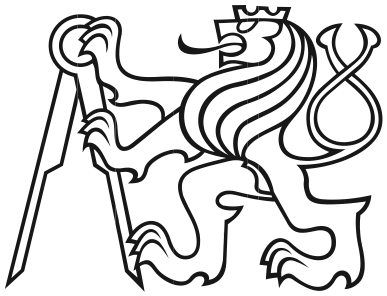

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of electrical engineering
Department of Computer Graphics and Interaction

Suppressing distractors from driver's field of view

Master's thesis

Study program: Human-computer Interaction
Field of study: Open informatics

Supervisor: doc. Ing. Zdeněk Míkovec, Ph.D.

Tomáš Kolařík

Prague 2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kolařík** Jméno: **Tomáš** Osobní číslo: **393241**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Potlačování distraktorů v zorném poli řidiče během řízení vozidla

Název diplomové práce anglicky:

Suppressing distractors from driver's field of view

Pokyny pro vypracování:

Prozkoumejte, jaké rušivé předměty řidič vnímá ve svém zorném poli při řízení vozidla. Prozkoumejte způsoby detekce vybraných rušivých předmětů. Vybraný způsob se pak pokuste implementovat. Dále prozkoumejte možnosti blokovat nebo potlačovat tyto detekované předměty. Při výběru metod potlačení se zaměřte na to, jak jsou rušivé, invazivní a jaká je míra jejich účinnosti. Vytvořte prototypy, které průběžně vyhodnocujte uživatelským testováním a v případě možnosti využijte simulátor auta.

Seznam doporučené literatury:

Colin Ware: Visual Thinking for Design, Elsevier Inc., 2008
Colin Ware: Information Visualization - Perception for Design, Elsevier Inc., 2004
Alan J. Parkin: Essential Cognitive Psychology, Psychology Press Ltd., 2000
Sonka, V. Hlavac, R. Boyle. Image Processing, Analysis and Machine Vision. Thomson 2007
A. Forsyth, J. Ponce. Computer Vision: A Modern Approach. Prentice Hall 2003
Šonka M., Hlaváč V., Boyle R.: Image Processing, Analysis and Machine vision, 3rd edition, Thomson Learning, Toronto, Canada, 2007

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Zdeněk Míkovec, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.02.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

doc. Ing. Zdeněk Míkovec, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání Podpis studenta

Declaration

„I declare that this thesis is my own work and that I cited all information sources according to Methodical instructions for adhering with ethical principles for creation process of final College theses.”

Prague, 2018

Preface

I would like to thank my thesis supervisor doc. Ing. Zdeněk Míkovec, Ph.D. for steering my work in right direction and helping me with the concept of the work. I would also like to thank Faculty of transportation and science for allowing me to use the simulator setup for testing my methods and I would especially like to thank Jan Válek from Faculty of transportation for helping me with the simulator setup and to create the training data.

Abstract

First, I review the distraction caused by roadside advertisement. In the work, I focus on billboards. I name most common billboard types and the distraction they cause.

Then, I create two methods to potentially reduce the distraction – blackening the content of the billboard and hiding the billboard in the scene using methods like content aware fill. After establishing methods, I create a simulator setup to test these two methods and then compare the results.

After evaluating the results, I create a proof-of-concept implementation using publicly available computer vision framework and the training data extracted from simulator testing.

Key words

Car, driving, billboards, distraction, advertisement, blocking, distractors, virtual reality, simulator testing

Anotace

V práci je čtenář nejdříve seznámen s problematikou reklamy podél silnic a distrakcí, kterými na řízení působí. V práci samotné se potom soustředím konkrétně na billboardy. Vyjmenuji jejich druhy a specifické vlastnosti.

Poté navrhuji dvě metody redukce distrakcí způsobených billboardy – začernění obsahu a skrytí billboardu ve scéně pomocí metod jako například Content Aware Fill. Po definování těchto metod tyto dvě metody testuji pomocí simulátoru řízení a potom vyhodnocuji závěry z tohoto testování.

Na závěr ukazuji implementaci výsledné metody z předchozí kapitoly pomocí veřejně dostupného projektu zaměřeného na počítačové vidění a trénovacích dat extrahovaných z testování na simulátoru.

Klíčová slova

Auto, řízení, billboardy, distrakce, reklama, blokování, distraktory, virtuální realita, testování na simulátoru

Table of contents

List of figures

1 Introduction

Driver distraction is a cause of many crashes, with some of them being lethal. In fact, 25% of all crashes are caused by distractions according to the National Highway Traffic Safety Administration statistics [1] from 2000. The 100-Car Naturalistic Driving Study [2] from 2006 even shows, that 78% of all crashes are caused by distractions (distraction in this case meaning variety of factors including fatigue, looking in mirrors etc.) Also, while other causes of crashes like alcohol are on decline, distraction related crashes are on a rise as newer cars have more and more things distracting the driver, like in-car build-in multimedia systems. Phone distraction is also a huge factor, as not using a phone for a longer period of time is problem for an increasing amount of people.

Distraction in this case is defined as a process of diverting the attention from a desired area of focus and thereby blocking or diminishing the reception of desired information. In the case of driver distraction, we can divide the distraction factors into two basic groups - in-car distractions and outer distractions.

Most studies focus only on the first group, the in-car distractions. Probable reason is that crashes caused by these types of distractions are in majority. According to study External-to-vehicle driver distraction [3], chance of crash caused by in-car distraction is 1.25 times more probable than crash caused by outer distraction. In-car distractions are also easier to control than outer ones. To limit the in-car distraction, we can change the dashboard interface, remove distracting functions, we can allow users to use voice control instead of for example a touchscreen input and other examples. Outer distraction on the other hand are much harder to control.

Although harder to control, with the recent advances in computer vision and computational capabilities as a whole, there is an increasing number of ways to manipulate or control this environment in way that it is less distracting for a driver. Also modern cars have ways of detecting distracted driver by position of his eyes or by checking the position and speed difference to other cars. These car systems can then react and help prevent dangerous situations and potential crashes.

Target of this study is to find, whether it is possible, to reduce or to block harmful effects of external distractions on drivers, more specifically roadside advertisement and to do so possibly without introducing other harmful effects [4] and then create sample implementation of the selected approach. The best approach is the result of user testing.

Roadside advertisement in this case is defined as a device or a sign visible to road users that is used to promote services, products, places, businesses or organizations and serve purpose of bringing benefit from displaying the advertisement.

Typical advertisement sign cannot be easily defined. It varies considerably both in the form of the sign itself and the content of the sign. There are different signs along highways and in the cities (Figure 1). Highway signs are usually larger and further away from the road then advertisement signs near normal roads. The content itself is also very variable and changes regularly.

Figure 1 Normal road advertisement sign versus highway sign [5]

My hypothesis is that by reducing visibility of a billboard in the field of view by covering it up with other content effectively hiding it in the scene in way that is as natural in the particular scene as possible, will make it less distracting for drives despite introducing other possible distractions like artifacts and other possible harmful effects.

The second hypothesis is that removing only the content of a billboard will reduce the distraction caused by this billboard, because it will remove the motivation for a driver to look at such a billboard, even though the billboard itself will still be visible.

2 Problem description

2.1 Advertisement signs

2.1.1 Static advertisement signs

Static signs are the most common type of signs. The reasons are lower cost and less complexity compared to dynamic sign. The content does not change automatically and there is no visual movement in the content of the sign. These signs vary from small to large dimensions. From a distraction standpoint, they are less harmful and they draw less attention and for shorter amounts of time.

2.1.2 Dynamic advertisement signs

These signs change content automatically or have animated or movable content. These signs are typically alongside more frequented roads as the cost is higher compared to static signs. These signs are much worse distraction-wise, as they draw the attention much more aggressively and for longer time, as discussed later in this thesis. There are typically three principles these signs use. [4]

First is sign with vertical adjacent prisms with three sides that rotate, displaying one of three pictures (Figure 2). These are typically smaller than other dynamic signs, due to the construction limitations (typically 3m x 6m) [4]

Second is scrolling signs (Figure 2). These have a scroll of vertically adjacent ads that spools alongside vertical axis so that the spool displays the ads sequentially on the display. These are typically smaller than previous group of prism signs again due to construction limitations and are typically placed alongside smaller streets on street level and are typically illuminated.

Figure 2 Rotating prisms sing and scrolling sign [5] [6]

Last type is digital displays (Figure 3). These are the most recent ones and by far the worst distraction wise. They can show static images or animated sequences or animated individual images or videos. They are typically greater than 4 square meters [4] and the individual pixels are made out of small led lights.

Figure 3 Pixels consisting of led lights [7]

These signs (Figure 3), especially when used with animated content, cause the most distraction, because the animation together with very high level of illumination make this type of advertisement sign most attention heavy.

For example, on Prague's most busy road called 'Pražská magistrála', after installing dynamic led billboards (Figure 4), ŘSDP (Czech road and highway headquarters) reported that the number of crashes in the proximity of these billboards was 35% higher than without the billboards. [6]

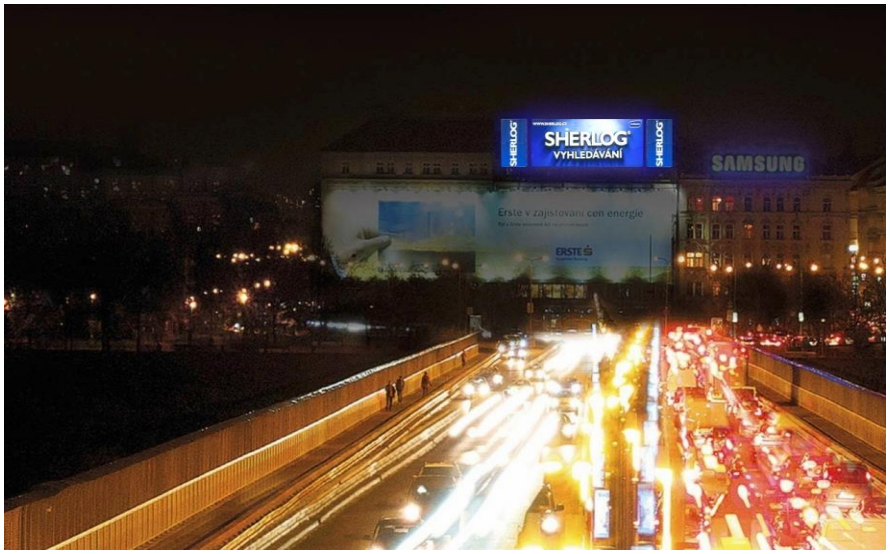

Figure 4 Led billboard located on 'Pražská magistrála' [9]

2.1.3 Legislation in Czech Republic

Allowance for installing traffic signs is issued for 5 years and must comply with these rules:

1. They cannot be designed in a way that they can be mistaken with traffic signs
2. The signs cannot glare the traffic users or disturb the traffic in any other way
3. The sign must put behind crash barrier or made safe in a way of vehicles crashing into the sign
4. Signs on highways and main roads can be used only to promote local businesses maximally 50 meters from the sign itself in a case of build-up areas or 200 meters otherwise.

These have to be valid for the whole 5 years of issued allowance, otherwise the signs must be taken away on the expanses of the owner of the sign.

2.2 Distractions while driving

Again, distraction is defined as a process of diverting the attention from a desired area of focus and thereby blocking or diminishing the reception of desired information. [7] In the case of driver distraction, we can divide the distraction factors into two basic groups - in-car distractions and outer distractions. This is only one of many possible ways to divide driving distractions into groups. Another possible grouping could be manual distractions like eating or smoking, visual distractions like changing radio station and cognitive distractions like talking to passenger. [8] In my work, I use the in-car distractions and outer distractions groups.

As already mentioned, according to studies, distractions cause big percentage of crashes, depending on study, from 10% [1] all the way to 78% [2]. The big difference between study results is caused by different definitions of distraction. Some studies' distractions include factors like fatigue or day dreaming, while other studies do not include these factors. But even the low estimate is still a big number.

When we take a look number of in-car distractions versus outer distractions, in-car distractions are about two times more probable than outer distractions according to study conducted by Stutts et al. (2001) for AAAFTS [9], but this number varies a lot trough out different studies. If we take this number together with the fact that in-car distractions like entertainment systems or mobile phones are easier to control means that most studies that focus on reducing or controlling distraction focus on in-car distractions.

Two most common in-car distraction according to 100-car naturalistic study [2] were other car occupants and using other device brought into vehicle. In case of external extractions,

studies do not typically divide this group into specific distraction groups. However, there are studies that focus specifically on one type of distraction.

Some studies, focusing specifically on billboard distraction define distraction as a glance at billboard longer than 0.75 seconds (minimum perception–reaction time) [10] as this number is a border value that may cause a dangerous situation. According to systematic literature review by US National Institutes of Health, 10-20% of glances on billboards are longer than 0.75 seconds. They also show, that in case of dynamic billboards, both glances longer and shorter than 0.75 were more often than in case of static billboards.

Also, according to aforementioned study, the danger factor of distraction is very variable. Despite already mentioned length of glance, the situation were the distractions appears have a big impact on how dangerous the distraction is.

In environment were driver expects all driver related stimuli and situations like following a lead car or to concentrate on traffic signs. However, in situations were driving task suddenly and unexpectedly becomes more difficult after a period of relatively low complexity, these distractions may cause a considerable risk.

3 Distraction reduction methods

To reduce the effect of distractors, billboards in this case, I came up with two methods of reducing their effect.

First method is to make the billboard itself as invisible as possible. Methods like content aware filling (Figure 6), filling the billboard with colors from the edges of the billboard or other alternatives. Making billboard invisible using these methods together with detection in a real environment is impossible, especially if we add motion. There will always be some artifacts (as seen on the example below). These could defeat the purpose of the distraction reduction as they would cause additional distraction themselves. Another problem could also be detecting the whole billboard (not just the billboard content) in the scene if we would want to implement this method in a real environment. Detecting the content itself is an easier task. The billboard body and legs are more variable and many of them do not share common features that could be used for their detection.

Second method is to blacken only the content of the billboard (Figure 5). Blackening the content is an easier task compared to the first method. There is also lower chance of artifacts appearing, as the artifacts in real environment would be introduced only by detection errors, not by the method itself.

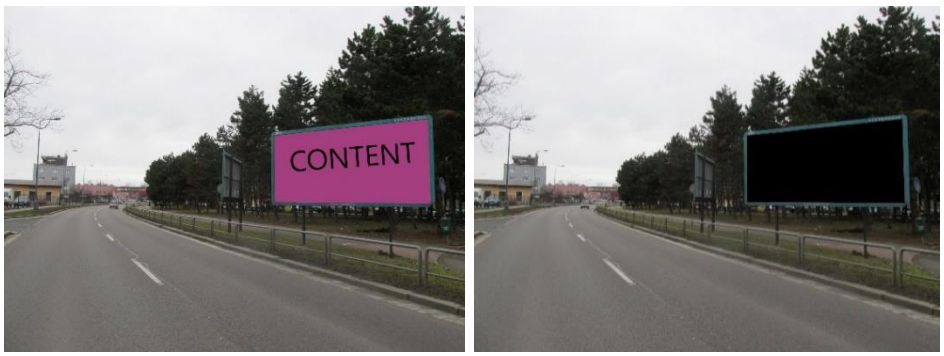

Figure 5 Normal billboard with content, Billboard with blacken content

Figure 6 Billboard removed using content aware fill method

4 Simulation experiment

Before implementing the discussed methods, I decided to do a simulation of these two methods to determine their effects in an ideal situation, find out if they actually help to reduce the distraction and to find out which of these methods is better to implement it later.

4.1 Environment

To collect the data, most studies use controlled environments to measure and observe the driving. Study Assessing Cognitive Distraction in the Automobile [1] shows that there is some difference in focus and distraction between real-life driving and driving in a simulated environment but the difference is not significantly different to affect the results of the researches (Figure 7).

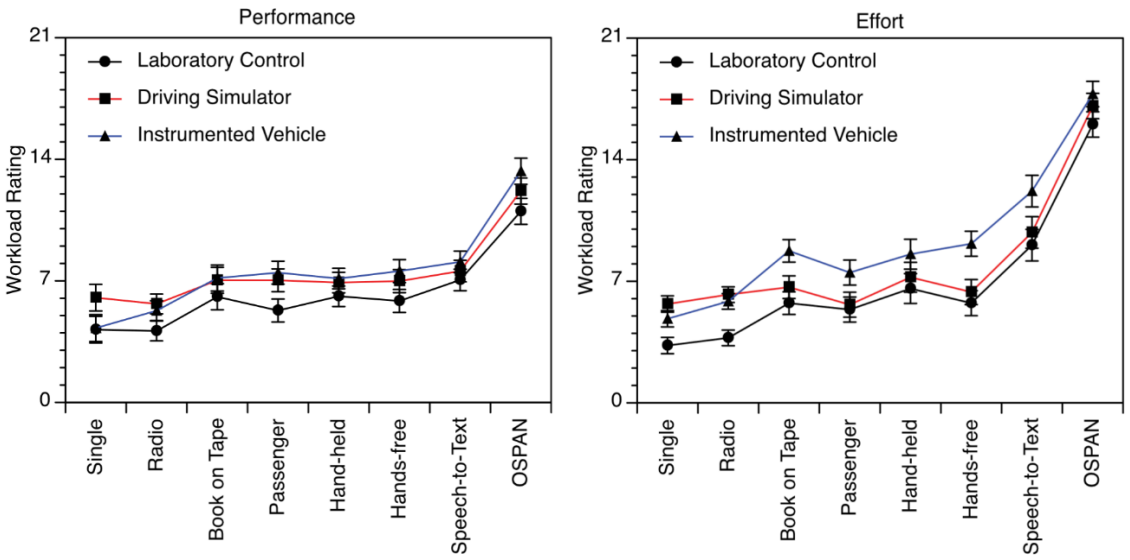

Figure 7 Environment comparison [1]

The same study [1] shows that the same tasks conducted in laboratory, simulator and real car have a different cognitive load in each environment, although the performance is very similar (Figure 8). The result of this study is, that both the simulator and laboratory testing is ecologically valid – as stated in the study – "*lessons learned in the laboratory and driving simulator are in good agreement with studies of cognitive distraction on the road-way*".

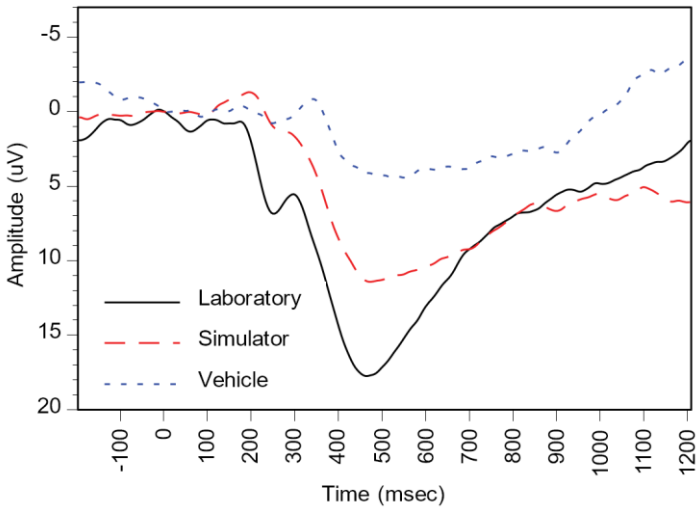

Figure 8 Cognitive load based on environment [1]

Getting the data in a real driving environment is also very expensive. Apart from cameras that collect the data regarding the driver attention, which are common for both simulated and real driving environments, there needs to be additional sensors installed to collect the data. Sensors like accelerometers are needed to identify the data interesting for the study [2]. Also, to collect data from as many participants as possible, all the data collection equipment needs to be installed into multiple cars. For example, in the case of NTDS study [2], 100 cars had the data collection equipment installed. Also, there could be legal issues depending on the country of the study.

Second option is to use car simulator. All the controls are very similar to real car and there is higher amount of immersion compared to laboratory testing. All of this, together with other factors means that there is less cognitive load on the participants of the study. The environment is also controllable. We can for example generate hazardous situations and measure the participant's reaction, which cannot be done in a real car. As we are responsible for all the simulation creation, we know what is happening on the road at any given time. This means we do not need to measure nearly as much information as in the case of a real car.

Third option is regular laboratory study. This has the advantage of being the cheapest and fastest way. There is no need for special equipment that isn`t useable for other studies. We have the same amount of control as when using the simulator, but the immersion is lacking behind the previous two options.

As the graph shows, simulator requires up to 2 times more cognitive load than driving real car and laboratory testing requires up to 4 times more cognitive load than real car driving [1].

4.2 Participants

Driving experience may also affect driver's cognition. Beginner drivers are the most likely to crash compared to other groups of drivers. Although the number of glances at external distractions are comparable in both groups, the effects are different.

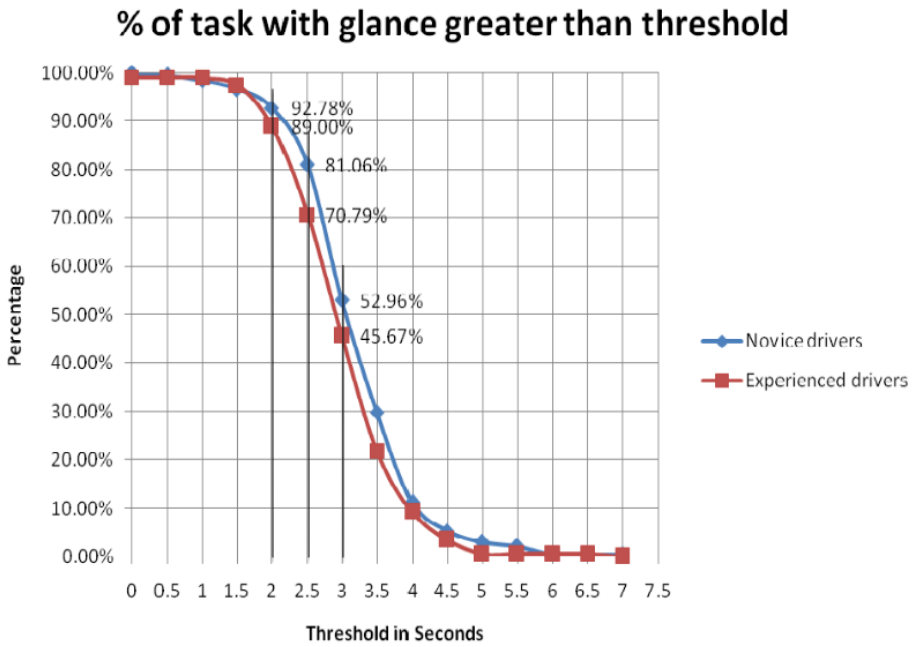

Figure 9 Glance comparison of novice and experienced drivers [11]

In a study The Effect of External Distractions on Novice and Experienced Drivers [11], novice drivers exceeded lane boundary in 26% of scenarios compared to 4% for the experienced drivers (Figure 9).

The next graph (Figure 10) shows the percentage of novice and experienced drivers who recognized hazards during test scenarios. Again, experienced drivers recognized by average 58% of hazards while novice drivers recognized 43%. Difference of 15%.

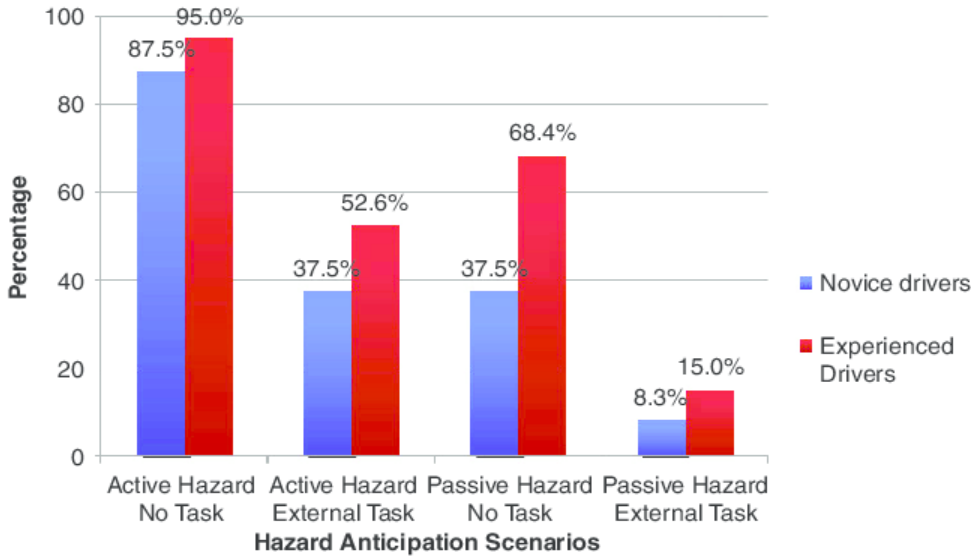

Figure 10 Comparison of hazard anticipation between novice and experienced drivers [11]

For example, as stated in the aforementioned study - "*long glances away from the forward roadway did not affect the lane and speed maintenance ability of the experienced driver but this was not completely true for the younger novice drivers* ". [11]

All these factors need to be addressed when choosing the participants or when evaluating the data. Different driving experience of drivers in different tests could cause error in result data.

Age is not the only factor that has a role in influencing driving capabilities. Some studies show for example differences in gender and other factors, although these factors should not be significant enough to influence the results of this experiment.

4.3 Experiment setup

To collect the data and conduct the experiment, I am using driving simulator Octavia II located at Faculty of Transportation Science, Czech technical university.

4.3.1 Simulator description

Simulator consists of front half of a Skoda Octavia. It has original dashboard including most of control elements like steering wheel, shifter etc. It also has original seats, a-pillars and closable doors. The transmission is automatic so that user is not distracted by shifting. Original speedometer and tachometer show real data from the simulation. The steering wheel also has force feedback based on the simulation. This all adds to the immersiveness, although the lack of centrifugal forces is counter intuitive in the beginning as it makes it much harder to tell the driving speed of the car, breaking speed or degree of turning just by looking at the road and

driving as in a real car. This should be taken in account when testing with participants and let them get used to these differences before starting the relevant tests.

The system of the simulator consists of 4 parts, each of which is proprietary for this simulator.

1. Physics simulation system.
2. Audio and visual rendering system. The rendering part is quite simple. It has one global shader that defines the rendering of the whole scene. Scene itself is defined as a normal 3D scene with triggers, that can turn on or off some of the actions like moving a car based on the location in the 3D scene. This mean that it can display only basic types of visual simulations as it lacks more advanced features like illumination.
3. Input and output system. This system takes care of inputs like steering wheel rotation, gas and brake pedal position or handbrake. It also outputs information to in-car equipment like speedometer.
4. Logging system. This part saves the data collected during simulation to devices connected to the simulator.

These systems communicate with each other using a UDP based protocol trough network and have control system with user interface to control the simulator (Figure 11).

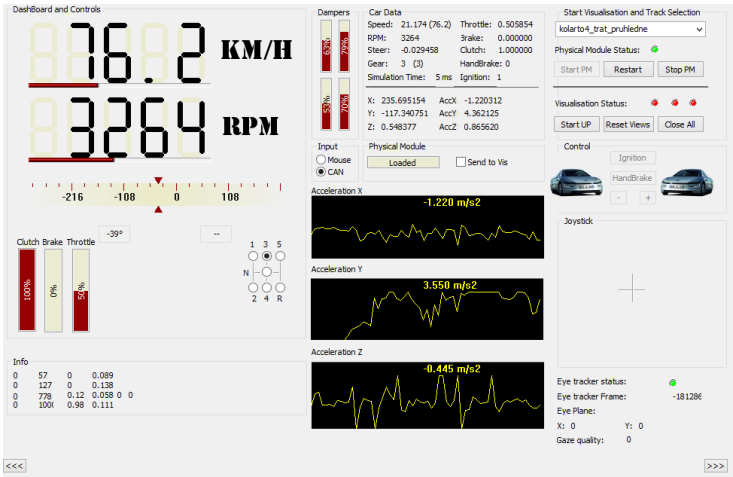

Figure 11 Simulator control center

There are 3 back-projection screens around the car cockpit that display the simulation (Figure 12, Figure 13). One in front and two on both sides. Each of the screens has Full-HD resolution of 1920x1080 pixels and 60 Hz refresh rate. [12]

Figure 12 Simulator setup [15]

Figure 13 Internal components connection and back projection

4.3.2 Practicing driving

Before testing itself, each participant was given about 5-10 minutes of practice driving to get used to driving using this simulator. The most unnatural thing when driving a simulator is lack of centrifugal forces. It is hard to estimate speed and braking distance when there is lack of these forces. This is eliminated by the practice driving as people get used to looking more at speedometer instead of estimating their traveling speed like when driving normally.

One of the problems may also be the sound of the engine while driving, which a lot of people use to estimate they're speed and it might be different in the simulator than in the car they are used to.

4.3.3 Eye tracking

For tracking the participant's eyes, a helmet with two mounted cameras is used to track the users viewing target and viewing fields (Figure 14). The result is a video with marker of the

viewing target. From this data, test results regarding the time of the participants looking at each billboard is extracted.

The device needs to be calibrated for each of the participants. The front looking camera needs to be pointed towards the front view of the participant while he remains in a natural driving position. The area of interest – road ahead, should be in the center of the view of this camera where tracking is most accurate. Second camera needs to be looking at the eye of the participant through a semi reflective glass with infrared light attached to it and shining in the eye from the same direction as camera recording the eye.

The camera looking at the eye detects position of the pupil compared to the center position of the infrared light and computes a vector, which is used to find a location in the view of the second front facing camera based on the calibration.

The calibration itself is done in a special software. The participant cannot move his head during calibration. On the connected computer, there are by default 5 points presented in a field of view of the front camera. One in each corner of the field of view and one in the center. Participant then needs to look at each of them. Also brightness and dynamic range needs to be set so that eye pupil is detected all the time. After all the 5 points are confirmed, it is better to validate the calibration (best working and fastest way I found was to use a laser pointer which the participant followed with his eyes and checking the tracking of the laser pointer on the computer screen) and in case it is not perfectly accurate, repeat all the calibration steps (Figure 15).

Figure 14 Eye tracking helmet

Figure 15 Eye tracking result, the red cross is the point of view

4.3.4 Simulated track

The track used in this experiment is based on track from thesis Driver's useful field of view [12] (Figure 16).

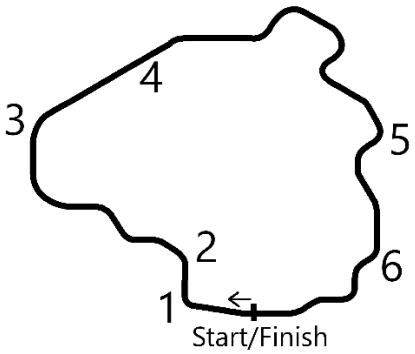

Figure 16 Map of the testing track

The track is 5264 meters long and there are 20 curves. The surrounding is neutral - no heavy traffic or crossroads but mostly natural environment to minimize the unwanted disturbance for the person to drive through the track and finish the task. The driver is supposed to trail a lead car that is going about the speed of legal speed limit, but he is not required to do so and can overtake the car if he wants to.

There are 6 places with billboards in total. Two places (1 and 5) have two billboards side by side, other places have only one billboard. They are placed so that only one place can be seen at any given time and all of them are visible at least from 200m away.

The content of the billboards is selected so that it is similar to the real world billboard content and is represented by ads for fictional products like phones, bank services, perfumes etc. The billboards are 50% bigger compared to a real billboard. This had to be done to make the billboard ads readable from a larger distance due to lower resolution of simulator projectors. [12] Also, all the billboards used in this experiment are static due to simulator limitations.

4.3.5 Tasks

Participant completes the same track 3 times. First time with billboards fully shown. Second time with billboards hidden from a further distance gradually increasing transparency to the point, where they are barely visible (about 85-90% transparency) (Figure 18). Lastly, the content of the billboard will be gradually blacken out at the same distance as the transparency. The order of these 3 tracks for each participant is chosen using Latin square counterbalancing. This ensures that the order of the tracks does not affect the experiment results. The distance, where transition of the billboards from normal rendering to black or transparent rendering begins is set to the point maximally 300 meters away from the billboards itself and at least 1/3 of the billboard must be visible. This simulates limitations of image recognition in real environment (Figure 17).

Figure 17 The environment of the track

In case of the track with transparent billboards, the billboards do not disappear fully, but they are left barely visible. Billboards visibility is controlled using alpha channel. The target visibility is 15-20%. This simulates the fact that we are not able to fully remove the object from visual field in real application either (Figure 18).

Figure 18 Reducing distraction using transparency vs. blackening the content and comparison with original billboard

The result of these tasks is the eye-tracking data and times with how long each participant spent looking at each billboard together with questionnaire, where the participant will answer questions about the content and amount of billboards he detected during each round (questionnaire in Czech language added at the end of this document).

4.3.6 Procedure

The testing procedure was kept as similar between different participants as possible. When participant arrived, all he knew was that he will be driving using a driving simulator. After arriving, he was shown around the simulator room. He was explained how the simulator and all its controls work. Also, the whole procedure was explained including the questions after driving, eye tracking, driving itself, how long will the test take and what data will be collected.

Before the start of the testing, each participant was given about 5-15 minutes of practice driving to get used to driving using this simulator. The most unnatural thing when driving a simulator is lack of centrifugal forces. It is hard to estimate speed and braking distance when there is lack of these forces. This is eliminated by the practice driving as people get used looking more at speedometer instead of estimating their traveling speed.

One of the problems may also be the sound of the engine while driving, which a lot of people use to estimate they’re speed and it might be different in the simulator than on the car they are used to.

The participants were instructed to drive as long as they want and say when they feel they are able to keep the speed they want and place and steer the car exactly where they want.

After practice driving, they were given the eye tracking helmet and the helmet was calibrated. A test track was loaded where we asked the participants to look at specific objects on the screen and checked if the tracking is accurate. Then the testing track was loaded and the participant was instructed to drive the same way as he would drive a real car in a real environment - not going over speed limit, stopping at traffic lights etc.

After he finished the first round, second track with different billboard rendering was loaded and the participant again finished the track and finally the same with third track. The order of the three methods was chosen using Latin square counterbalancing method to remove the learning effect on the resulting data.

Each of the tracks took about 5-6 minutes to finish depending on driving speed of the participant. Including all the track loading and driving, participant spend about 25 minutes with the eye tracking helmet on, which seemed like an upper limit of being comfortable with the helmet on. Participant could not move the helmet on his head because it would have ruined the calibration. This, combined with the heat of all the projectors and computers in one room meant more than half an hour would probably be uncomfortable.

After each round, the eye-tracking data was saved as the new track was loading. After all three rounds the participant was instructed to take off the helmet and we talked about the testing and the tracks. I revealed the subject of the experiment and asked him about his opinion about each of the 3 rendering methods. I had a questionnaire prepared so that the questions asked for each participant are exactly the same. Expect the aforementioned opinion about the three methods, we also asked question “How many billboards have you seen in each round?” to which the right answer was 8 in all of them. I also asked each participant a question about the billboard content - “Do you remember content of some of the billboards and from what testing round?”, but because only some participants were able to recall the content and all of them from the track with normal rendering, I did not use the question in the testing results as I consider it not relevant. As we finished discussing the questions, the testing was ended.

Including all the parts of the procedure, each participant testing took from one to one and a half hour.

4.3.7 Participants

The only requirement for participants was a previous experience with driving a real vehicle. Participants were not aware of what the target of the experiment is before the end of

the driving simulation and where told to drive as if they were driving their own car in real environment.

The experiment was attended by 15 people, 10 of which were used in this experiment. The data from four of the attendants were not used, because it was corrupted during the testing. The eye tracking system was not sufficiently tracking their eyes which resulted in missing or false data. One attendant could not finish the experiment due to motion sickness.

Participants were recruited from a student groups and using contacts provided by these students.

One of the problems encountered during testing was eye tracking with glasses. The problem was that glasses reduced the dynamic range of the resulting image of the camera, because the lens reflects some of the light. Because of that, camera could not detect the iris properly and all participants had to drive without glasses.

5 Results of the experiment

To determine the results, I used open source media player VLC. It can slow down the video or play the video frame by frame. If we do not take into account the latency of the eye tracking, the theoretical maximal error is about +-0.04 seconds because the eye tracking video is recorded at 25 frames per second. Each look at billboard was counted from the time participant's view crossed the outer border of the billboard and ended when the view left the billboard. Minimal viewing time was set to 0.1 seconds to eliminate counting in situations, where participants wants to look at something next to the billboard and just quickly moves his point of view over the billboard are to look at it.

Participant	1. bb	2. bb	3. bb	4. bb	5. bb	6. bb	7. bb	8. bb	Billboard rendering
1	3,6	2,7	1,1	1,6	0,7	1,4	2,2	3,2	normal
1	2,2	0,5	0,9	0,1	0,2	0	0,1	0,8	black
1	1	1	1,8	0,9	0,9	0,9	0,3	1,2	transparent
2	0,4	0	0,1	0,2	0,2	0	0	0	normal
2	0,4	0,3	0,3	0	2	0	0	1,7	black
2	0,8	2,1	0,6	0	0,9	0	0	1,4	transparent
3	4,6	1,7	2	0,3	2,5	0,7	1	0,5	normal
3	0	0	0	0,2	2,1	0	0,5	0,1	black
3	1,8	2,8	0,9	1,4	1,7	0,2	0	0,3	transparent
4	1,1	0,4	0,7	0	0,2	0	0	0	normal
4	0,1	0	1,1	0,6	0,8	0	0	0	black
4	2,3	0,3	0,9	0,2	0,6	0	0	0	transparent
5	3,3	0,5	1,3	1,9	2,3	0,4	1,2	3,1	normal
5	2,5	0	0,3	0	0,1	0	0	0,2	black
5	2,6	1,3	0,6	1,2	2,1	0,3	0,1	2,1	transparent
6	1,6	1,5	1,6	1,4	0,3	0	0	0,6	normal
6	1	0	0,2	0,1	0,3	0	0	0	black
6	1,1	0,2	0,5	0	0,7	0	0	1,1	transparent
7	0,2	0,7	1,5	0,7	0,6	0	0	1,2	normal
7	2,1	0,8	0,5	0,6	1,5	0	0,3	0,3	black
7	0,6	0,4	1,4	0,4	0,4	0,2	0,2	2	transparent
8	3,1	1,4	2,5	1,3	1,4	1	1,1	1,4	normal
8	0,2	0,3	0,4	0,1	0	0,1	0	0,9	black
8	1,8	2,6	1,8	0,4	1,6	0,2	0,2	1,4	transparent
9	0	0,1	0,9	0,3	2,6	0,6	1,3	2,6	normal
9	0	0	0	0,2	2	0	0	0	black
9	0,1	1,6	1,1	0,7	0,9	0	0,6	0,3	transparent
10	2,8	1,4	2,3	1,6	2,8	0,9	1	1,5	normal
10	0,9	0,4	0,6	0,3	0,5	0	0,1	0,4	black
10	1,7	1,9	1,2	1	1	0,4	0,5	1,2	transparent

Figure 19 Results of eye tracking results per billboard and participant

After testing all the participants, manually collecting all the data from the eye tracking results and questioners and excluding unsatisfactory data (Figure 19), following results were collected.

The average time spend looking at billboards was counted depending on the type of the billboard rendering. As expected, the highest time was achieved when rendering the billboards normally, with content and without transparency. The time participants spent looking at all billboards throughout the whole testing with all participants when rendering billboards normally was 94,9 seconds and participants spend 1.19 seconds looking at one billboard on average. These are the base numbers to compare the two other methods with.

The transparency method's time spent looking at the billboards throughout the whole testing was 70,9 seconds while each participant spent looking 0.89 seconds on each billboard on average, which is 25.29% decrease compared to the normal rendering. One of the participants mentioned after he finished driving – "*I tried to recognize, what was on the billboard, but I could not tell.*" Some of the participants even spend looking more time on some of the transparent billboards then the normal billboards. The transparent billboard is not a natural thing drivers are used to and attracts attention even if the content itself is not recognizable. Long term testing could bring a different result as getting used to this type of object in a field of view could reduce motivation to look at transparent billboards.

Last method was to blacken the content of the billboard, leaving the billboards itself visible but removing the content of the billboard. This method's time spent looking at billboards of all participants throughout the testing was 33.2 seconds and average time spent looking at one billboard was 0.42 seconds, which is the smallest measured number and is 65,02% smaller than in case of normally rendered billboards. Some of the participants said that they looked at first few of a billboards and then stopped looking at them as they knew the billboards have no content. From the interviews of the participants and also from the numbers, this method looks more natural for the participants compared to the transparent billboards and does attract less attention. For drivers, these where simply billboards with no content so they had no motivation to give them attention.

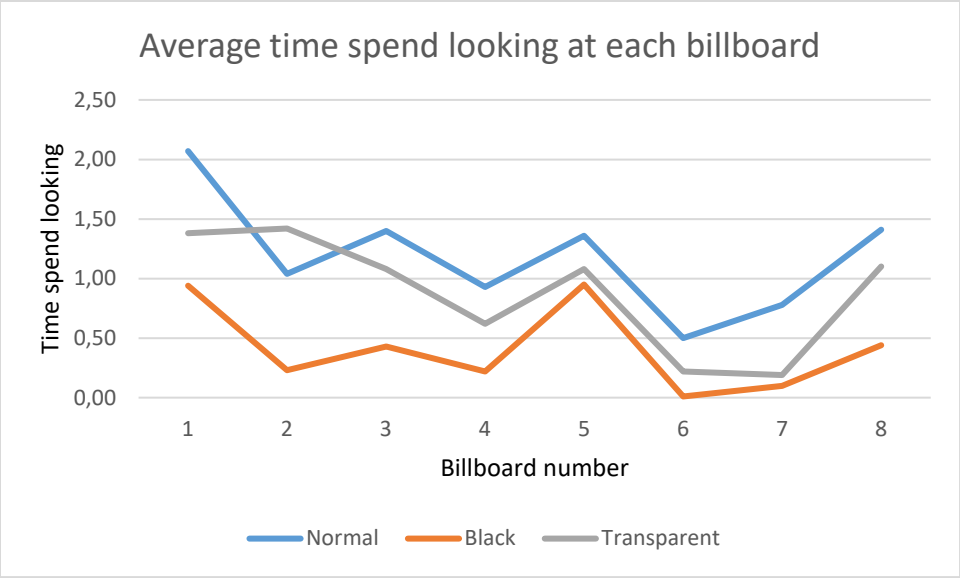

Figure 20 Average time spend looking at each billboard

The graph Figure 20 shows average time spend looking at each of the billboards in each of the rendering methods. We can see that the blackening method has the best results with all billboards compared to both normal and transparent rendering. Transparency is better than normal method except in case of second billboard.

Expect from collecting time spend looking at each billboard, number of looks at each billboard by each of the participants was also collected (Figure 21).

In case of normal rendering, participants looked in total 184 times at a billboard throughout the whole testing. That means 2.3 looks on average per billboard.

Transparent rendering's total number looks throughout the testing is 152 and participants looked on average 1.9 times at each of the billboards. That is decrease of 17.39% compared to normal rendering. 17.39% is a smaller decrease compared to 25.29% decrease in case of time spend looking on average on a billboard with transparent rendering.

Participant	1. bb	2. bb	3. bb	4. bb	5. bb	6. bb	7. bb	8. bb	Billboard rendering
1	9	3	1	1	2	2	2	6	normal
1	2	2	1	1	1	0	1	2	black
1	2	1	3	1	2	1	1	2	transparent
2	3	0	1	1	1	0	0	0	normal
2	3	2	2	0	4	0	0	3	black
2	2	4	3	0	3	0	0	2	transparent
3	9	6	7	2	6	1	1	2	normal
3	0	0	0	1	5	0	1	1	black
3	5	4	4	3	5	1	0	1	transparent
4	4	2	2	0	1	0	0	0	normal
4	1	0	2	2	2	0	0	0	black
4	4	1	2	1	1	0	0	0	transparent
5	7	5	3	3	3	3	3	3	normal
5	3	0	2	0	1	0	0	1	black
5	3	3	2	2	3	1	1	3	transparent
6	3	3	2	3	1	0	0	1	normal
6	1	0	1	1	1	0	0	0	black
6	2	1	1	0	1	0	0	2	transparent
7	2	2	2	1	2	0	0	3	normal
7	5	3	2	2	2	0	1	2	black
7	1	1	2	2	2	1	1	3	transparent
8	5	3	2	2	2	1	1	4	normal
8	2	2	3	1	0	1	0	3	black
8	6	5	4	2	2	1	1	3	transparent
9	0	1	2	1	4	1	2	4	normal
9	0	0	0	1	4	0	0	0	black
9	1	2	3	2	3	0	2	1	transparent
10	3	2	2	3	5	1	1	2	normal
10	4	2	2	1	2	0	1	1	black
10	3	4	1	2	1	2	2	2	transparent

Figure 21 Number of looks per billboard and participant

Blackening method`s total number looks throughout the testing is 94 and participants looked on average 1.21 times at each of the billboards. This means decrease of 47.28% compared to normal rendering and 36.2% decrease compared to transparent rendering.

This also means that blackening method has again the best results, although the improvement is slightly lower compared to the time spent looking at each billboard.

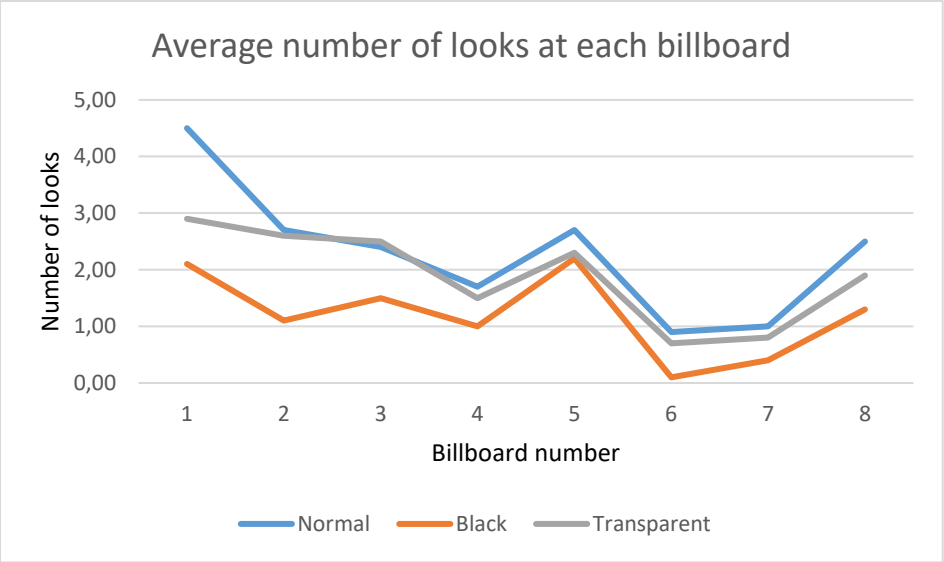

Figure 22 Average number of looks at each billboard

Graph Figure 22 shows average number of looks at each of the billboards. We can see that the black method is still better than both normal and transparent rendering at each of the billboards. Except for the first billboard, normal and transparent rendering are comparable in terms of number of looks.

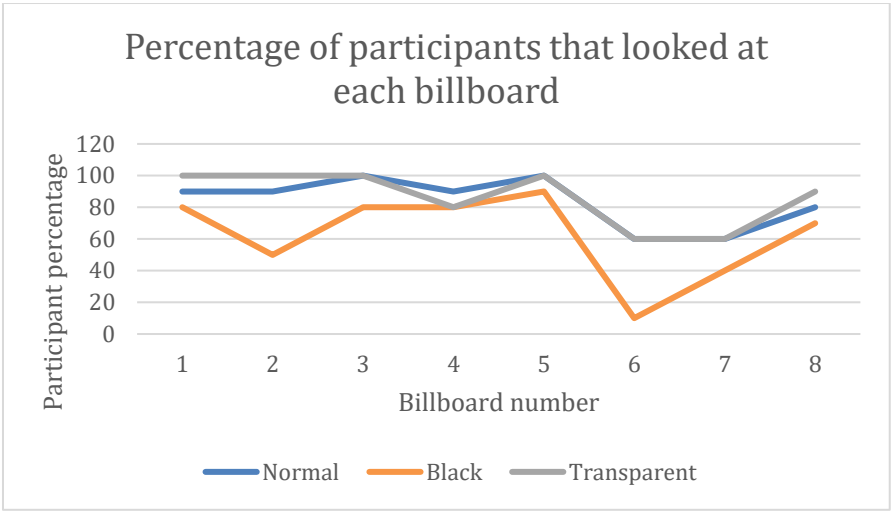

Figure 23 Percentage of participants that looked at each billboard

One the graph Figure 23 we can see that in terms percentage of billboards participant looked at, black rendering had again the best results. Black rendering had the lowest percentage of all three methods for each of the billboards. In this metric, transparent method is worse than rendering billboards normally.

Participants looked in average on 83.75% of the billboards in case of normal rendering. With transparent method, participants looked at 86.25% percent of all billboards, which is 2.98% increase compared to normal rendering. Finally, in case of black rendering, participants looked at 62.5% of billboards, which is 25.37% decrease compared to normal rendering.

Normal	Black	Transparent
8	6	8
10	8	6
6	6	7
8	5	6
7	4	6
10	10	10
8	5	7
5	4	5
9	7	8
10	6	8
8	7	7

Figure 24 Number of guessed billboards by each participant in each driving round

As a part of the questioner at the end of the testing, I asked each participant to guess number of billboards for each of the driving rounds (Figure 24). The right answer was 8 for all three rounds. Most participants guessed highest number of billboards with the normal billboard rendering – an average of 8.1, which is even higher than real number of billboards. 40% of participants guessed higher number than 8, which was the right amount. Second was the transparent method, where the average amount of billboards guessed was 7. The smallest number of billboards was guessed with the blackening method – an average of 6.2.

I also asked each of the participants which method would they personally prefer, and 8 out of 10 saw the blackening method as the better one. Most argued that the blackening method removes their motivation to look at the billboards as they know there is no content in case of the transparent method.

Based on the data collected both for number of looks and for time spent looking at each billboard, the blackening method showed the best results, 56.15% reduction in distraction amount on average trough out all the main metrics used. The transparent method was worse than the blackening method but still better than normal rendering at 21.34% in distraction decrease. Also number of guessed billboards and other questions related to billboards given to each of the participants showed best results for the blackening method.

For these reasons, I will be showing an implementation of this method in the next chapters of this work.

6 Implementation of distraction reduction method

6.1 Sketch of real environment implementation possibilities

The hardware implementation is not the target of this thesis. Target is to find, whether it is possible to detect and block roadside advertisement and if it is efficient and helpful, but not to implementing the feature in a real car. Sample implementation and demonstration will be shown in next chapters.

The vision is, that this functionality will be implemented into a mixed reality headset (for example Hololens-like device) (Figure 25) or window of a car will be used to block the vision of the driver while cameras and other equipment will track the drivers position and detect the roadside advertisement (Figure 26).

Figure 25 Example of possible future implementation in car environment [16] [17]

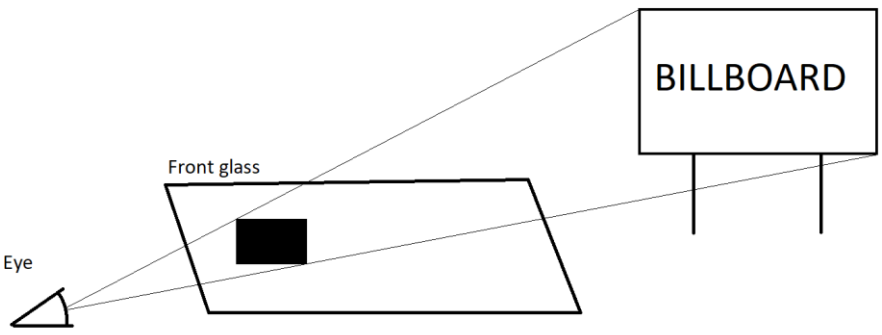

Figure 26 Illustration of the method implemented with front window of a car

This functionality is called mixed reality or augmented reality. It is defined as "*either as a standalone concept or used to refer to the entire spectrum of situations between actual reality*

(i.e. real world) and virtual reality, attempts to combine the best of both virtual reality and augmented reality. When both real and virtual worlds are merged together, new environments and visualizations become possible where physical and digital objects can coexist and interact in real time." [13]. Objects in mixed reality can also be called holograms.

In case of future implementation, the quality of holograms would need to be very high. These are the hologram qualities [14]:

- **Accuracy.** Once the hologram is world-locked and placed in the real world, it should stay where it was placed, relative to the surrounding environment, independent of user motion or small and sparse environment changes. If a hologram later appears in an unexpected location, it is an accuracy problem.
- **Jitter.** Users observe this as high frequency shaking of a hologram. This can happen when tracking of the environment degrades.
- **Judder.** Low rendering frequencies result in uneven motion and double images of holograms. This is especially noticeable in holograms with motion. Constant 60 FPS needs to be maintained.
- **Drift.** Users see this as hologram appears to move away from where it was originally placed.
- **Jumpiness.** When a hologram "pops" or "jumps" away from its location occasionally.
- **Swim.** When a hologram appears to sway corresponding to the motion of the user's head.

All of these would need to be addressed. Refresh rate would need to be consistently above 60 FPS, as this is a recommended minimum. If the mixed reality would not fulfill all the quality criteria, it would probably be even more distracting than the roadside advertisement itself and could even be dangerous. For example, hologram that should originally cover a billboard could block a view of some important road sign or traffic.

If the hologram placement and rendering would be perfect, there still could be distracting and dangerous problems. For example, detecting a traffic sign as billboard or not detecting an important object in front of the billboard and blocking its view.

In case of border cases where the detection or rendering is not correct with high certainty, blocking should be turned off. In other words, false positives cannot appear in a real world use.

There might be also legal issues as most countries specify by law what exactly can and cannot be in a driver's fields of view and glasses and other devices could interfere with these laws.

6.2 Implementation of the method

Again, to stress the goal of this work, the implementation shown here will be a proof of concept, not a final implementation usable in real environment.

To implementing this method, there were multiple ways how to do it. What we need is to create pixel wise detection system that is able to detect one class – billboard, and separate it from the background.

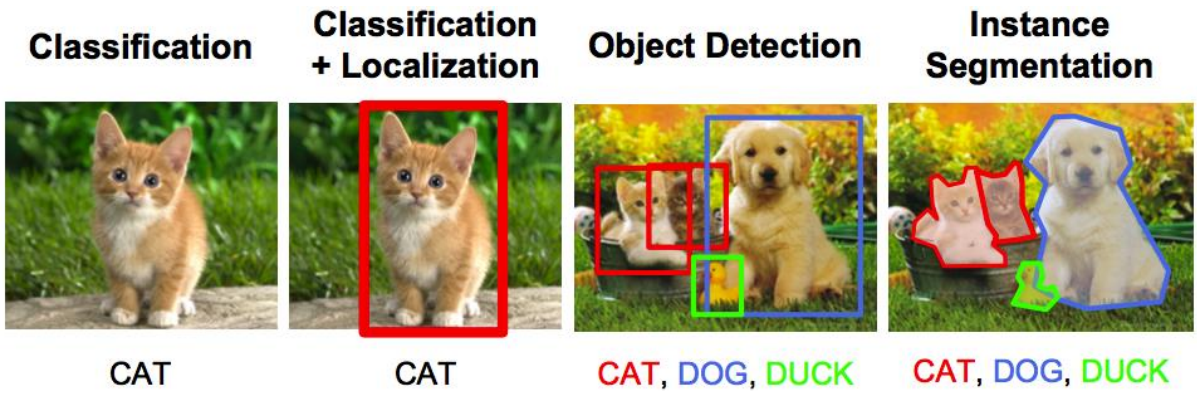

Figure 27 Computer vision algorithms comparison [20]

In computer vision terms, this is called instance segmentation. There are multiple levels of computer vision algorithms. The basic algorithms do only image classification. That means they are able to detect what is the subject of the image and at what probability, but they do not provide any additional information like object position, number of objects of certain class and they can classify only one class, for example cat on the image above (Figure 27).

Classification algorithms with localization add information about object position and size in the image. This is still not enough information for my particular use case.

Next group of algorithms is object detection. These add multiple class detection and at the same time multiple instances of one class. They still do not provide the information of shape, only position and size of each of the class instances. Algorithms from this group could theoretically be used in this work, but the results would be very inaccurate. We would be able to detect only rectangles with position and size. If there would be something in front of the billboard, it would get blocked together with the billboard and there would be many other problems with the inaccuracy (Figure 28).

Figure 28 Instance segmentation vs. object detection on billoard example [21]

The last group of algorithms is called instance segmentation algorithms. These are able to detect multiple classes with multiple instances of one class in one image. They also detect position of the object and exact shape of the object. This is the type of algorithm used in this work.

More basic computer vision algorithms providing only object detection are not sufficient here as we need to know exact shape and pixel-wise information about billboards to be able to block them efficiently and with as little distraction as possible.

6.3 Existing projects for instance segmentation

There are multiple projects with functionality that supports instance segmentation. In this case, we need both to train the model and then use the trained model to apply the trained weights to segment the images. A lot of projects do only one of these things, they either train the model or use already trained model to do the segmentation itself. I decided to use project that does both, as it eliminates possible problems with compatibility. [15]

Most, if not all of these projects are based on one of a three deep learning libraries – Caffe, Tensorflow or Torch/PyTorch.

PyTorch is a python implementation of Torch library. It was developed by Facebook and made open-source. Main advantages are modularity, easy way to create new layer types that run on GPU and lots of compatible pre-trained models that make it easier and faster to train models for a new specific classes.

Tensorflow is a library developed by Google. Compared to other libraries mentioned here, it has much wider functionality like reinforcement learning, which means it is slower in some aspects compared to other libraries that have more limited use. It has python API and the engine itself is written in C/C++. There is lower number of pre-trained models, which can be an issue for some projects, but it was not important in this case. As I plan to train the model by myself, it is not important what was the pre-trained model originally trained on. Together

with this library, there is also possibility to use Keras, which provides an intuitive and simpler API inspired by Torch library. Also very useful utility made for Tensorflow is Tensorboard, which visualizes trained process in real time, allowing to react to for example incorrectly set hyper-parameters much quicker than waiting for some part of the training process to finish.

Caffe is C/C++ port of Matlab's convolutional networks library. It has python API as the rest of the mentioned libraries. In contrast with other mentioned libraries, the functionality is made strictly for image processing. Other advantages are very easy model training and simple python API. Disadvantages are no support in python API to create new layers and smaller support from community compared to the two other mentioned libraries.

All of the three mentioned libraries are suitable for use in this concrete use case. After searching through existing projects and deep learning libraries, I decided to base my work on a Mask R-CNN project which uses Tensorflow library [16].

6.4 Mask R-CNN

The reason I decided to use this project as a starting point was that if offers functionality both for training the network and to use the trained model to do the instance segmentation. The other reason I decided to use this project is that there already are many project based on this one which means there is better community support and smaller chance of incompatibility and errors. I tried a few smaller projects with lower community support and had problems either with poor segmentation results or some sort of incompatibility – drivers, pre-trained models etc.

Mask R-CNN, or Region-based Convolutional network, is based on the Faster R-CNN project. Faster R-CNN detector works in two stages. Firs stage – Region Proposal Network, proposes candidate objects bounding boxes. Second stage extracts features from each proposed bounding-box and preforms classification and bounding-box regression (Figure 29).

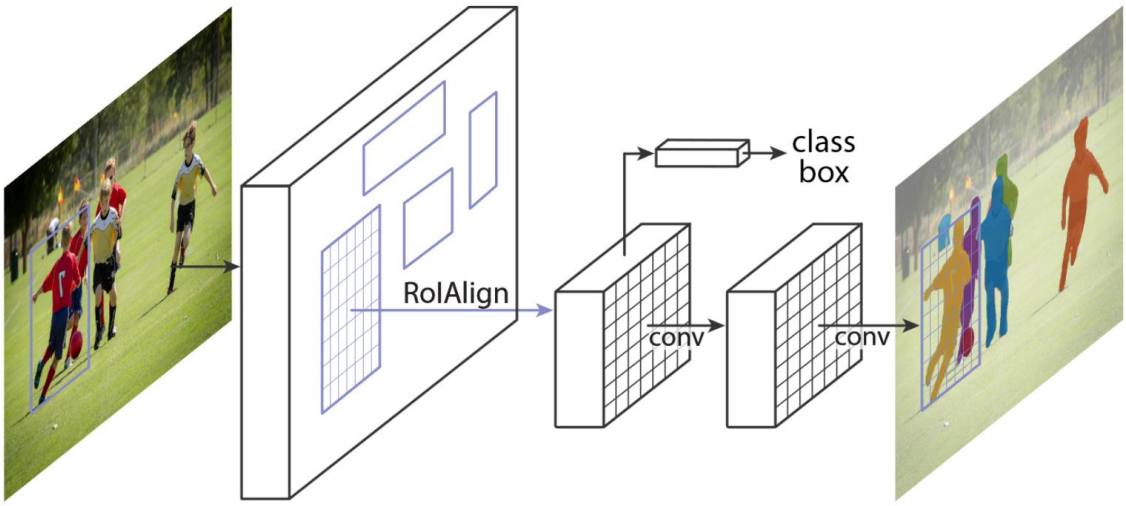

Figure 29 High level Mask R-CNN process [23]

6.4.1 High level view of Mask R-CNN process

First component of the process is backbone network – that is a convolutional layer that servers as feature extractor. Lower levels of network extract lower level features like edges while higher levels extract higher levels of features like for example billboard in this case.

By default, it converts dataset input images to 1024x1024 pixels and then to a feature map of 256x256 pixel. This map is then set as input to another layer and converted to a lower resolution again. There are 5 feature map layers in total (Figure 30). [16]

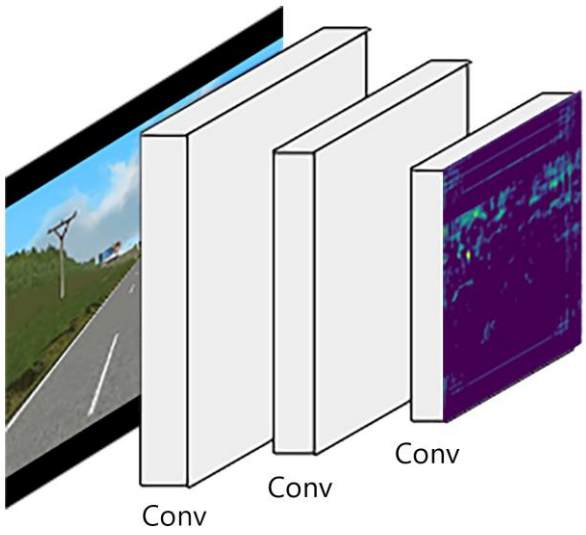

Figure 30 Convolutions and different feature levels visualization

Although this approach can be improved. In the original feature map, lower level layers have access only to lower level features. Feature pyramid network (Figure 31) improves the

standard feature extraction pyramid by passing higher level features to lower level layers, which improves object detection at different object scales. [17]

Figure 31 Feature pyramid networks for object detection [23]

Next step is region proposal network. It scans the already generated backbone feature map using sliding window approach and using a neural network, detects areas that contain some objects. These areas are called anchors. They have predefined shapes (Figure 33) that should be specified so that they correspond with the shape of objects we want to detect. They overlap each other and have different sizes and positions. If there are several closely overlapping anchors, we keep only anchor with highest probability of containing an object. This is called Non-max Suppression. As the sliding window and anchor generation is handled in a parallel manner by GPU, it is very quick to compute. For the image below, the total amount of anchors was 261 thousand. First high resolution feature layer had about 197 thousand anchors (Figure 32) while the last feature layer had only 768 anchors as there is no need for more anchors in a very low resolution space.

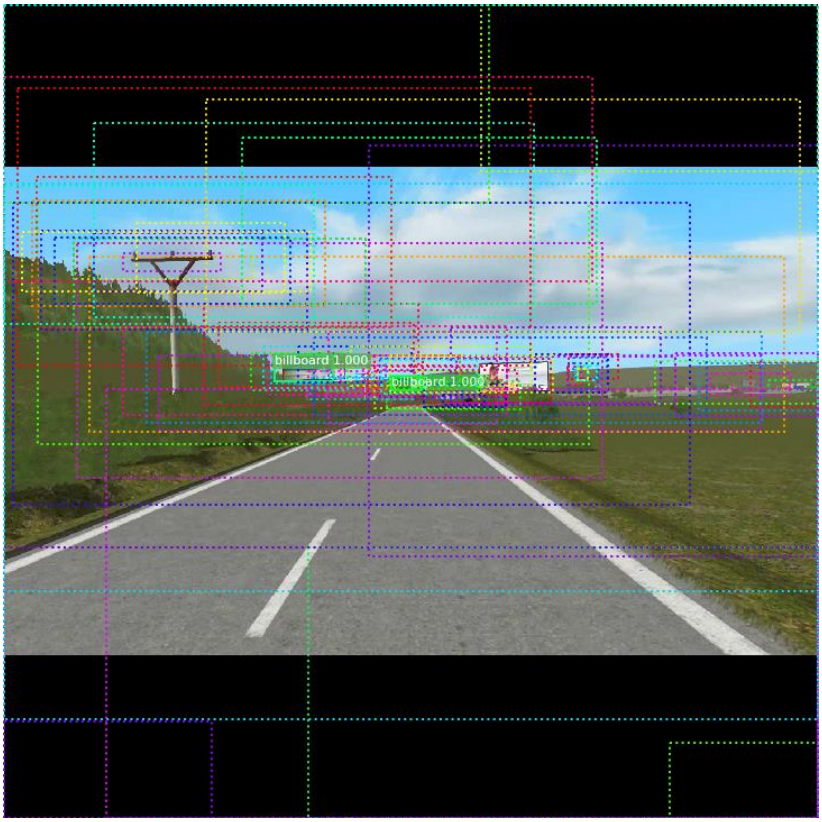

Figure 32 200 randomly selected anchors, about 0.01% of all generated anchors

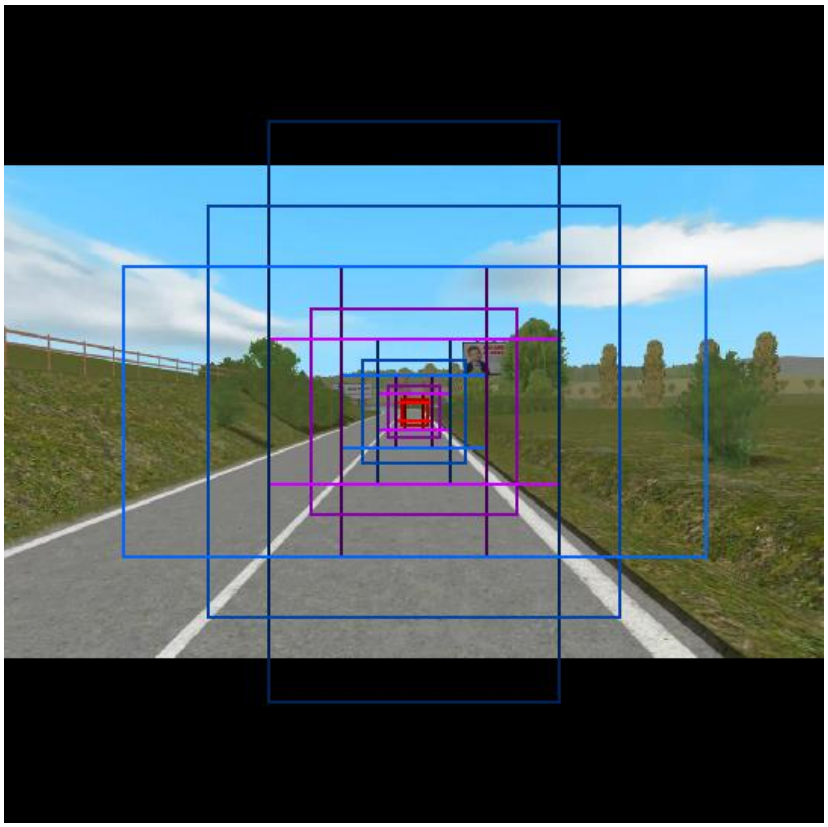

Figure 33 Different anchor side ratios and sizes used

For each of the anchors, region proposal network generates a score that tells at what probability the anchor contains some object and what class it contains. If the probability is less than 0.5, the anchor contains background class or in other words – it probably does not contain any objects (Figure 35). If it has probability more than 0.5 than the anchor most probably contains some objects and we can also say it contains foreground class (Figure 34). If it has probability close to 0.5, then the anchor is called neutral and it is not useful for training and is not used later for other steps of the process (Figure 36).

Box refinement is also calculated for each of the positive anchors. It makes sure that each of the object's box is centered and sized as precisely as possible. More anchor boxes can be merged into one box during the refinement. We can see the refinement on the image bellow, where anchor boxes before refinement are dotted and after refinement are solid.

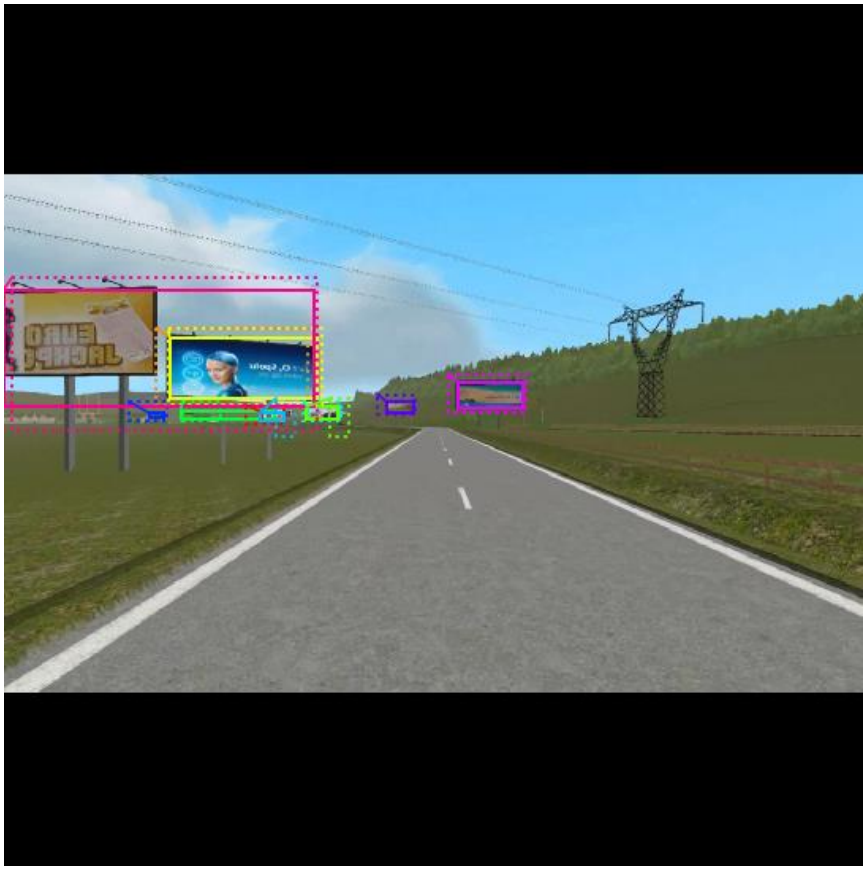

Figure 34 Positive anchors

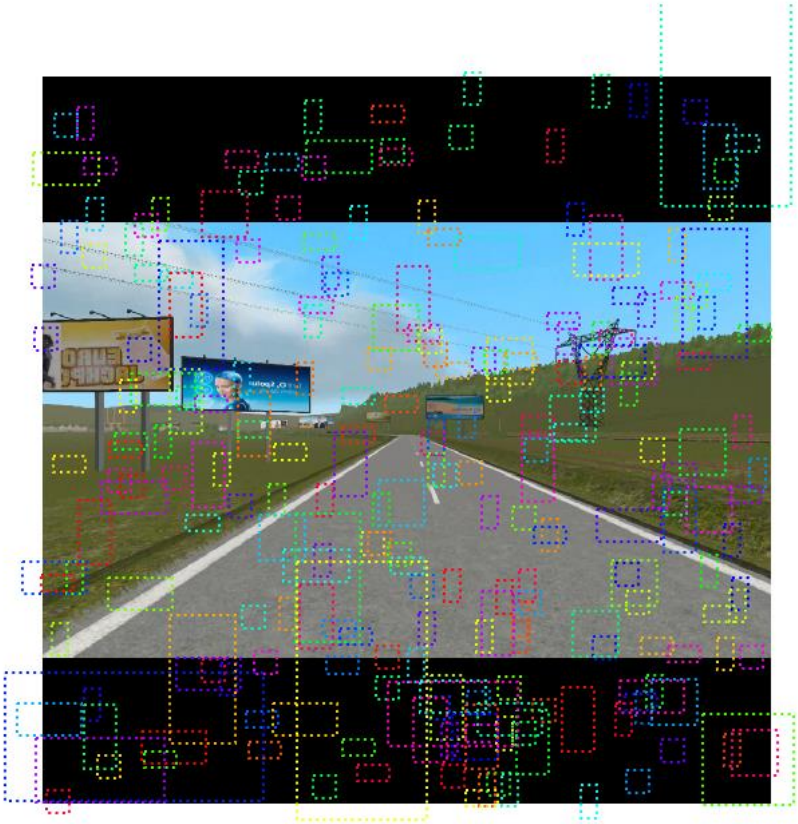

Figure 35 Negative anchors

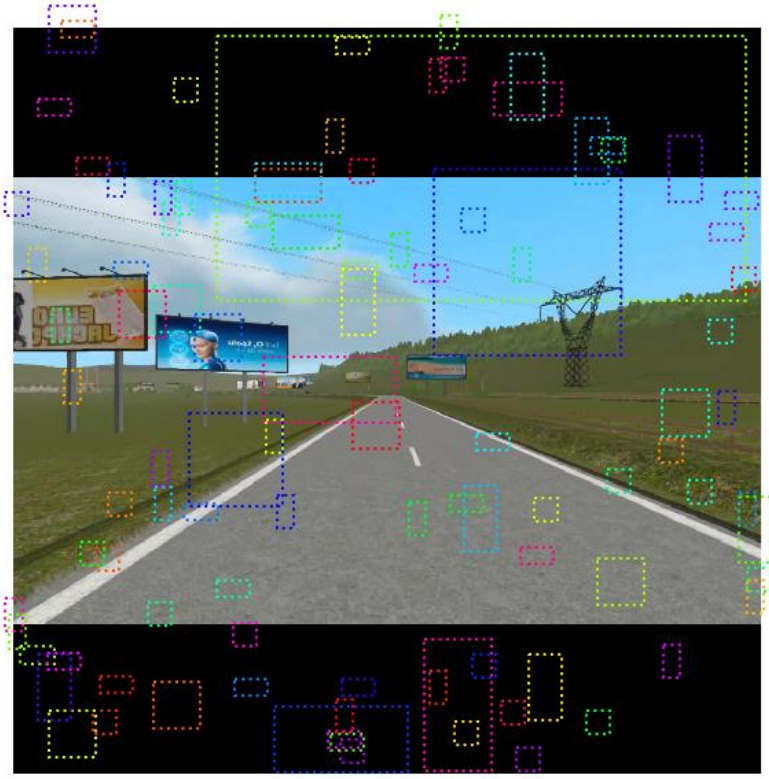

Figure 36 Neutral anchors

After these steps, we have our regions of interest – the positive anchors are parts of the image that we search for the classes we want to detect – billboard in this case.

For each region of interest, we generate class and we do one more bounding box refinement. The neural network in this case is deeper and generates the final classes we want to detect. [17] If this neural network detects only a background class for some of the regions of interest, we discard that region.

To be able to detect class of each region of interest, we need to convert each of the regions to the same size. In this case, resizing and cropping using bilinear filtering is used to convert each region of interest to 7x7 pixels (Figure 37).

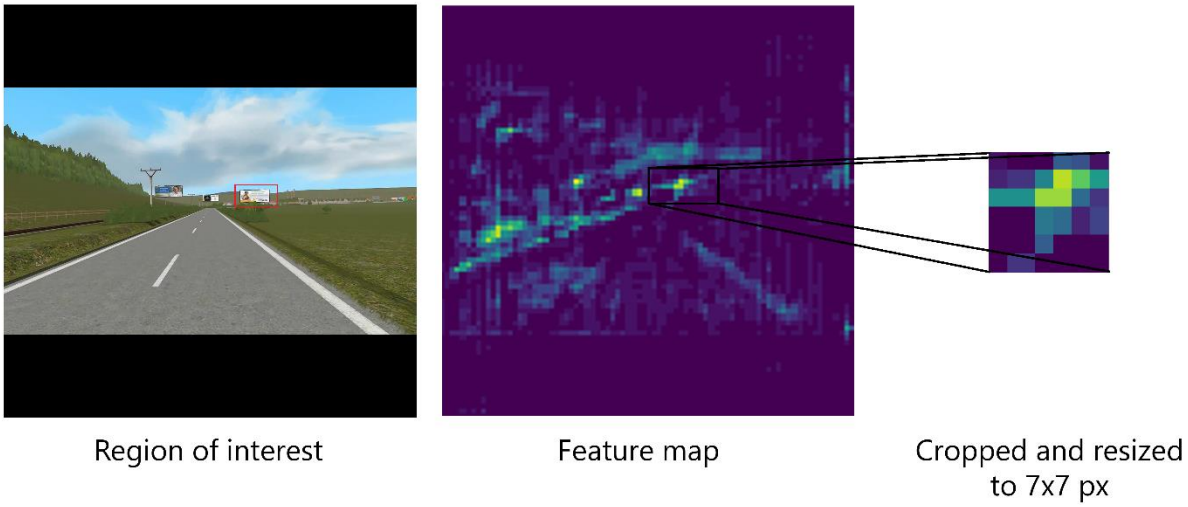

Figure 37 Extracting class of region of interest

All steps until now are originally from Faster R-CNN project. [18] Mask R-CNN projects adds the ability to do pixel-wise detection. It uses convolutional network to do so.

First it down-samples the original masks to low resolution of 56x56 pixels (configurable) (Figure 39). The reason is to make the training faster and less memory heavy. As we need a mask file for each object instance in a picture, 100 objects in a scene would mean around 100 MB of memory and much slower training speed (Figure 38). There is a loss of precision for objects larger then 56x56 pixels, but it is negligible. Also the mask down-sampling can be turned off in configuration of the project, but it significantly increases training time and memory usage and has very little effect for mask accuracy (Figure 40).

The training mask is a ground-trough mask and we train the network minimizing a loss function comparing the ground-trough mask and our detected mask.

Figure 38 Original input masks

Figure 39 Minimized input masks

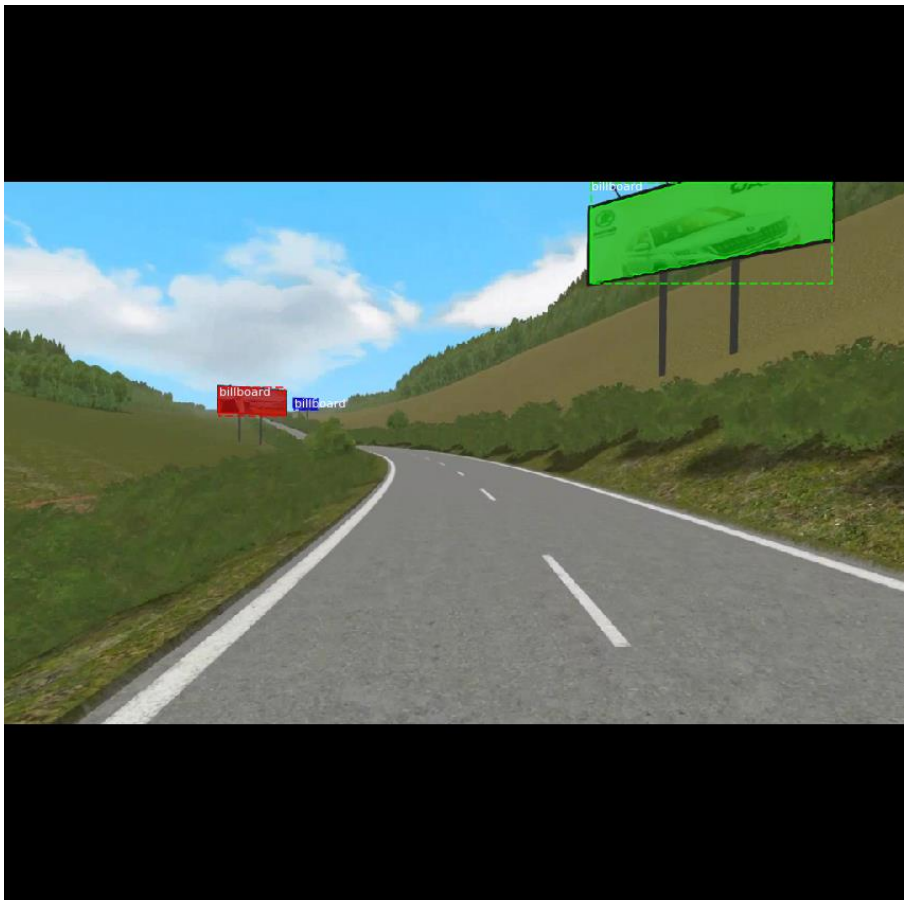

Figure 40 56x56 pixel masks fitted into 1024x1024 pixel image

6.4.2 Training data

Training data and its quality are essential for a training process. The more training data we have the better the results generally are. The quantity needed to train a model to a certain detection quality depends on what we want to detect. The more classes we have, the harder the classes are to tell apart and the more complex objects, the higher amount of training data we need. But there is not any way to tell exact number needed for a good detection results.

In my case, I have around 482 instances in my dataset. The training data I am using was extracted from the simulator track used for testing in the previous chapters. There are additional billboards added (Figure 41) making the total of around 65 billboards. This was done to maximize the amount of usable training data gathered from the track as there is a visible billboard almost on every part of the track.

Figure 41 Billboards added to testing track

There was a curve added that the camera was following, moving at a speed of traffic and recording at 60 frames per second. The result was around 16400 captured images from the 4 minutes 32 seconds long video. As the simulator engine version I was using did not have required functionality to record this video, a newer version of the engine had to be used. As there were license issues and the newer version of simulator not being fully released at the time of making this project, I had to prepare the track, send the files to a colleague with access to a new version who recorded the video and provided it to me.

For training process, we also need the segmentation data. These were created the same way on a same track but with different textures. Every object was set to black color and each billboard surface got its own color (Figure 42). Again, video was recorded and provided to me.

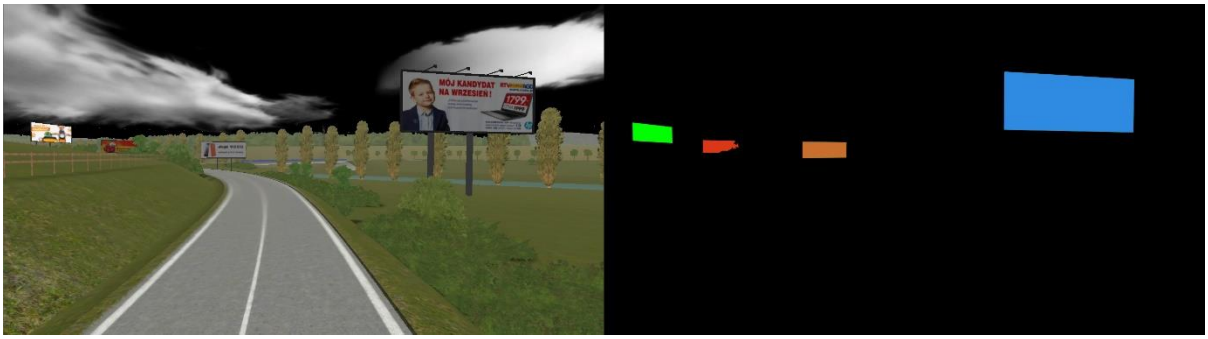

Figure 42 Image and its corresponding mask

My original vision was to synchronize the two videos to start at the exact same time and frame, sample pictures from each video at the same sample rate and create data this way. Sadly, the two videos were slightly out of sync trough out the whole videos, not only at the start of the video. Some of the parts of the track were slightly faster on one track, some parts were faster on the other track.

That meant that I could not find any way to completely sync these videos together and as I could not influence the creation of the videos, because I had no access to required version of the engine, I had to sync both videos frame by frame by hand. To extract the frames and sync videos, I used open source utility FFmpeg.

First I synchronized the start of each video. I found and exact frame on each of them and cut both videos to start with this frame.

```
ffmpeg -i input.mp4 -ss 00:00:00.70 -c copy output.mp4
```

Then, I extracted the frames from the video. I extracted every eighth frame from the video with normal textures and every frame from the segmentation video.

```
ffmpeg -i input.mp4 -filter:v fps=60 image_%0d.jpg
```

After getting the images, I selectively reduced the number of pictures with normal textures so that there are more samples from places with more billboards and less samples from places where there are less billboards. This gave me 482 images.

Now, when I had the images, I had to pair the images from track with normal textures with images from segmentation track. First I selected one of a 482 images from normal track and then found a corresponding image from the 16400 segmentation images. Despite both videos being recorded at 60 frames per second, some of the images had no suitable segmentation image pair (Figure 44). This was problem especially when there were objects

moving fast relative to camera and placed between the billboard and the camera (Figure 43). In these cases, I had to modify the mask images or create a completely new mask images. Due to all these reason, this process was very time consuming. That was the reason I chose to train using only 482 instances.

Figure 43 None of two adjacent images from segmentation video fit the original image

Figure 44 Again, no fitting mask for original image

After fitting all the segmentation images and images from the normal track together, I found out other problem. Due to the original segmentation video not being encoded in a lossless format, the colors of the billboards were shifted from the original color of the texture (Figure 45). Also the color of single billboard in one image was not the same on the whole billboard area. Especially near the edges, the colors where noticeably shifted.

Figure 45 Detail of a compressed edge of billboard mask

First I tried loading the segmentation data this way – read image pixel by pixel. If the color of pixel +- 15 in each of the RGB channels was already seen, add this pixel to that set of pixel, otherwise create new set of pixels. The problem of this approach was that it was quite slow and still was not reliably working, especially around edges, creating multiple billboard instances from a single original one. To make the process faster and more accurate, I decided to create a CSV file for each data instance containing the right color for each billboard instance and then using this color with +- 15 tolerance in each RGB channel to find each billboard instance area. This proved to be much more accurate and faster approach. The negative point was the need to create the CSV files for each of the data instances.

After all data instances were created, the data was split in 80:20 proportion for training and validation data. Training data is used for training itself, validation data is used during training too, but for parameter selection and to prevent overfitting, not for training weights itself. The 80:20 ratio called Pareto principle is usually a recommended value for splitting the data or at least a good starting point [19].

6.4.3 Using Mask R-CNN

To be able to use the Mask R-CNN project to train model using your own dataset and then apply the trained model to a set of images, we need to first create functionality to load the dataset.

For each of the units of the dataset, we need to have the input picture itself and the segmentation mask. Segmentation mask does not need to be a picture necessarily. Some other popular formats are JSON or XML files specifying boundaries of each object with set of points. We can then connect those points, fill the inside of the newly created object and create mask this way.

When using a picture as a mask, we need to be able to tell apart each instance of each of the classes. That means it is a good practice to give each instance its own color (or differentiate them some different way). If all instances would have the same color, it would be impossible to tell apart overlapping objects.

In my own example, I also had CSV file for each dataset instance containing correct color for each billboard instance mask.

To be able to load the dataset properly, we need to implement our own Dataset class extending the utils.Dataset class. We need to implement at least two methods in this class – method to load all our dataset pictures. To add picture to dataset, we call Dataset method add_picture, which adds “link” to the image that is later used to load the image itself and to find the image mask.

Second method we need to implement is load_mask that based on the information saved by the add_image method, loads masks for a specific image. In my case, I load both the segmentation image and CSV file with defined colors. Then I use OpenCV library to filter the segmentation image. For each color from CSV file, I apply inRange filter on the image with lower and upper boundary set to +-15 from each channel of RGB image. This return a binary array with ones where the color was in the desired range and zeros otherwise. I add this array to final mask array with dimensions of mask height x mask width x number of instances. We return this array together with 1 dimensional array with class defined for each instance, that means array of size 1 x number of instances in our case were we have only one class – billboard.

If we do not want to use the default configuration for training, we need to extend the Config class. Here we set things like number of images per GPU, number of classes, learning rate and other things.

If we want to do the training itself, we need instances of previously created configuration class and dataset class.

```
model = modellib.MaskRCNN(mode="training", config=config,
                          model_dir=MODEL_DIR)
```

When we have the model created, we need to decide what initial weights we want to start with. We can either create empty weights and start training from the beginning or use a pre-trained model and train it to our needs. The initial weights can be trained on a totally different classes then we want to detect. The advantage is that it makes the training faster and reduces the dataset instance count needed, because even though the model is trained on different classes, billboard class will most probably share many features with them and thus reduce the training time as we do not need to extract those feature anymore.

In my case, I decided to use a model pre-trained on COCO - Common objects in Context dataset. We also need to exclude things from dataset that we do not use.

```
model.load_weights(COCO_MODEL_PATH, by_name=True, exclude =
                   ["mrcnn_class_logits","mrcnn_bbox_fc","mrcnn_bbox","mrcnn_mask"])
```

After loading the initial weights, last two things we need to decide before learning is if we want to train all the neural network layers, or only the top layers extracting only higher level features and the number of epochs. Training only top layers is recommended when we train object similar to objects used to train the initial model, if we have very small dataset or when the object, we want to detect is very simple. None of these things is true in the case of billboards, so we will train all layers. This is indicated by layers='all'.

We also need to set the number of epochs we want to train. Each epoch trains the model using the amount of dataset instances defined in configuration class in parameter STEPS_PER_EPOCH. This is by default 1000, but it should be ideally set to number of instances in dataset. At the end of each epoch, checkpoint with trained weights is saved. The number of epochs is unknown in advance. Generally, we want to stop the learning process when the validation loss stops improving and starts “jumping around”. The number of epochs should not be set very high to avoid overfitting. The number again depends on many factors like set learning rate, number of training instances, complexity of class etc. In my case, I trained 80 epochs. I was using Nvidia Titan XP GPU with 12 GB of memory, which was capable of training with minibatch of 2 images. One epoch took about 20 minutes, so the whole training process took about 27 hours. When trying to implement training process using CPU only, one epoch took about 100-200 times longer than compared to using GPU.

```
model.train(dataset_train, dataset_val, learning_rate=config.LEARNING_RATE,
            epochs=epoch_num, layers='all')
```

When we finish the training and want to start detecting billboards on images, we need to create another class that extends config class as detection and training configurations are different. We need to set IMAGES_PER_GPU to 1 as we will detect billboards on one image at the time. Also, we can set DETECTION_MIN_CONFIDENCE parameter, which sets the minimum detection confidence. This parameter is 0.9 by default. I am using 0.95 – 95% confidence detection. After that we create model that we will use for detection and initialize it with weights from previously trained model.

```
model = modellib.MaskRCNN(mode="inference", config=detection_config,
                          model_dir=MODEL_DIR)
model.load_weights(model_path, by_name=True)
```

Now we can start detecting the billboards. We read the image using scikit-image library in this case and run the detection process using model.detect method.

```
image = skimage.io.imread(img_path)
masks = model.detect([image], verbose=1)[0]
```

This method, besides other things, returns array of masks with dimensions of image height x image width x detected instances count. Now all we need to do is collapse all the instance masks into a single mask, apply the mask to all RGB channels and we have the resulting image. What I also added was a Gaussian blur with sigma (standard deviation for Gaussian kernel) dependent on image size to make the edges of the masks more seamless.

6.5 Implementation results

After training, I used the detection to classify the whole simulator video from testing track. I again created frames from the video file using FFmpeg utility. After running the classification, a used the FFmpeg to create back the video from the segmented frames.

```
ffmpeg -framerate 30 -pattern_type glob -i '*.jpg' -c:v libx264 -profile:v high -crf 20 -pix_fmt
yuv420p result.mp4
```

Results were generally very good (Figure 46, Figure 47 and Figure 48). All the billboards were detected at a very high confidence levels, vast majority of billboards at confidence higher than 99%. Even partially covered or overlapping billboard were detected well. Trough out the whole track, there was only 1 billboard that did not get detected properly.

Figure 46 First segmentation result from simulator testing track

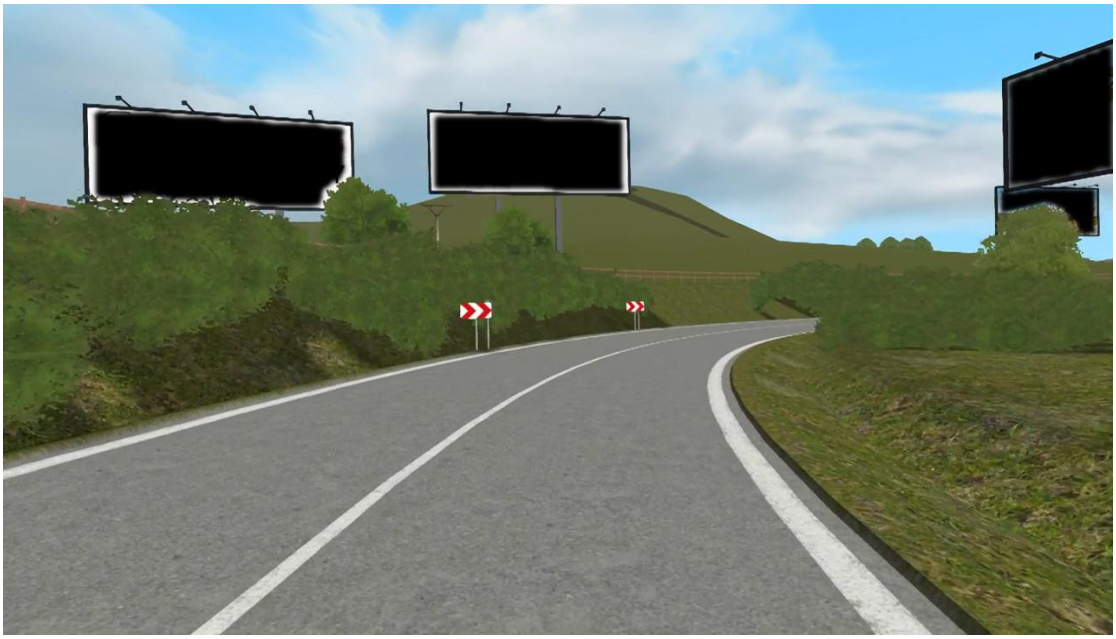

Figure 47 Second segmentation result from simulator testing track

Figure 48 Third segmentation result from simulator testing track

There were also some problems with detection. On the images above, we can see that the billboards are not detected all the way to the edge of the billboard area. To make it more precise, more training data would probably be needed. I think that also the compression of the video mentioned in training data chapter might have influenced the training masks and some of them have not been covering the whole area of the billboard. That is the reason I added the Gaussian blur, to make this effect less visible.

Next problem was occasional flickering of the mask (Figure 49). When the billboard was for example partially covered, and on one frame we detected the billboard, on the next

frame we did not and on the next frame we again detected it, it created flickering that might have been distracting.

Figure 49 Zoomed in detail of flickering on 3 consequent frames of partially covered billboard

Other problem was detection of other objects as billboards. For example, a car, sign or a wall of a building (Figure 50). That was resolved by increasing the detection confidence from 0.95 to 0.99. It introduced slight flickering at some of the images as shown above, but it significantly reduced the number of wrongly detected objects.

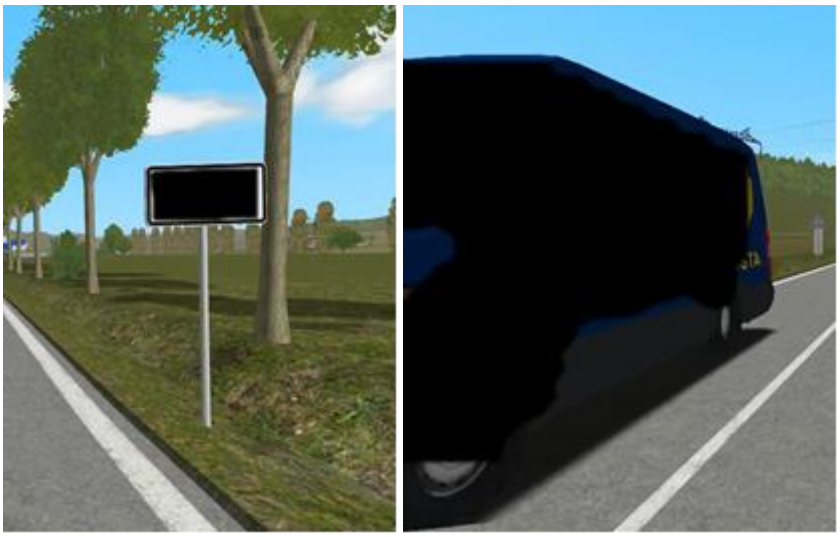

Figure 50 Zoomed in detail on wrongly detected objects

Even if counting all these inaccuracies, detection on the testing track was very good trough out the whole track. Even billboards that were partially covered, billboards behind moving objects or overlapping billboards were detected correctly.

Besides the testing track, I also used the segmentation on a recording from a real traffic (Figure 51, Figure 52, Figure 53 and Figure 54). The video was taken in Prague, street K

Barrandovu. It was processed the same way and using the same model as the recording from the testing track.

The problems with detection are basically the same as with the previous recording, but more pronounced. As the model was trained using simulator dataset, not all the features are common with real world, so decrease in detection quality was expected.

Figure 51 First example of image classification from real driving environment

Figure 52 Second example of image classification from real driving environment

Figure 53 Third example of image classification from real driving environment

Figure 54 Fourth example of image classification from real driving environment

As already mentioned, the detection problems that were rare in the case of the simulator test track, were much more pronounced here. Flickering was an issue but it was much more noticeable. Some billboards were not detected properly at all and some objects that should not be detected were detected and blocked. That is especially problem with traffic signs or cars in this case as it can easily cause dangerous situations (Figure 55).

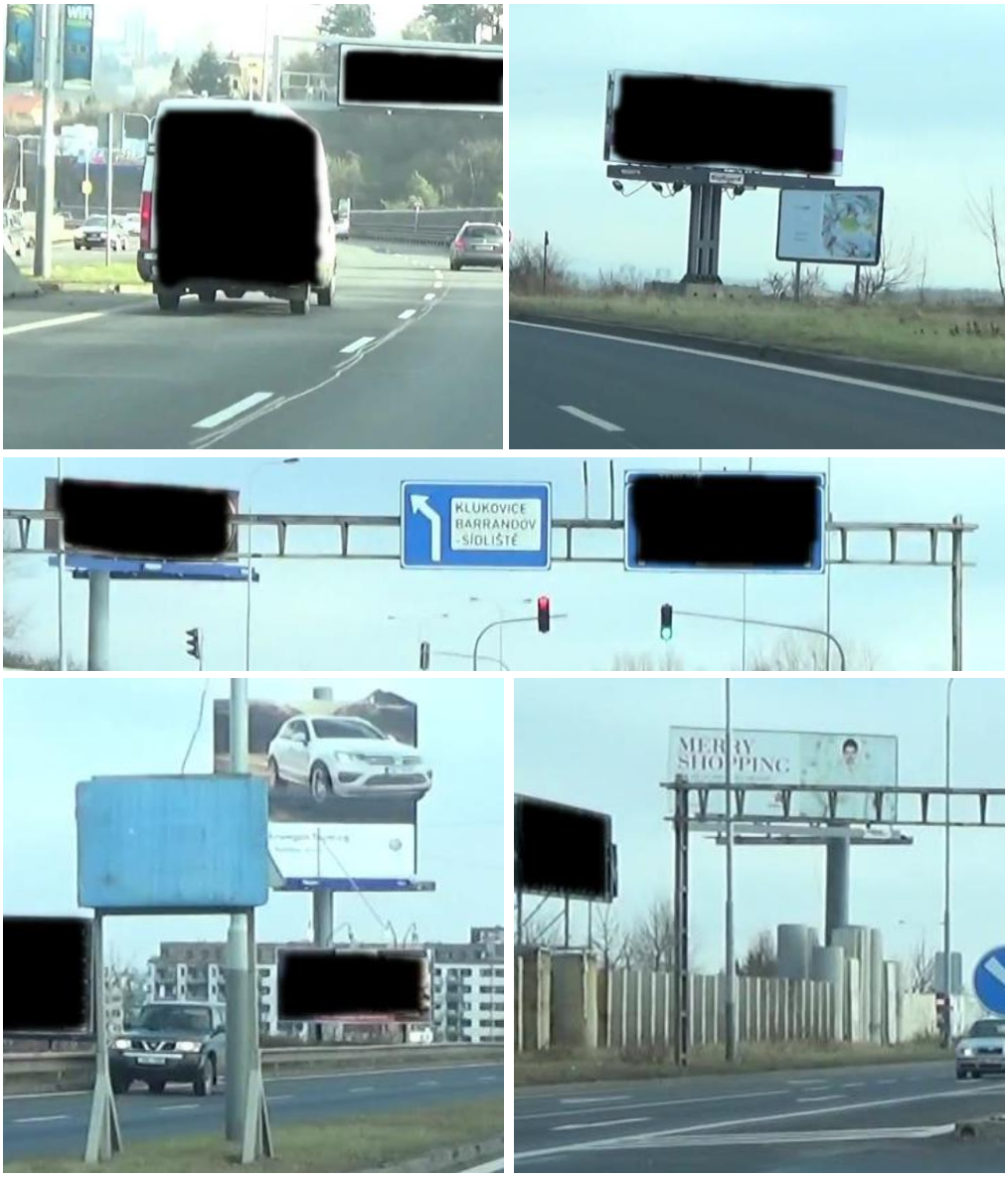

Figure 55 Detection errors from a real environment data

Despite the mentioned detection errors, I believe the results are generally good as demonstration of the blackening method. The model is of course far from being usable in real traffic. For that to be true, training data from a real driving environment would be needed with all the edge cases that are possible. But I believe that the trained model used in this project would be a great starting point for further training, if someone would want to implement this feature in real environment.

6.6 Future work and possible detection improvements

As I already mentioned in the previous chapter, the model is not perfect. I would like to mention a few approaches that I believe would increase the accuracy of the detection.

The data used for training were synthetic and do not fully represent real environment. That means we cannot extract all the features we would need to do the detection more reliably in real environment. For that to improve, we would need training data from a real environment. As the real environment is more complex with more edge cases, training set of would need to be bigger than the one used in this work.

Next possible improvement would be to use between-frame information. In this work, the segmentation is done only on a single frame basis. Frames do not influence each other's segmentation results. There already are papers and projects using this approach, like for example Learning to Segment Moving Objects [20]. This would probably help to at least reduce or remove the flickering mentioned in previous chapter.

Another approach that could help to increase the detection quality would be using depth data (Figure 56). Self-driving cars already use depth data from lidars for object detection. Adding information about space would mean more possible features to extract and thus better detection quality.

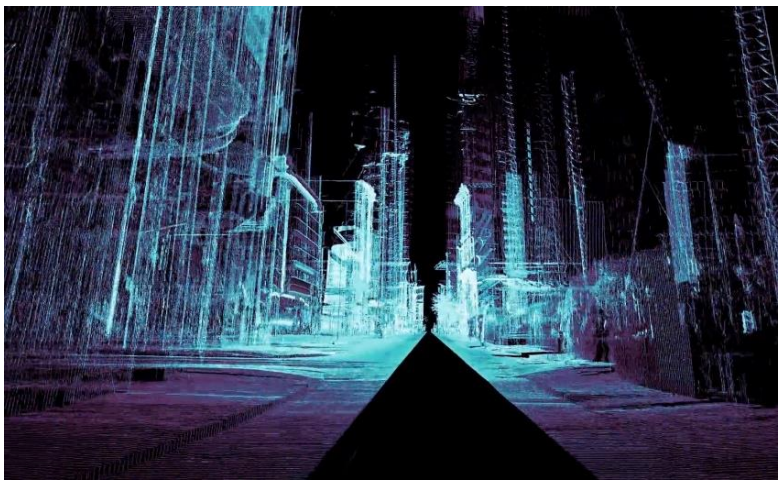

Figure 56 Visualisation of depth data from self-driving car [28]

To further increase detection quality on the simulator training data, we would simply need more dataset instances. But with increasing number of instances, if we would want to use the model on a different data than the original dataset, we would need to be careful not to over-fit the model thus making the detection more accurate on the training and validation data, but worse on different input data.

Also, for this to be usable in real environment, detection would need to be done in real time. That means ideally minimally 60 frames per second. Using the code shown here to load, segment and save images one by on Nvidia Titan XP yields around 3-4 frames per second. The reason is that there is no performance enhancement done with the model. With these optimizations, the target frame rate should be possible even on less powerful hardware.

7 Conclusion

The goal of this work was to define distraction caused by roadside advertisement, billboards more specifically. After defining the distraction and billboards themselves, the next goal was to find and make a simple implementation of method that reduces the distraction caused by these billboards.

First I made a research about roadside advertisement. I named most types of billboards, their placement, legislation and the spoke about the distraction they cause.

The goal of this work was to find a good method of reducing distraction caused by these billboards. The picked blackening method from the two tested methods was by a good margin better according to user testing and it significantly reduced distractions caused by the billboards. The user testing proved significant results, where the picked method was better trough out the whole testing track in all measured variables. Also, post-test participant opinions showed again the blackening method as the better one.

Based on the testing, I showed an implementation of this method as a proof of concept. I explained that we need a pixel-wise detection system and that it is called instance segmentation. I named the frameworks projects supporting instance segmentation use and chose to base my work on Mask R-CNN project as it supported functionality both to train and then do the instance segmentation itself and it was one of the most community supported projects I found.

I showed the stages of Mask R-CNN and high level view of what the projects does. After that, I showed the process of creating the test data based on the track used for testing in previous chapter for participant testing. I showed that I thought that creating the data will be simple automated process, but in the end had to synchronize and create the data instances by hand, which was very long and tedious process and that the manual creation was the reason I created only around 500 training instances.

After creating the training set, I mentioned most important steps how to make the Mask R-CNN project work with your own data. First I talked about how to load the dataset, how to prepare it for training process and how to create the training model. I mentioned that it is better to use an already pre-trained model, even if it does not support the classes we want to detect, because it makes the training process faster as it already contains some trained features we no longer have to train ourselves.

Then I showed the results of detection of the trained model on two sets of data. Simulator testing track and recoding of driving in real environment. The results on the simulator testing track were good with only a few detection errors, most common of witch was mask flickering were billboard is detected on the first frame, not detected on the next frame and again detected on the next frame. But the detection was accurate on this set of input data.

The detection on the data from real environment was worse. The flickering was more prominent, some billboards were not detected and there were also problems with some unwanted objects like cars or traffic sign being detected. I mentioned that the model is not ready for real environment use, but this was not the target of this work and if someone would want implement this method of blocking in real environment, the trained model used here would be a good starting point.

At the end, I mentioned ways I can see to improve the real environment results. Those were using training data from a real environment with as many dataset instances as possible, using between-frame information and adding depth information.

8 Sources

[1] J. Turill, J. Cooper, J. Coleman, N. Mediros-ward and F. Biondi, "Assessing Cognitive Distraction in the Automobile," 2015. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0018720815575149.

[2] V. Neale, S. Klauer, R. Knipling, T. Dingus, G. Holbrook and A. Petersen, "The 100 Car Naturalistic Driving study," 2002. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/100carphase1report.pdf.

[3] B. Wallace, "External-to-vehicle driver distrction," 2003. [Online]. Available: http://www.gov.scot/Publications/2003/08/17916/24500.

[4] P. Roberts, K. Boddington and L. Rodwell, "Impact of roadside advertising on road safety," 2013. [Online]. Available: http://scenic.org/storage/PDFs/austroads%20research%20report%20on%20impact%20of%20roadside%20advertising%20on%20road%20safety.pdf.

[5] "ABC," 2015. [Online]. Available: http://www.abc.es/tecnologia/redes/20150806/abci-matrimonio-201508060940.html.

[6] "Statistické vyhodnocení vlivu bezpečnostního opatření na nehodovost v silničním provozu na vybrané lokalitě," Reditelství služby dopravní policie PP ČR:, Praha, 2012.

[7] "Distractions," UIA, 2000. [Online]. Available: http://encyclopedia.uia.org/en/problem/139057.

[8] "Three Types of Driving Distractions," [Online]. Available: https://www.dmv.org/distracted-driving/three-types-of-distractions.php.

[9] K. Young and M. Regan, "Driver distraction: A review of the literature," 2003. [Online]. Available: http://acrs.org.au/files/papers/15%20Young%20A%20review%20of%20the%20literature.pdf.

[10] J. S. Decker, S. J. Stannard, B. McManus, S. M. O. Wittig, V. P. Sisiopiku and D. Stavrinos, "The Impact of Billboards on Driver Visual Behavior: A Systematic Literature Review," 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4411179/.

[11] G. Diveka, "The effect of external distractions on novice and experienced drivers` anticipation of hazards and vehicle control," 2011. [Online]. Available: http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1710&context=theses.

[12] D. Nádeník, "Driver`s Useful Field of View," 2015. [Online]. Available: https://dspace.cvut.cz/handle/10467/64055.

[13] "Mixed reality," Reality technologies, 2016. [Online]. Available: http://www.realitytechnologies.com/mixed-reality.

[14] A. Turner, M. Zeller, E. Cowley and B. Bray, "Hologram stability," Microsoft, 2018. [Online]. Available: https://docs.microsoft.com/cs-cz/windows/mixed-reality/hologram-stability.

[15] "Comparing Top Deep Learning Frameworks: Deeplearning4j, PyTorch, TensorFlow, Caffe, Keras, MxNet, Gluon & CNTK," DL4J, 2017. [Online]. Available: https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch.

[16] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2018. [Online]. Available: https://arxiv.org/pdf/1703.06870.pdf.

[17] A. Waleed, "Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow," Matterport, 2018. [Online]. Available: https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46.

[18] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015. [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf.

[19] I. Guyon, "A scaling law for the validation-set training-set size ratio," 1996. [Online]. Available: https://pdfs.semanticscholar.org/452e/6c05d46e061290fefff8b46d0ff161998677.pdf.

[20] P. Tokmakov, C. Schmid and K. Alahari, "Learning to Segment Moving Objects," 2017. [Online]. Available: https://arxiv.org/abs/1712.01127.

[21] S. G. Klauer, F. Guo, B. Simons-Morton, M. C. Ouimet, S. E. Lee and T. A. Dingus, "Distracted Driving and Risk of Road Crashes among Novice and Experienced Drivers," 2014. [Online]. Available: http://www.nejm.org/doi/full/10.1056/NEJMsa1204142#t=article.

[22] B. Simons-Morton, S. Klauer and M. Ouimet, "Naturalistic Teenage Driving Study: Findings and Lessons Learned," 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4583651/.

[23] C. Blanchard and A.-L. Banse, "BFMTV," 2018. [Online]. Available: http://www.bfmtv.com/societe/a-paris-les-grands-panneaux-publicitaires-vont-disparaitre-1112732.html.

[24] "Ultravision international," 2017. [Online]. Available: http://www.ultravisioninternational.com/.

[25] P. Draznik, "lednuselskymost," BigMedia, 2016. [Online]. Available: http://lednuselskymost.cz/.

[26] "triplesign," 2007. [Online]. Available: http://www.triplesign.com/TriplesignSystemAB/Technicalinfo/ECOTechnology.aspx.

[27] V. Biryukov, "Kaspersky blog," Kaspersky, [Online]. Available: https://www.kaspersky.com/blog/evolution-of-human-car-interaction/6953/.

[28] "Microsoft Hololens," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/hololens.

[29] B. F. Duffy and D. R. Flynn, "A Year in Computer Vision," The M Tank, 2017. [Online]. Available: http://www.themtank.org/a-year-in-computer-vision.

[30] M. Říský, "Billboardů výrazně ubude," deník.cz, 2017. [Online]. Available: https://www.denik.cz/regiony/billboardu-vyrazne-ubude-20170822.html.

[31] G. Manugh, "The Dream Life of Driverless Cars," The New York Times Magazine, 2015. [Online]. Available: https://www.nytimes.com/2015/11/15/magazine/the-dream-life-of-driverless-cars.html.

9 Attachment list

1. Post-test questionnaire
2. Project source code
3. Training and validation dataset
4. TensorFlow model used in this work