schiller. *Model-based, composable simulation for the development of autonomous miniature vehicles*. In Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium (p. 17). Society for Computer Simulation International, 2013.

[B2010] D. Brink. *Essentials of Statistics*, David Brink & Ventus Publishing ApS, ISBN 978-87-7681-408-3, 2010.

[BKZ92] D. P. Brutzman, Y. Kanayama, M. J. Zyda. Integrated simulation for rapid development of autonomous underwater vehicles. *Autonomous Underwater Vehicle Technology AUV'92*, Proceedings of the 1992 Symposium on (pp. 3-10). IEEE.

[CB2002] A. Cao, J. Borenstein. *Experimental characterization of polaroid ultrasonic sensors in single and phased array configuration.* AeroSense 2002 (pp. 248-255). International Society for Optics and Photonics.

[CKW96] J. Cremer, J. Kearney, P. Willemsen. A directable vehicle behavior model for virtual driving environments. *Proceedings of 1996 Conference on AI, Simulation, and Planning in High Autonomy Systems,* La Jolla, CA. http://www. cs. uiowa. edu/~ cremer/papers/aisp-
final. ps.

[C2009] J. W. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2009.

[ElS98] M. El Shobaki. Verification of embedded real-time systems using hardware/software co-simulation. *Euromicro Conference*, 1998. Proceedings. 24th (Vol. 1, pp. 46-50). IEEE.

[FEZM2006] D. Floreano, Y. Epars, J. C. Zufferey, C. Mattiussi. *Evolution of spiking neural circuits in autonomous mobile robots*. International Journal of Intelligent Systems, 21(9), 1005-1024, 2006

[FDF2000] E. Frazzoli, M. A. Dahleh, E. Feron. Robust hybrid control for autonomous vehicle motion planning. *In Decision and Control,* 2000. Proceedings of the 39th IEEE Conference on (Vol. 1, pp. 821-826). IEEE.

[G92] E. Gat. *Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots*. In AAAi (Vol. 1992, pp. 809-815), July 1992.

[GPdSV2006] O. Gietelink, J. Ploeg, B. De Schutter, M. Verhaegen. *Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations.* Vehicle System Dynamics, 44(7), 569-590, 2006.

[HZ2010] A. Hore, D. Ziou. Image quality metrics: PSNR vs. SSIM. *In Pattern Recognition (ICPR), 20th International Conference* on (pp. 2366-2369). IEEE, August 2010.

[JHH95] N. Jakobi, P. Husbands, I. Harvey. *Noise and the reality gap: The use of simulation in evolutionary robotics.* In Advances in artificial life (pp. 704-720). Springer Berlin Heidelberg, 1995.

[KK96] I. D. Kelly, D. A. Keating. Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots. In *Proceedings of The Third Int. Conf. on Mechatronics and Machine Vision in Practice* (Vol. 1, pp. 1-4), 1996.

[KPBBTBL2009] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey,
S. Linkman. *Systematic literature reviews in software engineering – A systmatic literature review*. Information and Software Technology 51, 7-15, 2009.

[MBH2013] M. A. A. Mamun, C. Berger, J. Hansson. MIDE-based sensor management and verification for a self-driving miniature vehicle. In *Proceedings of the 2013 ACM workshop on Domain-specific modeling* (pp. 1-6). ACM, October 2013.

[PvGVVCTK2004] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch. *Visual modeling with a hand-held camera.* International Journal of Computer Vision, 59(3), 207-232, 2004.

[RNK97] P. Rander, P. J. Narayanan, T. Kanade. Virtualized reality: constructing time-varying virtual worlds from real world events. *Proceedings of the 8th conference on Visualization'97*. IEEE Computer Society Press, 1997

[RH2009] P. Runeson, M. Höst. *Guidelines for conducting and reporting case study research in software engineering*. Empirical Softw. Eng., vol. 14, pp. 131-164, 2009

[SSS2006] N. Snavely, S. M. Seitz, R. Szeliski. *Photo tourism: exploring photo collections in 3D.* ACM transactions on graphics (TOG), 25(3), 835-846, 2006.

[TH2003] S. N. Torres, M. M. Hayat. *Kalman filtering for adaptive nonuniformity correction in infrared focal-plane arrays*. JOSA A, 20(3), 470-480, 2003.

[UOK96] K. Umeda, J. Ota, H. Kimura. Fusion of multiple ultrasonic sensor data and imagery data for measuring moving obstacle's motion. In *Multisensor Fusion and Integration for Intelligent Systems*. IEEE/SICE/RSJ International Conference on (pp. 742-748). IEEE, December 1996.

[W2012] J. L. Waller. *How to perform and interpret chi-square and t-tests*, Georgia Health Sciences University, Augusta, Georgia, Paper 155-2012

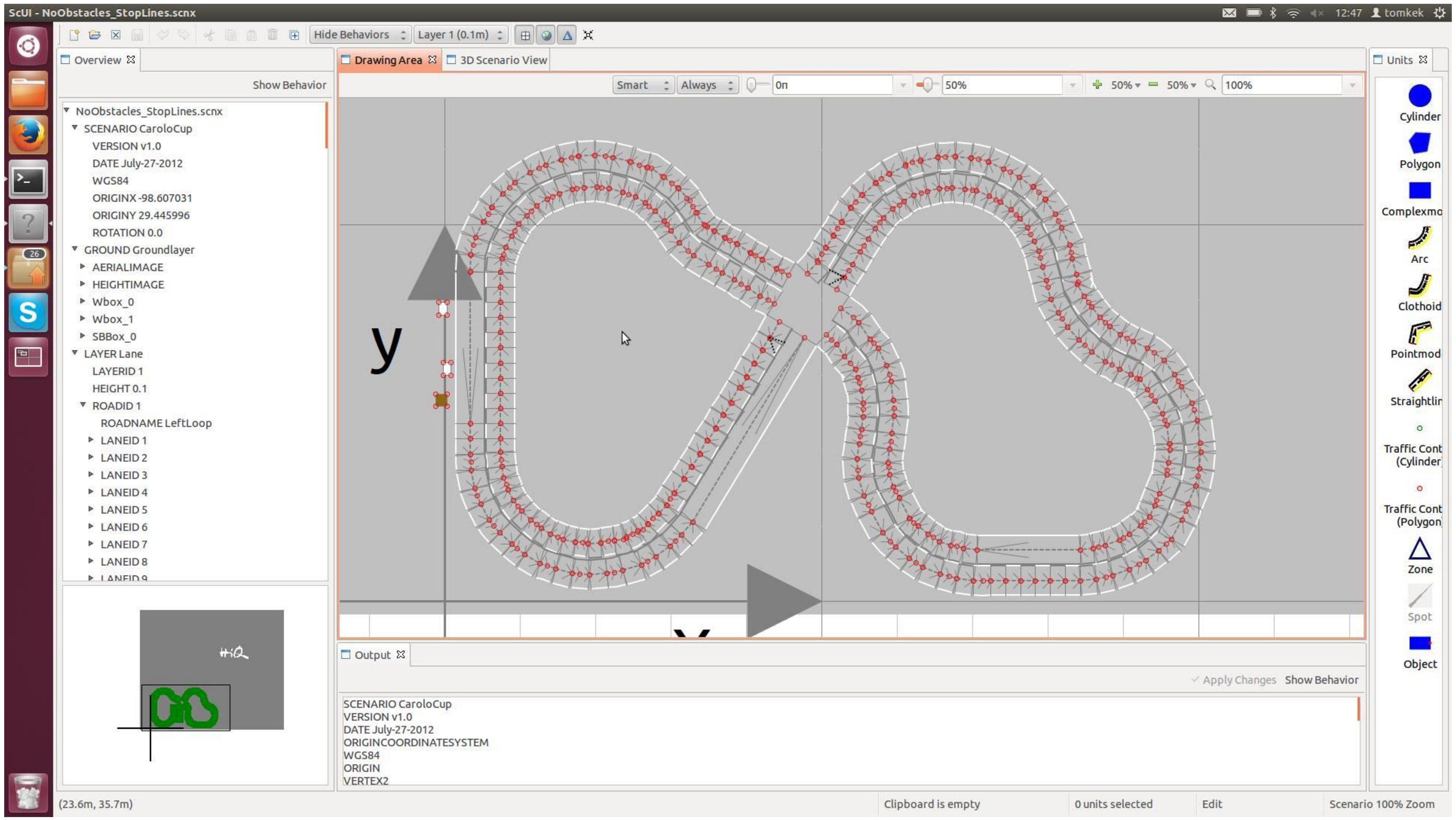
[WP2007] M. V. Woodward, J. Pieter. Challenges for embedded software development. In *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems/IEEE International NEWCAS* (pp. 630-633), August 2007.

[YS2012] A. Yusra, D. Soong. *Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQ*. International Journal of Scientific & Engineering Research, Volume 3, Issue 8, June 2012.

[ZW97] M. V. Zelkowitz, D. Wallace. *Experimental validation in software engineering*. Information and Software Technology, 39(11), 735-743, 1997.

[CC2014] Online information regarding Carolo Cup 2014; https://wiki.ifr.ing.tu-bs.de/carolocup/

Appendix 1



```
▼ NoObstacles_StopLines.scnx
  ▼ SCENARIO CaroloCup
      VERSION v1.0
      DATE July-27-2012
      WGS84
      ORIGINX -98.607031
      ORIGINY 29.445996
      ROTATION 0.0
  ▼ GROUND Groundlayer
    ▸ AERIALIMAGE
    ▸ HEIGHTIMAGE
    ▸ Wbox_0
    ▸ Wbox_1
    ▸ SBBox_0
  ▼ LAYER Lane
      LAYERID 1
      HEIGHT 0.1
    ▼ ROADID 1
        ROADNAME LeftLoop
      ▸ LANEID 1
      ▸ LANEID 2
      ▸ LANEID 3
      ▸ LANEID 4
      ▸ LANEID 5
      ▸ LANEID 6
      ▸ LANEID 7
      ▸ LANEID 8
      ▸ LANEID 9
```

```
SCENARIO CaroloCup
VERSION v1.0
DATE July-27-2012
ORIGINCOORDINATESYSTEM
WGS84
ORIGIN
VERTEX2
```

Appendix 2

SIMULATION AND HARDWARE IMAGES DATA RECORDING RESULTS

MSSIM value	PSNR value	Hardware timestamp	Software Timestamp
95.61%	26.59	012759190	184007272
95.75%	26.72	012759290	184007372
95.77%	26.62	012759390	184007486
95.91%	26.84	012759490	184007585
95.91%	26.84	012759590	184007672
95.85%	26.91	012759690	184007802
95.73%	26.69	012759790	184007872
95.61%	26.51	012759890	184007972
95.37%	26.38	012759990	184008071
95.24%	26.16	012800090	184008172
95.01%	25.97	012800190	184008285
94.86%	25.87	012800290	184008371
94.54%	25.53	012800390	184008472
94.46%	25.45	012800490	184008571
94.31%	25.39	012800590	184008685
94.27%	25.32	012800690	184008787
94.20%	25.13	012800790	184008920
94.23%	25.17	012800890	184008984
94.38%	25.37	012800990	184009087
94.41%	25.50	012801090	184009186
94.50%	25.55	012801190	184009286
94.50%	25.54	012801290	184009390
94.86%	25.76	012801390	184009505
95.01%	25.95	012801490	184009585
95.21%	26.16	012801590	184009703
95.41%	26.35	012801690	184009786
95.69%	26.62	012801790	184009884
95.74%	26.75	012801890	184009989
95.75%	26.70	012801990	184010084
95.61%	26.53	012802090	184010206
95.62%	26.54	012802190	184010284
95.25%	26.13	012802290	184010391
95.09%	25.97	012802390	184010487
94.88%	25.81	012802490	184010588
94.59%	25.58	012802590	184010698
94.30%	25.37	012802690	184010792
94.20%	25.28	012802790	184010888
93.97%	25.07	012802890	184010987
93.78%	24.80	012802990	184011084
93.81%	24.89	012803090	184011188
93.81%	24.99	012803190	184011304
94.01%	25.10	012803290	184011389
94.09%	25.21	012803390	184011490
94.35%	25.43	012803490	184011605
94.43%	25.56	012803590	184011684
94.62%	25.76	012803690	184011788
94.73%	25.88	012803790	184011885
94.72%	25.89	012803890	184011992

95.14%	26.27	012803990	184012088
95.38%	26.52	012804090	184012184
95.36%	26.47	012804190	184012284
95.27%	26.43	012804290	184012399
95.28%	26.48	012804390	184012490
95.15%	26.21	012804490	184012585
94.92%	25.99	012804590	184012691
94.70%	25.71	012804690	184012789
94.56%	25.64	012804790	184012889
94.25%	25.40	012804890	184012985
94.03%	25.21	012804990	184013090
93.76%	25.21	012805090	184013189
93.61%	24.92	012805190	184013284
93.59%	24.83	012805290	184013384
93.73%	24.88	012805390	184013484
93.72%	24.98	012805490	184013590
93.78%	24.97	012805590	184013685
94.07%	25.08	012805690	184013784
94.02%	25.32	012805790	184013909
94.02%	25.31	012805890	184014006
94.32%	25.31	012805990	184014084
94.55%	25.57	012806090	184014185
94.72%	25.75	012806190	184014295
94.64%	26.04	012806290	184014385
94.60%	25.85	012806390	184014484
94.55%	25.87	012806490	184014590
94.43%	25.83	012806590	184014690
94.21%	25.68	012806690	184014784
94.06%	25.38	012806790	184014884
93.57%	25.39	012806890	184014984
93.46%	24.95	012806990	184015086
93.09%	24.91	012807090	184015184
92.77%	24.66	012807190	184015284
92.59%	24.41	012807290	184015384
92.40%	24.32	012807390	184015484
92.33%	24.23	012807490	184015584
91.94%	24.24	012807590	184015686
91.89%	24.08	012807690	184015784
91.77%	23.97	012807790	184015891
91.72%	23.96	012807890	184015984
91.68%	24.00	012807990	184016108
91.78%	23.90	012808090	184016184
91.86%	24.08	012808190	184016292
91.94%	24.13	012808290	184016385
91.94%	24.24	012808390	184016484
91.97%	24.26	012808490	184016584
91.91%	24.34	012808590	184016684
91.91%	24.28	012808690	184016784
91.71%	24.37	012808790	184016885
91.89%	24.11	012808890	184016992
91.78%	24.31	012808990	184017088
91.76%	24.35	012809090	184017184
91.75%	24.42	012809190	184017284

```
91.61%           24.33           012809290              184017384
91.19%           24.20           012809390              184017484
91.07%           24.11           012809490              184017584
90.92%           24.05           012809590              184017684
91.35%           23.92           012809690              184017784
91.31%           24.11           012809790              184017893
91.29%           24.19           012809890              184017993
91.74%           24.01           012809990              184018084
91.75%           24.26           012810090              184018184
91.78%           24.31           012810190              184018293
91.48%           24.46           012810290              184018384
91.29%           24.10           012810390              184018486
91.29%           24.05           012810490              184018584

PSNR MEAN: 25.29 PSNR MEDIAN: 25.00 PSNR STANDARD DEVIATION: 0.87

MSSIM MEAN: 93.81 MSSIM MEDIAN: 94 MSSIM STANDARD DEVIATION: 1.45
```

Appendix 3

SIMULATION AND HARDWARE SENSORS DATA RECORDING RESULTS

IR 1	IR 2	IR 3	IR 4	US 1	US 2	Hd timestamp	Rd Timestamp
0	0	0	0	102	46.522	012759542	174405204
0	0	0	0	100	109.522	012759642	174405304
0	0	0	0	99	96.522	012759742	174405404
0	0	0	0	98	63.522	012759842	174405504
0	0	0	0	96	63.522	012759942	174405604
0	0	0	0	63	98.522	012800042	174405704
0	0	0	0	91	89.522	012800142	174405804
0	0	0	0	75	74.522	012800242	174405904
0	0	0	0	88	125	012800342	174405004
0	0	0	0	63	125	012800442	174406104
0	0	0	0	86	67	012800542	174406204
0	0	0	0	84	67	012800642	174406304
0	0	0	0	82	121	012800742	174406404
0	0	0	0	65	123	012800842	174406504
0	0	0	0	80	126	012800942	174406604
0	0	0	0	67	141	012801042	174406704
0	0	0	0	67	74	012801142	174406804
0	0	0	0	74	144	012801242	174406904
0	0	0	0.0001	74	135	012801342	174406004
0	0	0	0.0219	71	135	012801442	174407104
0	0	0	0.0219	67	73	012801542	174407204
0	0	0	0.9781	67	75	012801642	174407304
0	0	0	0.0219	67	75	012801742	174407404
0	0	0	0.0219	64	77	012801842	174407504
0	0	0	0.0219	62	82	012801942	174407604
0	0	0	0.0219	60	82	012802042	174407704
0	0	23	1.9781	58	77	012802142	174407804
0	0	19	23.0127	58	83	012802242	174407904
0	0	4.0015	0	58	86	012802342	174407004
0	0	4.0219	0	55	86	012802442	174408104
0	0	4.0219	0	55	81	012802542	174408204
0	0	5.0219	0	55	83	012802642	174408304
0	0	2.0219	0	54	93	012802742	174408404
0	0	2.0219	0	75	53	012802842	174408504
0	0	3.0219	0	79	93	012802942	174408604
0	0	3.0219	0	70	61	012803042	174408704
0	0	0.9781	0	75	57	012803142	174408804
0	0	3.9929	0	75	127	012803242	174408904
0	0	0	0	75	99	012803342	174408004
0	0	0	0	75	99	012803442	174409104
0	0	0	0	60	61	012803542	174409204
0	0	0	0	70	64	012803642	174409304
0	0	29	0	69	67	012803742	174409404
0	0	0	0	69	92	012803842	174409504
0	0	0	0	66	73	012803942	174409604
0	0	0	0	64	69	012804042	174409704
0	0	0	0	64	72	012804142	174409804
0	0	0	0	62	84	012804242	174409904
0	0	0	0	64	71	012804342	174409004
0	0	0	0	64	103	012804442	174410104
0	0	0	0	64	81	012804542	174410204
0	0	0	0	64	83	012804642	174410304

0	0	0	0	89	83	012804742	174410404
0	0	0	0	89	87	012804842	174410504
0	0	0	0	89	87	012804942	174410604
0	0	0	0	89	87	012805042	174410704
0	0	0	0	105	89	012805142	174410804
0	0	0	0	105	93	012805242	174410904
0	0	0	18.8272	105	96	012805342	174410004
0	0	0	6.9829	106	95	012805442	174411104
0	0	0	1.9829	95	76	012805542	174411204
0	0	0	1.9829	117	76	012805642	174411304
0	0	0	1.9829	117	104	012805742	174411404
0	0	0	1.9829	117	104	012805842	174411504
0	0	0	0.9829	103	104	012805942	174411604
0	0	0	0.9829	107	86	012806042	174411704
0	0	19	1.9829	78	68	012806142	174411804
0	0	19	2	114	111	012806242	174411904
0	0	4.0571	26	110	74	012806342	174411004
0	0	0.9829	0	121	115	012806442	174412104
0	0	0.0171	0	115	69	012806542	174412204
0	0	0.0171	0	125	72	012806642	174412304
0	0	0.0171	0	69	71	012806742	174412404
0	0	0.0171	0	69	75	012806842	174412504
0	0	0.0171	0	130	75	012806942	174412604
0	0	9.9829	0	113	78	012807042	174412704
0	0	18.0171	0	86	78	012807142	174412804
0	0	18.0009	0	86	78	012807242	174412904
0	0	0	0	93	85	012807342	174412004
0	0	28	2.8334	129	84	012807442	174413104
0	0	0	2.9871	129	84	012807542	174413204
0	0	0	3.9781	88	84	012807642	174413304
0	0	0	3.9781	132	84	012807742	174413404
0	0	0	4.9781	124	84	012807842	174413504
0	0	0	4.9781	124	93	012807942	174413604
0	0	0	2.9781	124	68	012808042	174413704
0	0	0	1.9781	124	58	012808142	174413804
0	0	0	23.0219	121	98	012808242	174413904
0	0	0	23.0117	121	100	012808342	174413004
0	0	24	0	75	100	012808442	174414104
0	0	1.0116	0	96	104	012808542	174414204
0	0	2.0219	0	81	107	012808642	174414304
0	0	1.0219	0	72	86	012808742	174414404
0	0	0.0219	0	72	106	012808842	174414504
0	0	0.0219	0	114	110	012808942	174414604
0	0	0.0219	0	114	98	012809042	174414704
0	0	0.0219	0	114	59	012809142	174414804
0	0	0.0219	0	73	59	012809242	174414904
0	0	1.0115	0	104	105	012809342	174414004
0	0	24	0	78	70	012809442	174415104
0	0	28	0	136	61	012809542	174415204
0	0	28	0	74	102	012809642	174415304
0	0	0	0	139	69	012809742	174415404
0	0	0	0	139	83	012809842	174415505
0	0	0	0	116	77	012809942	174415604
0	0	0	0	125	66	012810042	174415704
0	0	0	0	125	102	012810142	174415804
0	0	0	0	53.522	65	012810242	174415904
0	0	0	0	90.522	92	012810342	174415004

	0	0	0	0	19.522	92	012810442	174416104
	0	0	0	0	159	111	012810542	174416104
	0	0	0	0	95	67	012810642	174416104
	0	0	0	0	131	108	012810742	174416104
	0	0	0	0	112	108	012810842	174416104
MEAN:		0	0	2.88972	1.46064	89.2506	87.2559	
MEDIAN:		0	0	0	0	86	84	
STD DEV:		0	0	7.19093	4.74485	25.6315	19.9203	