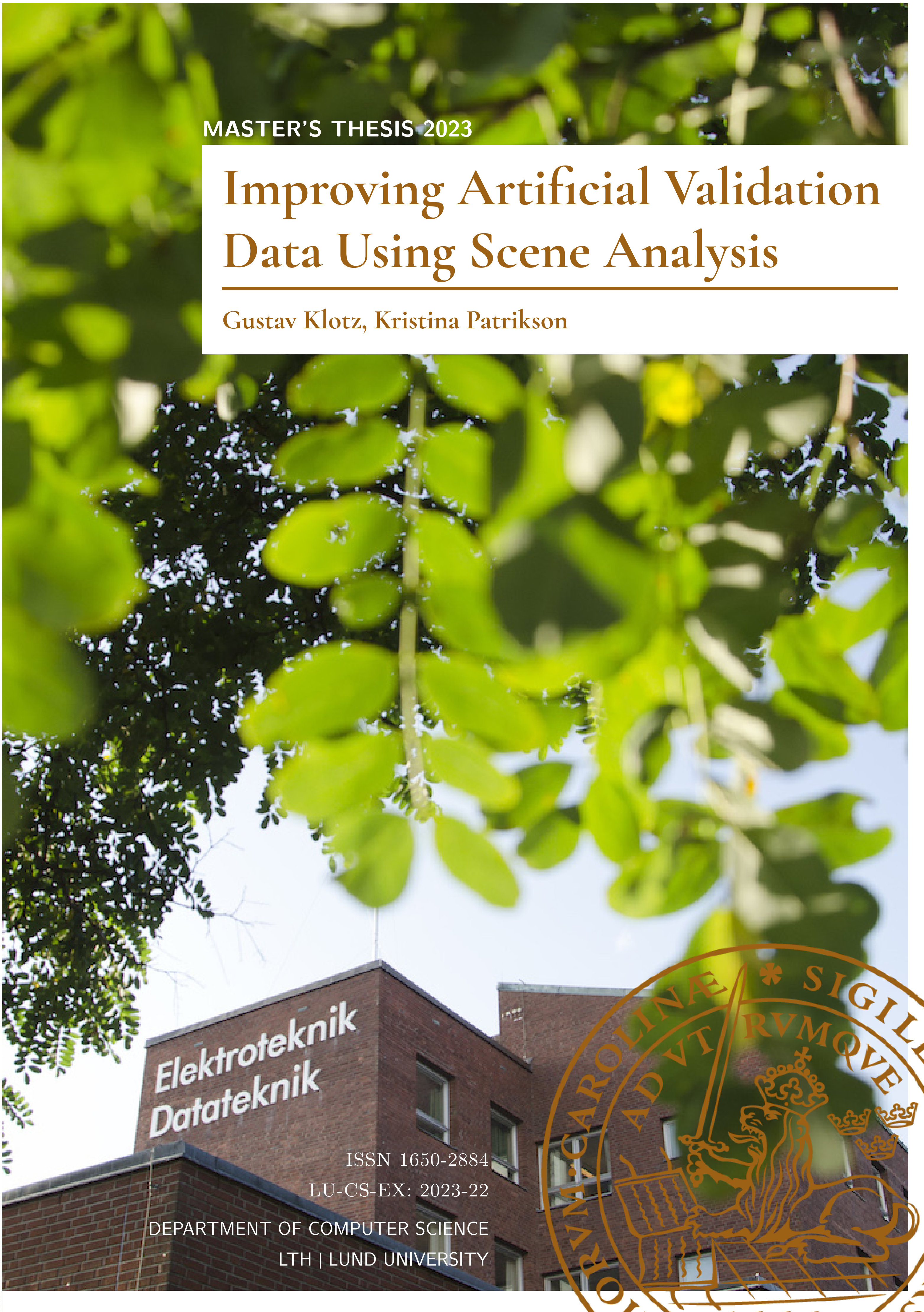


MASTER'S THESIS 2023

Improving Artificial Validation Data Using Scene Analysis

Gustav Klotz, Kristina Patrikson

ISSN 1650-2884
LU-CS-EX: 2023-22

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-22

Improving Artificial Validation Data Using Scene Analysis

Förbättrad artificiell valideringsdata med hjälp av bildanalys

Gustav Klotz, Kristina Patrikson

Improving Artificial Validation Data Using Scene Analysis

Gustav Klotz
gustav.a.klotz@gmail.com

Kristina Patrikson
kristina.patrikson@gmail.com

June 20, 2023

Master's thesis work carried out at Axis Communications AB.

Supervisor: Michael Doggett, Michael.Doggett@cs.lth.se
Examiner: Flavius Gruian, flavius.gruian@cs.lth.se
Assistant supervisors: Linus Jacobson, Linus.Jacobson@axis.com

Mikael Murstam, Mikael.Murstam@axis.com

Abstract

This project explores the possibility of creating virtual validation data by using augmented reality. We have developed a shader that aims to make a virtual car look as similar as possible to a real car for an object detection artificial neural network (ANN), especially in low-light situations. The shader takes the output from a graph cut-based shadow segmentation algorithm in order to match the lighting of the scene. A virtual car rendered with this shader is compared to a realistic model car both by using output values from an object detection ANN and by visual assessment. The results are also compared to another virtual car using a simpler shader that does not take the scene into account. Our results show that our shader makes the virtual car look visually similar to the model car compared to the simpler shader, but it was not possible to show that the ANN found our shader more realistic than the simpler shader.

Keywords: Augmented reality, computer graphics, C++, shadow segmentation, scene analysis, artificial validation data

Acknowledgements

We would like to thank our university supervisor Michael Doggett for his help and for the discussions about new ideas that we had every week.

We would also like to thank our supervisors at Axis, Linus Jacobson and Mikael Murstam, for guiding us through this project and assisting us when we got stuck.

Our gratitude is extended to everyone at Core Technologies Graphics at Axis, for valuable input and help.

Lastly, we would like to thank Gustaf and Filipp for their friendship and support while working on this thesis.

Contents

Chapter 1
Introduction

In the video surveillance industry there are currently many artificial neural networks (ANN) in development for detecting trespassers or certain kinds of objects. These object detection tools can be used for preventing crime by alerting security staff of people or vehicles entering restricted areas, or for other security reasons such as detecting people or vehicles in dangerous areas like construction sites or train tracks. However, one downside of any kind of ANN is the need for huge amounts of training and validation data. It is important to have validation data to ensure that the ANN is able to give reliable results in varying types of situations, like different environments and lighting conditions. For example, the same ANN should ideally detect a trespasser walking outside during nighttime in a snowy landscape, as well as during the day in a busy warehouse. Naturally, some situations are more common than others, and the ANN therefore risks performing worse in more rare scenes and generating false negatives or positives. Recreating videos of these conditions can be difficult, especially since the desired object needs to be present. One way to bypass this problem is to use augmented reality to render a virtual object on top of the real scene. A big challenge of this approach is to match the lighting of the object to that of the scene.

In this project, we have worked with Axis Communications AB to try and augment a virtual car by matching the lighting of a static scene recorded by a camera, using the graphics API OpenGL. The goal of the project is to recreate low-light situations that generate known false negatives for an object detection ANN, but with a virtual car instead of a real car. This would open up the possibility of using the virtual car for generating validation data that can be used to further develop a vehicle-detection ANN.

1.1 Project scope

Matching the lighting of a virtual object in augmented reality is a well known problem with a lot of research. Yet, research on augmented reality using only a single image, especially in unusual lighting conditions, is rare. The main idea used in this project is to create a shadow segmentation algorithm to find the shadows in the image by classifying pixels as either shadow or non-shadow. Then, similarly to Madsen and Nielsen [10], it is assumed that the ratio between shadow and non-shadow areas of the same object is approximately the same in the whole scene. This ratio will be found from the shadow segmentation, and used to color the car and its cast shadow. Additionally, the shadow segmentation will also be used to determine where it is appropriate to render a shadow. To evaluate the performance of the shader based on scene analysis, the same 3D model of the car will be rendered with a simpler shader that only uses the color information from the 3D model and no information from the scene. This simpler shader will be referred to as the *basic shader* in this report, and the shader based on scene analysis will be called the *advanced shader*. Both of these virtual cars will be compared to the ground truth, the real car, and if the advanced shader is more similar to the ground truth than the basic shader is, then this shows that our method has made the virtual car more realistic. This evaluation is partially done using two purely visual comparisons, but the most important part is using an object detection ANN to determine if a virtual car can be detected in the same way as a real car. To our knowledge, using an object detection ANN to compare the appearance of a virtual object to a real counterpart is not something that has been done before.

1.1.1 Limitations

For this project, a single light source is assumed to be present in the scene. Its position, color and strength are assumed to be known, and the car is assumed to be moving on a flat surface. The virtual car will also not affect the lighting of the real scene, apart from casting a shadow on the ground plane. The position of the ground plane is also assumed to be known, at least approximately. This is a valid assumption since the user already has to manually place the car in the virtual space along with its animation points. The ground plane is simply a virtual quad that covers at least the area the car moves in. All image analysis is performed on a single video frame. The car is never occluded by any objects in the scene, but it can be shadowed. Only low-light scenes are used, but there needs to be clear shadows visible in the scene. The materials in the scene should predominantly be diffuse.

1.2 Research questions

The research questions for this thesis are as follows:

- Will the output from the ANN be more similar to the ground truth when using the advanced shader, compared to the basic shader?
- Will the PSNR value between the ground truth and the advanced shader be higher, compared to the PSNR between the ground truth and the basic shader?
- How do the virtual cast shadows and blur impact the PSNR value and output from the ANN of the virtual car compared with the ground truth?
- Does the advanced shader make the virtual car look more similar to the ground truth, from the perspective of a human observer?

1.3 Contributions

By investigating how to create a shader that adapts based on a single image of the background, this thesis contributes to the research field of augmented reality. Furthermore, the project compares how a vehicle detection ANN reacts to a video of a real car compared to a similar video with a virtual car. This explores the possibility of using augmented reality to create artificial validation data, specifically in low light scenes with a single light source.

There are some ethical and societal concerns regarding both the use of augmented reality and object detection that need to be addressed. Firstly, realistic artificial videos might be used to spread disinformation that can be harmful to governments, corporations or individuals. When developing and implementing methods for generating artificial data, it is therefore important to use this data wisely. In regards to creating artificial validation data for an ANN, one benefit of using virtual objects rather than real objects is that this removes many privacy concerns. With a virtual car, there is no problem with storing identifying aspects such as license plates that otherwise would need to be censored or protected. It might also safer to use a virtual car for scenes where it could be dangerous to use a real car, such as building sites or train tracks.

Additionally, there is a risk of bias when using artificial data for either training or validation of a model. Since our project will only be used for validation, it cannot directly affect the training, but if the validation is faulty in some way, it could still lead the model astray. Of course, even real data can be bad, but for artificial data there will always be an extra step, a disconnection from the real world, to be aware of. It could therefore be problematic if too much trust is put in artificial data. This could lead to situations where the ANN does not work properly in critical situations, for instance not detecting a car standing still on train tracks as a train approaches. However, as a risk of bias exists from lack of real validation data as well, this project aims to counter an existing problem of object detection.

1.3.1 Work distribution

Most of the code was written in collaboration, with both people looking at the same screen and discussing what to write next. An equal amount of work from both people was also put into the writing of the report. Apart from that, a few things were handled separately.

Gustav was responsible for implementing and tuning the graph cut algorithm and for processing the output from the ANN. This included parsing, removing outliers and matching videos. Kristina was most responsible for finding relevant research, investigating how to improve and implement the histogram equalization and matching the animation of the virtual car to the real video.

When filming the videos for the ground truth, Gustav controlled the movement of the model car and Kristina handled the filming, as well as labeling of videos.

1.4 Structure of this thesis

The **Introduction** includes motivations and research goals, as well as other related research.

The next chapter, **Theory**, provides the theoretical background that the work is based on. The chapter contains, among other things, an introduction to computer graphics, graph cut algorithms and histogram equalization. The chapter **Methods** includes the method and experiment setup to collect ground truth data.

The chapter **Results** presents all results related to the research questions. The **Discussion** discusses our approach and our results. This chapter also contains thoughts on future work which could improve the performance. Finally, the **Conclusion** answers the research questions presented in the introduction.

1.5 Related work

Augmented reality can be used for many different purposes, and therefore different kinds of assumptions can be made depending on the situation and available resources. In general, it is very difficult to match the lighting of a virtual object to that of a real scene from just a single frame. In fact, obtaining the light information from just a single frame is an ill-posed problem [10], which means that assumptions of the scene or additional information are necessary in order to find a unique solution. There are also many factors that make up the light conditions, such as how many lights are present, what types of light sources they are and the color and intensity of the light sources. To limit the scope of this project, the position of the light source is assumed to be known. In this section, we will present some previous research on matching the light conditions in augmented reality and obtaining light information from an image, along with their assumptions and limitations.

1.5.1 Finding light information with known light position

In an outdoor environment, the light direction can be found by assuming that the time, date, position and orientation of the camera is known and in that way calculating the position of the sun, as is done in a paper by Madsen and Nielsen [10]. In this paper, they assume that the sun can be modeled as distant disc light source, and the sky can be modelled as a sky dome which gives the scene some ambient light. The radiance and irradiance from the sun and sky is found by finding the relationship between areas in shadow and non-shadow, together with the calculated position of the sun and the normal of the ground plane. The shadow and non-shadow areas of the scene are found with the help of a soft-shadow segmentation algorithm based on graph cuts. The algorithm only uses information from a single image to segment the shadows, with the only assumption being that the scene only contains diffuse surfaces. This approach could be suitable for outdoor images taken during day-time, but not so much for images or videos taken during nighttime when the sun is no longer the primary light source. The test set for this project will include images with one primary light source in an otherwise dark environment, which is a very different situation compared to what this algorithm was made for. But perhaps the light source in our images could be modeled as the sun in this method. It seemed like the main difference would be that the user would have to know the position of the light source, rather than to calculate it using the time and geographical location. Another potential problem is that the light from the sun is assumed to be directional due to the fact that the sun is very far away from the earth, whereas the light sources in our images are point lights. This could lead to slightly inaccurate results, but it might be possible to adjust the model so the vector of the incoming light is different depending on the where we are in the scene like for a point light, rather than static as is the case for directional light.

This method seemed to be quite suitable given the limitations for the project. However, when reading more closely it became clear that not enough information was given in the paper and that some additional limitations by the authors are present. The shadow segmentation algorithm runs in real-time, thus adapting as the scene changes which could be a very positive thing if one wants to record the same outdoor scene for a long period of time. Nonetheless, the algorithm has to be initialized with the mean ratio between objects in shadow and objects outside shadows. This initialization step should ideally be automatic, but in the paper they manually find this ratio by taking the mean value of the ground floor in shadow divided by the mean value of the ground floor in non-shadow. With this ratio, they are then able to use their algorithm to find how strong a shadow is at any given point in the image, therefore making it possible to detect soft shadows as well. But demanding that the user knows the mean ratio between shadow and non-shadow might be asking for too much in our case. Therefore it was decided to not use the approach by Madsen and Nielsen.

Some papers on shading virtual objects in augmented reality assume that there is a 3D model of the real scene available [7]. How detailed this mesh has to be and the way it is used differ a bit between papers. Jacobs et al. use a method that renders realistic shadows from a single image and with a corresponding 3D model of the scene [7]. The 3D model consists of both the geometric information and the textures, but the geometric reconstruction can be of poor quality and the algorithm will still give realistic shadows according to the authors. If a 3D model is not available, another way of acquiring 3D information of the scene is by using a camera with a depth sensor. Jiddi et. al. [8] present a method that uses a RGB-D camera to estimate the positions of multiple light sources and their corresponding intensities. This method is also adapted to work for scenes where textured surfaces may be present. They use the 3D information to estimate the illumination ratio of the scene, and through an iterative process find the light sources whose shadow maps most line up with the estimated illumination.

Chapter 2
Theory

The focus on this section lies mostly on different parts of computer graphics that are important to comprehend in order to understand the rendering of the virtual car. Some other relevant subjects for the project are also described.

2.1 Types of light

Three common types of light in computer graphics are ambient light, point light and directional light [1, p. 67]. Ambient light means that all points in the scene receive the same intensity. Point lights have an intensity that decreases with a factor of $1/r^2$ where r is the distance from a given point in the scene to the point light. Finally, directional light is when the rays from the light are parallel. For example, the sun is often modeled as a directional light, since it is so far away and therefore the light rays are assumed to be in the same direction.

2.2 Virtual 3D models

The most simple geometric shape in computer graphics is the triangle, where each corner is a point with known coordinates in a 3D space [1, p. 463]. These points, called vertices, can possess other attributes like normal direction and color. Typically, to create a more complicated shape, a mesh is created by connecting a large collection of triangles together. Each triangle that makes up the mesh is then called a polygon. However, to render the model to a screen to make it visible more steps are needed. Firstly, the coordinate space for the model is not the same as for the whole scene itself. Otherwise it would be difficult to coordinate having several objects in the scene. Secondly, the computer needs some way of determining how to color all parts of the 3D model on a 2D-screen. These things are done using shaders, which will be described further in the section below.

2.3 Shaders

In order to render an object in computer graphics, a shader needs to be attached to the object. A shader in OpenGL consists of two parts, the vertex shader and the fragment shader [11]. The vertex shader is called for each vertex in the model and can perform mathematical operations on them. The only mandatory operation is to transform the coordinate system of the vertex position from model space to clip space and give the result as an output. For more information about the clip space we refer to *Real-time rendering* by Akenine-Möller et al. [1, p. 18], but in short it is a coordinate space which is used to determine which projection the camera uses (e.g. orthographic or perspective) and which 3D points are too far away or too close and therefore should not be rendered. The fragment shader is called for each pixel of the screen that the object covers to determine which color they should have [11]. A very simple fragment shader could perhaps be to color every pixel of the object white, creating a white silhouette. For more complex fragment shaders additional information might be needed, such as more output from the vertex shader like vertex normals and colors. If there is at least one light source in the scene, information about it could be passed to the fragment shader to determine where the object should have shadows etc. The fragment shader could also be used to make objects appear more shiny or matte, or to add a texture to the object. From this, it is probably clear that different shaders can give an object very different appearances. What type of shader to use depends on what the desired outcome is. In OpenGL, shaders are written using a specific shading language called GLSL (Open**GL** **S**hading **L**anguage).

2.3.1 Phong shading

Phong shading is a popular shading model in real-time graphics. It approximates that the reflection of all materials consist of three parts [11]. An example of the three parts are shown in Figure 2.1

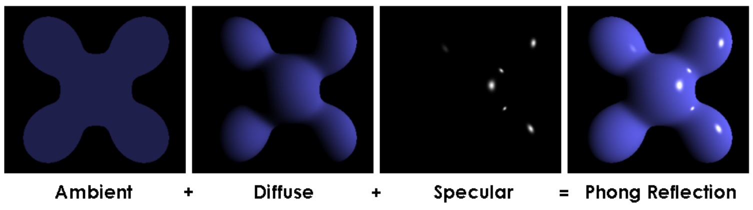


Figure 2.1: Visual example of the three components of the Phong reflection model [22].

To properly describe Phong shading, a few variables need to be introduced. For any point on a surface, there is a normal that points outwards from the surface and a view vector that points from the point of the surface to the camera. Also, there is a light vector that points from the point to the light source and finally a reflection vector that is the light vector mirrored in the normal. Figure 2.2 shows these four vectors in the same image.

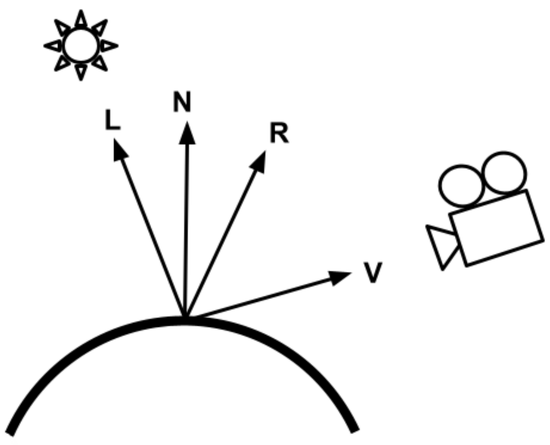


Figure 2.2: Vectors used to calculate Phong shading. The vector L is the light vector, N is the normal of the surface, R is the reflection vector and V is the view vector.

These vectors are then used for calculating how much light shines on each part of the object. The mathematical equation for calculating the intensity of a light source on an object is as follows,

$$I = k_a \cdot L_a + k_d \cdot L_d \cdot \max(\mathbf{l} \cdot \mathbf{n}, 0) + k_s \cdot L_s \cdot \max(\mathbf{r} \cdot \mathbf{v}, 0)^{\alpha}, \tag{2.1}$$

where k_a, k_d and k_s describe the color of the light from each type of reflection (ambient, diffuse and specular) and where L_a, L_d and L_s describe how much incoming light each part contributes with. The vectors **l**, **n**, **r** and **v** correspond to the vectors shown in Figure 2.2. The final term α determines how concentrated the specular highlight is. If α is a large value then the specular highlight will cover a smaller area, giving the appearance of a very shiny object.

2.4 Color spaces

The most common way to store color information about an image is to use the RGB color space. The color of each pixel is determined by looking at the contribution from the three color channels red, green and blue. Most commonly, each channel can have a value between 0 and 255, where a black pixel corresponds to all color channels being 0 and a white pixel means that all color channels have the value 255. When performing image analysis on an image, the RGB color space might not always be the best way of extracting the desired information. An alternative way of describing color is to use the HSV color space [9]. Similarly to RGB, HSV uses three channels. The three channels describe the color's hue, saturation and value. The range for hue is $[0^{\circ}, 360^{\circ}]$, where a low value represents red colors and higher values become yellow, green, cyan, blue and magenta. The saturation and value range from 0 to 1. Another approach is to use the CIELAB color space, also referred to as the L*a*b* color space. The first channel L* describes the perceptual lightness of the color, whereas a* places the color on a red-green scale and b* places the color on a blue-yellow scale [12]. The L* channel has the range $[0, 100]$, where 0 is defined as black and 100 is white. The a*-channel is usually clamped to be in the range $[-128, 127]$, where negative values are more green and positive values are more red. Similarly, b* is also usually clamped to be in the range $[-128, 127]$, where negative values are more blue and positive values are more yellow. In a pixelwise comparison between two images, there are several things that need to be taken into account. One such is which color space to work in. The default color space of RGB is good because of its simplicity, but is commonly regarded as unfaithful to how changes in color are seen by the human eye [14]. That is, a distance in RGB color space might not accurately represent how the human eye would describe the difference between two colors. Such a color space is called *not perceptually uniform*. An example of a color space that is perceptually uniform, however, is CIELAB. The HSV color space is *approximately-uniform*.

2.4.1 Converting an image to grayscale

There are a few different methods that can convert an image to grayscale. A grayscale image has a saturation of 0 in the HSV color space, meaning it can be used to reduce the saturation of the base image. Assuming the image is in the RGB color space, the simplest solution is to calculate the mean of the three channels for each pixel and use that for every channel. However, this is not an optimal solution. Often, the resulting grayscale image is too dark. This is because RGB color space does not correspond directly to how our eyes perceive color. In reality, the different color channels need to be weighted in order to produce a grayscale image that looks good. One common way of doing this is to convert each RGB pixel to a single luminance Y value in XYZ color space, as defined in the CIE 1931 color space [17]. The equation for the conversion is as follows.

$$Y = 0.2126R + 0.7152G + 0.0722B, \tag{2.2}$$

where Y is the luminance and R, G and B represent the color channels in RGB color space.

2.5 Histogram equalization

For a grayscale image with only one color channel, histogram equalization can be used to increase the overall contrast [15]. The main idea of this technique is to first create a histogram of the image with 255 bins, for each possible pixel value, and then redistributing these bins uniformly.

2.5.1 Contrast limited adaptive histogram equalization (CLAHE)

Since histogram equalization takes the entire image into account when increasing the contrast, there could be cases where the local contrast is not increased as much in certain regions of the image. Adaptive histogram equalization (AHE) works by dividing the image into regions, and performing histogram equalization in every region [16]. This would increase the contrast locally in this region. However, this also enhances the noise in the image, which might not be desirable. A variation of AHE that aims to decrease the noise is contrast limited adaptive histogram equalization (CLAHE). The main idea behind this technique is to clip high peaks of the histogram at a certain threshold, and redistributing the clipped values equally over the whole histogram.

2.6 Shadow mapping

While shaders such as Phong shading can render self shadows onto objects, shadow mapping is the process of rendering cast shadows from objects. In order to calculate where the shadow should be, a depth map from the light source's position is needed. A depth map is a grayscale texture where the value of the pixel describes the depth at that point. This makes sure that only the polygons closest to the light source can cast a shadow. When rendering the shadow from the main camera's perspective, the idea is to take each pixel of the ground floor and transform it to find the corresponding point in the depth map. If the depth between the light source and the transformed point is bigger than the depth from the depth map, it means that some other object exists between the point and the light source. Therefore there should be a shadow at that pixel. Depending on if the light source is directional light or a point light, the transformation between the camera's perspective and the light's perspective is either an orthographic projection or a perspective projection. The difference between these two projections is shown in Figure 2.3.

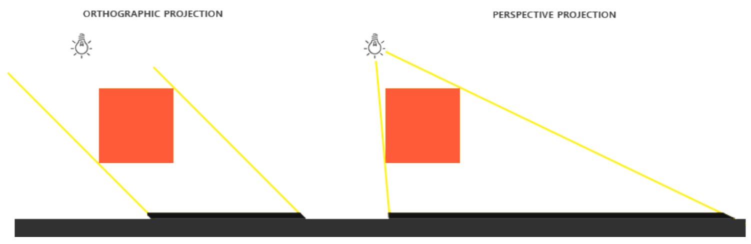


Figure 2.3: Visual example of the orthographic and perspective projection [20].

The results from this method will likely give shadow edges that look a bit pixelated due to low resolution. This happens because the depth map has a fixed resolution, which results in multiple fragments sampling from the same depth value in the depth map. This effect can be reduced by increasing the resolution of the depth map or decreasing the size of the area that the shadow map should take into account. Something to note is that the angle of the light rays from the light source and the ground plane also affects the rate of pixelation. A low angle between the light and the ground will result in more pixelation, because each pixel in the depth map corresponds to a greater area on the ground plane.

2.7 Image segmentation via graph cuts

Graph cuts can be used to partition an image into two disjoint sets of pixels, by creating a graph based on the image for instance as described by Boykov and Jolly [3]. Every pixel in the image is seen as a node, and 4-neighboring pixels are bound to each other with edges. In this report, these nodes will be referred to as pixel nodes, and the edges between them as neighbor edges. Additionally, there are two unique nodes called *terminals*. Individually, they are called the *source* and the *sink*, or **s** and **t**. From each of these two nodes, edges are drawn to every pixel node in the image. These edges are henceforth called *terminal edges*. This means there are virtually three sets of edges: edges between two pixel nodes, edges between a pixel node and the sink, and edges between a pixel node and the source. These edges can be either undirected or directed. In either case, they are all weighted, and the weights are what decides if the segmentation is successful. An example of how image segmentation via graph cut can look like is shown in Figure 2.4.

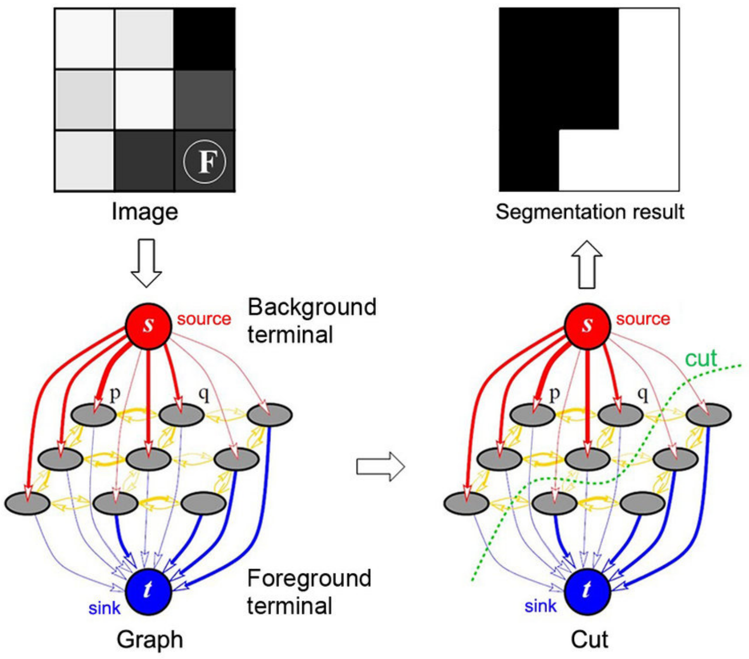


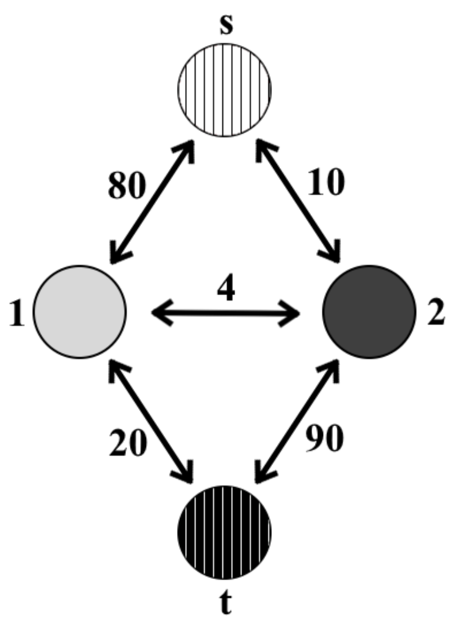
Figure 2.4: Visual example of an image segmentation via graph cut [23]. Note that in this report, the goal of the segmentation is not to segment foreground and background but rather shadowed and non-shadowed areas of the image.

To make a cut in a graph means that a set of edges is removed (or cut) from the graph such that there is no way to travel from the source to the sink. This specific graph cut is a so-called minimum cut, meaning the cut that minimizes the total weights of the removed edges is chosen. To find a minimum cut, a common process is to first solve the maximum flow problem for the graph. There are several well-known algorithms that can be utilized to do this [2][4][5].

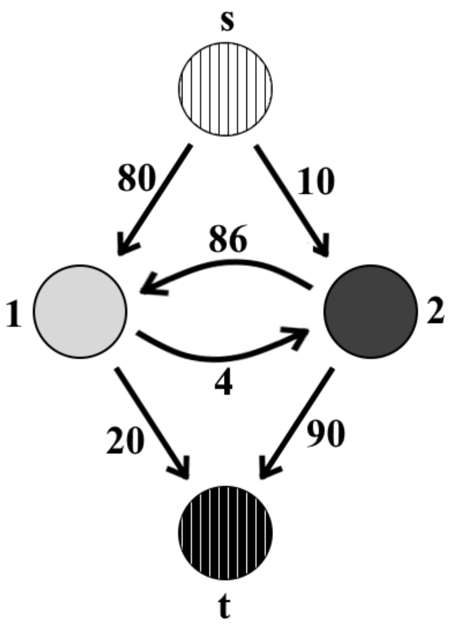
Since the minimum cut minimizes the total weights of cut edges, that means that a high weight on an edge between two pixels leads them to be closely connected, meaning it is difficult to separate them. On the contrary, a low weight means the pixels can easily be separated from each other, and a weight of 0 would mean they are separated by default.

One might ask how exactly the two parts of the segmented image are obtained from the graph. The trick lies in the terminal nodes (the source and the sink) and the fact that they are initially both connected to all nodes. After the graph has been cut, we know there is no way to go from the source to the sink. Since no more edges are cut than necessary, this means that all pixel nodes are connected to either just the source, just the sink, or neither of the source and the sink. All that is needed, then, is labeling all pixels according to whether they are connected to the source or not.

The most critical part of this algorithm is choosing the weights. As stated before, a high weight on an edge makes it more difficult to cut while a low weight makes it easier to cut. The process of choosing the weights depends entirely on the application, on what the two segmented parts of the background should represent. Something else to consider is whether to use a directed graph or not. To make things clear, a minimum cut can be found in essentially the same way for both a directed and an undirected graph, by finding the maximum flow. However, some algorithms for maximum flow work differently depending on the type of graph. If an undirected graph is used, that means there is one neighbor edge between every pair of neighboring pixel nodes. If a directed graph is used, there are instead two neighbor edges between each pair, because the flow needs to be able to go both ways between pixels. For the terminal edges, however, the flow is always going in one direction - out of the source, into the sink - which makes those edges virtually unchanged. A visualisation of the two kinds of graphs can be seen in Figure 2.5.



(a) An undirected graph



(b) A directed graph.

Figure 2.5: A comparison between two types of graphs. The weights on the edges represent capacity, not flow. Both graphs have two pixels, numbered 1 and 2 and marked with different shades of gray. The terminal nodes are marked with horizontal lines. The source **s** is on the top, and the sink **t** is at the bottom. The edge weights are not exact but should still aim to give an indication of range and perspective between different edges.

The advantage that comes from using directed graphs is that the neighbor weights do not just represent the probability of the two pixels belonging to different classes, they represent the probability of one of the pixels specifically being classified as one class and the other as the other class. This has the potential of changing the results in some situations.

2.8 Post-processing

In this context, post-processing refers to many different operations that can be done to alter an image after the rendering has been completed. Because of this, it is done mostly in two dimensions, and it can take into account more than just one vertex or pixel at once. If an image consists of multiple rendered objects, then post-processing can be performed separately for each one, with different operations, before combining the results. Some examples of post-processing operations are blur, inversion of colors, noise, pixelation or grayscale. In this report, only blur is used, but it is not unthinkable that other operations could be used to good effect as well.

2.8.1 Blur

One example of post-processing is blurring. There exists multiple ways to blur an image, and which to use depends mainly on how much the image should be blurred. A common model when working with blur and several other image processing operations is to use a kernel and a convolution. The kernel is a matrix that determines how to add or subtract different pixel values to get different results. In other words, the kernel shows how the neighbors of a pixel and the value of the pixel itself influence what the post-processed pixel value should be. Through convolution, the kernel is then applied to each pixel in the image. Some examples of kernels are presented in Equations (2.3), (2.4) and (2.5) below.

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \tag{2.3}$$

The kernel in Equation (2.3) blurs by using the average of the pixel and its 8-neighbors, weighting them all equally. This is called 3×3 box blur.

$$\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} \tag{2.4}$$

The kernel in Equation (2.4) blurs by using the pixel and its 8-neighbors, weighting the pixels closer to the middle higher than the ones further away. This specific matrix is an approximation of Gaussian blur.

$$\frac{1}{25}\begin{pmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{pmatrix} \tag{2.5}$$

The kernel in Equation (2.5) blurs by using the average of the pixel and the 24 pixels closest around it, weighting them all equally. This is called 5×5 box blur.

2.9 Peak Signal-to-Noise Ratio (PSNR)

The PSNR is a measurement that can be used for pixelwise comparison of two images [13]. The value is in decibels and the higher PSNR two images have, the more similar they can be assumed to be. The equation for calculating the PSNR is shown in Equation 2.7, where MAX_G is the highest pixel value of the ground truth image. The PSNR value is calculated using mean-square error (MSE), which is shown in Equation 2.6. The ground truth image is notated as G, and the other image as V. The values w and h are the width and height of the images in pixels. Since the images are represented in the RGB color space, the MSE includes a division by 3 and the third sum with k.

$$MSE = \frac{1}{w \cdot h \cdot 3} \cdot \sum_{i=1}^{w} \sum_{j=1}^{h} \sum_{k=1}^{3} (G(i,j,k) - V(i,j,k))^2 \tag{2.6}$$

$$PSNR = 20 \cdot \log_{10}(MAX_G) - 10 \cdot \log_{10}(MSE) \tag{2.7}$$

Chapter 3
Methods

3.1 Generating ground truth

In order to create different darker lighting conditions that the ANN struggled with, ideally the lighting should be controllable in the scene, and the environment should be relatively easy to change. This is difficult to do with a real car, since it requires a lot of space and that would make it hard to control the lighting. Therefore, a small model car was used instead in a dark room where the type and amount of light could be easily controlled. An image of this model car is shown in Figure 3.1. The car was moved using thin threads, where the color was chosen to be as unnoticeable as possible. The movement of the real car was made to be as even as possible, to make it easier to match the animation of the virtual car. In addition to this, the placement of the camera in relation to the scene would ideally be similar to the way real surveillance cameras are installed. Otherwise there would be a risk of the false negatives occurring due to the change in perspective rather than light. Usually, cameras are placed fairly high so that they are looking down at the scene in order to capture as much of the area as possible. This was also easier to achieve when having a smaller scene.

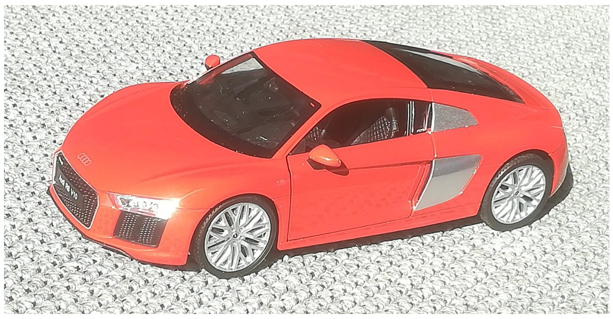


Figure 3.1: The model car used to create the ground truth. The scale is 1:24 to a real car.

It was known that the ANN is more likely to generate false negatives in low light situations with a strong visible light source in the image. Therefore, the majority of the ground truth data was captured under such lighting conditions. All images included one main light source, in or out of frame, with generally low levels of ambient light. The ambient light was regulated by additional low-intensity lights lit from far away. The intensities of the ambient light and the main light source were chosen in such a way that it was possible for a human to detect the car from the camera feed. For the algorithm to work, the scene requires strong cast shadows to be present somewhere at the lower half of the frame. To create these shadows, two boxes of different colors and materials were placed in the scene. The boxes were placed so that they would not occlude the real car in any way in the videos, since this behaviour would be impossible to easily recreate with the virtual car. The material of the ground floor was chosen to be as smooth as possible, to avoid self shadows that might make the shadow segmentation more difficult. In Figure 3.2 four screen shots from a ground truth video are shown. Similar screenshots from all of the ground truth videos can be seen in Appendix A.

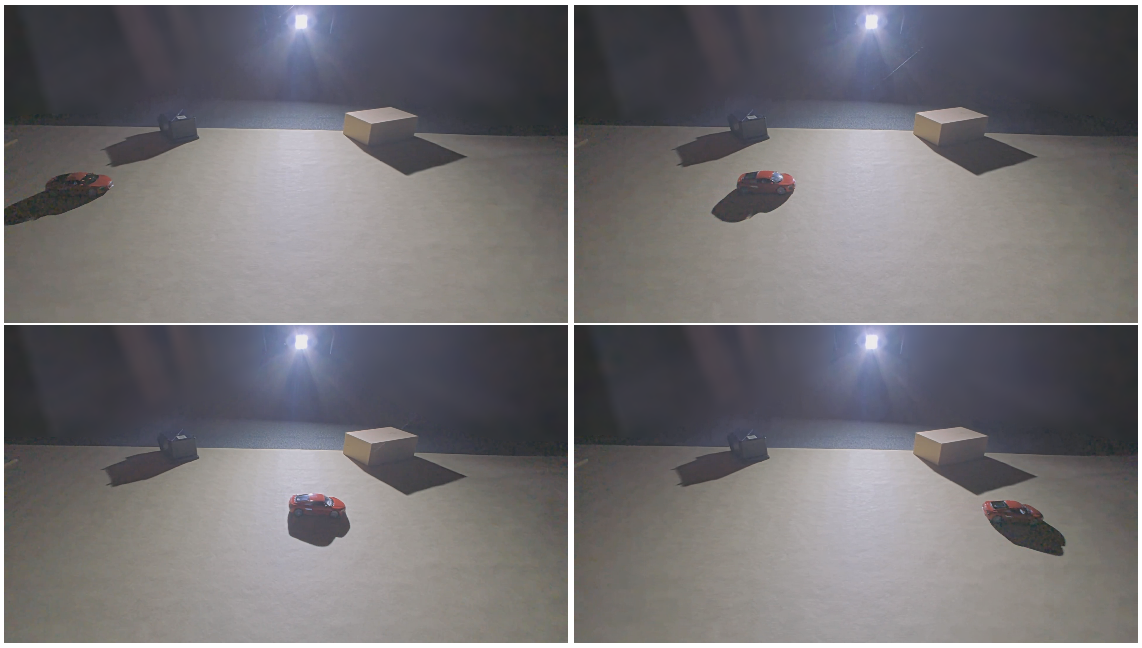


Figure 3.2: Screenshots from a video filmed as ground truth. The car moves from the left to the right over time. Note that parts of the background have been blurred to preserve confidentiality.

3.2 The virtual car, animation and virtual background

The virtual car was found online on the website cgtrader [19], and was created by the user TankStorm. It has 5000 polygons, and the colors were altered to match the real model car. This model was chosen due to the low number of polygons, since too many could make the animation of the car not run as smooth. The car is shown in Figure 3.3.

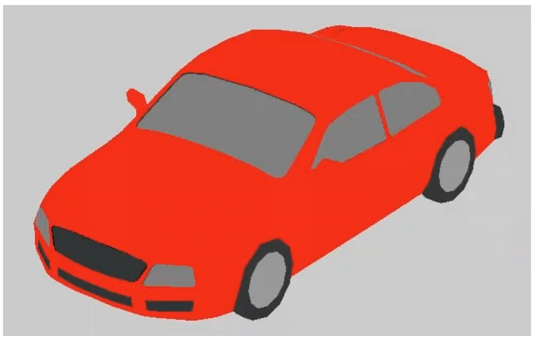


Figure 3.3: The virtual car with the basic shader.

The animation of the real car was manually adapted to be as close as possible to that of the real car. For videos which required heavy computations, the speed of the animation was affected and the car did not move as smoothly due to frame drops. To try and reduce the difference in speed between different virtual cars, the shader was adapted to run all calculations even if it was not necessary (e.g. compute placement of virtual cast shadow for videos where no cast shadow was used). Still, there were minor differences in animation speed, which were manually adjusted to match the video of the real car as closely as possible.

The background scenes for the videos with the virtual car was created by using a screenshot of the real scene and adding it as a texture to a quad that covered the screen. Ideally, the background for the virtual scenario should simply be a live video of the real scene, with the virtual car as an overlay. However, the static background was used instead to make testing more flexible and to save time.

3.3 Shadows

One key part to accurately render a virtual car in a real scene was to give the car a proper shadow. This was done using shadow mapping. A slightly simpler method than the one described in section 2.6 was used since the only virtual object in the scene was the car. Instead of having to compare depths, we could just see where there was any object at all that blocked the light rays and draw a shadow accordingly. This was possible since the car was the only virtual object that could cast a shadow on the ground. The shadow of the car was rendered onto a quad that represented the ground floor in the scene.

This virtual shadow also had to match the real shadows in direction and strength. To achieve this, the real shadows in the scene needed to be analyzed. It was assumed that the position, color and strength of the light source in the scene was known, since they would have been complicated to find by image analysis. Finding the light position in the image would have been relatively simple in an image where the light source is visible, just by locating the brightest part, but to then determine the depth along the possible line would have required much more information. We would have had to find shadow edges and triangulate their vectors, and that would only have given a rough estimate. Finding the color of the light source would have been possible if the position had already been found, but the strength would have been harder.

In order to calculate the proper light levels of the virtual shadow, the shadowed areas of the scene needed to be segmented from the directly lit areas. A graph cut algorithm was used for this. Then, similarly to the method proposed by Madsen and Nielsen [10] , three constants were calculated, one for each channel in RGB color space, corresponding to the ratio between lit and shadowed areas. The assumption about the relationship between light and shadow in the scene is illustrated in equation (3.1).

$$\begin{bmatrix} R_{shadow} \\ G_{shadow} \\ B_{shadow} \end{bmatrix} = \begin{bmatrix} r_R & r_G & r_B \end{bmatrix} \cdot \begin{bmatrix} R_{light} \\ G_{light} \\ B_{light} \end{bmatrix}, \tag{3.1}$$

where r_R, r_G, r_B are the three constants. These ratios were then used to convert lit pixels in the scene into shadowed ones, in order to create a virtual shadow for the car. The shadow segmentation algorithm was implemented in MATLAB, because it has support for many image analysis techniques. An attempt was made to implement the shadow segmentation algorithm in C++ and run it on the camera, but due to time constraints this could not be completed. From MATLAB, the segmented image and statistics regarding the color of shadow and non-shadow areas were exported to the camera, where the rendering was performed.

3.3.1 Shadow segmentation

Originally, a thresholding algorithm [12] was considered for the shadow segmentation, but it proved to be too inflexible. It could perform well on one scene and then perform significantly worse on another, with no way to adapt between them. By using a graph cut algorithm, the goal was to counteract some of these problems. The flexibility of being able to choose the edge weights freely and the fact that the graph could take the entire image into account, rather than analyze one pixel at a time like thresholding, were the main perceived advantages. A disadvantage, however, was that the graph cut was more complex to both implement and use, but we still deemed it necessary.

In this application, the focus was on segmenting shadow and non-shadow areas of the image. This fact was fundamental in choosing the weight functions. Shadow areas were defined as all places where the light from the light source could not reach. Since these areas were generally darker than the non-shadow areas, the most simple and natural way to discern whether a pixel is likely to be in shadow or not was to look at the light level of the pixel. This was done using the value channel V in the HSV color space. Most of the weights were based on the values of the pixels, after converting them from RGB color space to HSV. To increase the contrast of the V-channel, histogram equalization was used. For images where the light source was visible to the camera, the shadows closest to the light source would be wrongfully seen as brighter by the camera because the light bleeds out from the light source. This behaviour was made even more apparent after applying histogram equalization, often leading to these pixels being wrongfully classified as non-shadow pixels by the graph cut algorithm. Using CLAHE instead of regular histogram equalization made these shadow pixels darker, resulting in a generally more accurate shadow segmentation. For scenes where the light source was not visible by the camera, using CLAHE instead often worsened the shadow segmentation somewhat. Therefore CLAHE was only used for scenes where the light source was visible by the camera.

Weight functions

This section describes in detail how the graph was constructed and how the weights were determined. A directed graph was used, which does not impact the source and sink weights but changes the approach for the neighbor weights greatly. First, the sink was chosen to correspond to shadow pixels and the source was chosen to correspond to non-shadow pixels. Thus, the edges between the pixel nodes and the sink were weighted according to the probability that the pixel node was a shadow pixel, and similarly, the edges between the pixel nodes and the source were weighted according to the probability that the pixel node was a non-shadow pixel. Deciding the weights on both of these kinds of edges were complicated, for the same reasons. If it was easy to decide if a single pixel was in shadow or not, no graph cut would have been necessary. The problem stemmed from the fact that a shadowed area of a bright color can often be brighter than a lit area of a dark color. However, despite this ambiguity, the light level of the pixel was still regarded as relevant and not something that should be ignored. Therefore, the weights in both of these sets were directly based on the value V in HSV color space. The value V goes from 0 to 1 and corresponds to the brightness of the pixel. We chose to use weights that take into account both the value V after CLAHE and the value V in the original image. The second of these was taken only in reference to the mean of the entire image. For the source weights, the following formula was used.

$$W_s(V_h, V_o) = 60\,V_h + 50\max(V_o - \mu_o, 0), \tag{3.2}$$

where W_s is the weight on the edge between the source and a pixel node, V_h is the value of the pixel after CLAHE, V_o is the value of the pixel in the original image and μ_o is the mean value in the original image.

$$W_t(V_h, V_o) = 50\,(1 - V_h) + 80\max(\mu_o - V_o, 0), \tag{3.3}$$

where W_t is the weight on the edge between a pixel node and the sink, V_h is the value of the pixel after CLAHE, V_o is the value of the pixel in the original image and μ_o is the mean value in the original image.

These were the weights we found to work the best on our set of scenes. We tested many different weight functions and performed graph cuts on several different scenes to determine this. We chose to use the weights that worked well on the greatest number of scenes.

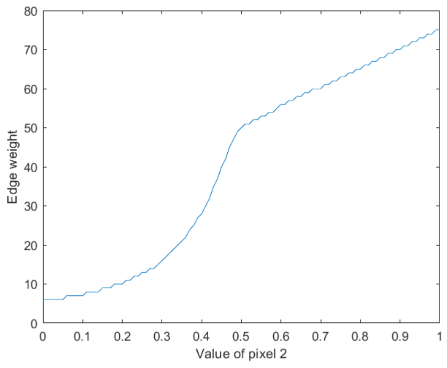
For the third set of edges, the neighbor edges, the fact that we used a directed graph had more importance. The weights were set to correspond to the probability that two pixels belonged to the same class, either shadow or non-shadow. Assume that we have two connected pixel nodes called pixel 1 and pixel 2. The weight on the edge from pixel 1 to pixel 2 would then be based on the probability of pixel 1 being non-shadow and pixel 2 being shadow, and the weight on the edge in the other direction would do the opposite. The exact values of these weights were also nontrivial to determine. Two pixels could have had very different brightness values but still both have been non-shadow, if they were on the border between two objects with widely different colors.

Two different functions were used to calculate the weights between two pixel nodes. Again using the example edge from pixel 1 to pixel 2, the function used depended on which of the pixel nodes had the highest value V after CLAHE. If pixel 1 had the highest value, Equation (3.4) was used. Otherwise, Equation (3.5) was used. In both equations, W_n is the neighbor weight, V_1 is the value of pixel 1 and V_2 is the value of pixel 2.

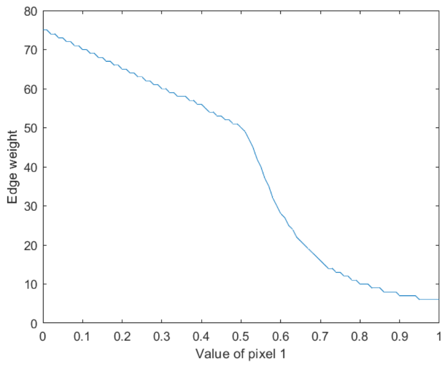
$$W_n(V_1, V_2) = \frac{2}{(V_1 - V_2)^{1.5} + 0.04} \tag{3.4}$$

$$W_n(V_1, V_2) = 50\,(1 - V_1 + V_2) \tag{3.5}$$

After both equations, the resulting weight was rounded up to the nearest integer. A visualisation of the weights can be seen in Figure 3.4.



(a) Neighbor edge weight as a function of the value of pixel 2. Pixel 1 has the value 0.5.



(b) Neighbor edge weight as a function of the value of pixel 1. Pixel 2 has the value 0.5.

Figure 3.4: Visualisation of neighbor edge weights.

These neighbor weights were, like the source and sink weights, determined by performing shadow segmentation on several different scenes with different weight functions. These functions led to the best results overall, giving good results on the greatest number of scenes.

Additional operations

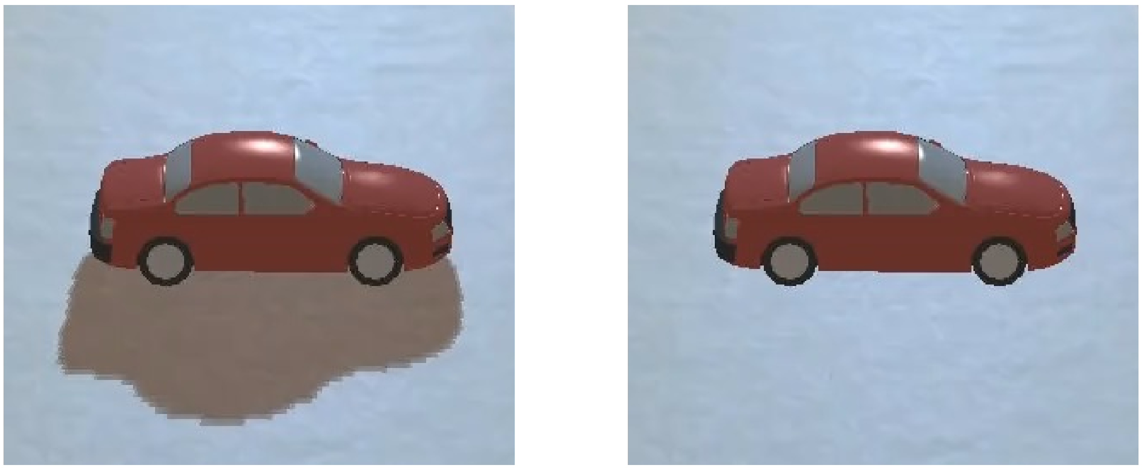
The general rule was to separate pixels with widely different lightness, and keep together those of similar lightness. However, a problem emerged with the fact that color seldom was completely changed from one pixel to another. Even in an image where clear borders between objects could be seen, zooming in often showed that the border was actually gradual over several pixels. This meant that even with a good approximation of how the color of a pixel changes when in shadow, such a comparison could very rarely be made between two neighboring pixels since the entire change was almost never seen from one pixel to its neighbor. This was countered somewhat by lowering the resolution of the image specifically for the graph cut. This also reduced the computational time, since fewer nodes were needed. However, the resolution could not be lowered too much because of the risk of loss of potentially important information. In our experiments, we lowered the resolution to 1/7 both row-wise and column-wise, using bicubic interpolation in MATLAB.

A common technique in image segmentation, for instance used in an article by Boykov and Kolmogorov [3], is to use a seed that is placed by the user. This seed initializes parts of the graph by marking shadow and non-shadow areas, and helps in making a good segmentation. However, we chose not to use a seed, since it would require user interaction that was problematic to achieve for a graph cut that was supposed to be automatic. A seed was not required for our segmentation to work, but it would have made it easier to classify especially bright shadow areas or dark non-shadow areas.

To obtain the shadow ratio r and the mean color of the shadows, only information from the lower half of the image that shadow segmentation had been performed on was used. In all the scenes we looked at, the background was generally a lot darker than the rest of the scene, due to the low-light situation. While it is not necessarily incorrect to classify these areas as shadow, they are not shadows cast by objects and should therefore be excluded from the estimate of r and the mean color of the shadows. By only including pixels from the lower part of the image, the found shadows would be more likely to be cast shadows since these types of shadows are usually found on the ground of the scene. Ideally, it would have been preferable to look at the pixels below the horizon line of the image, but it is difficult to determine where this line is in a given scene.

3.3.2 Virtual shadows on real objects

In order to ground virtual objects in the real scene, the object should cast shadows that affect the real scene. Otherwise, it will become obvious to an observer that the virtual object does not belong in the real world, or might give the impression that the object is floating. An example can be seen in Figure 3.5. It was unknown how the ANN would react to shadows or the lack thereof, but given the shadows' importance for a human observer, they were deemed reasonable to include regardless. For this project, the virtual shadows were achieved by rendering a virtual ground floor underneath the car and using shadow mapping to determine where the cast shadow from the virtual car should be. The mean ratio between shadow and non-shadow areas obtained from the shadow segmentation of the scene was used to determine the color of the virtual shadow. This was done by mapping the coordinates of the ground quad to the coordinates of the screen to get the correct pixel color of the background, and then multiplying that pixel value with the shadow ratio if there should be a shadow there according to the shadow mapping. Note that this method works under the assumption that the ground is flat, and that the shadow is only cast on the ground and not on any surrounding objects.


(a) With a virtual shadow.

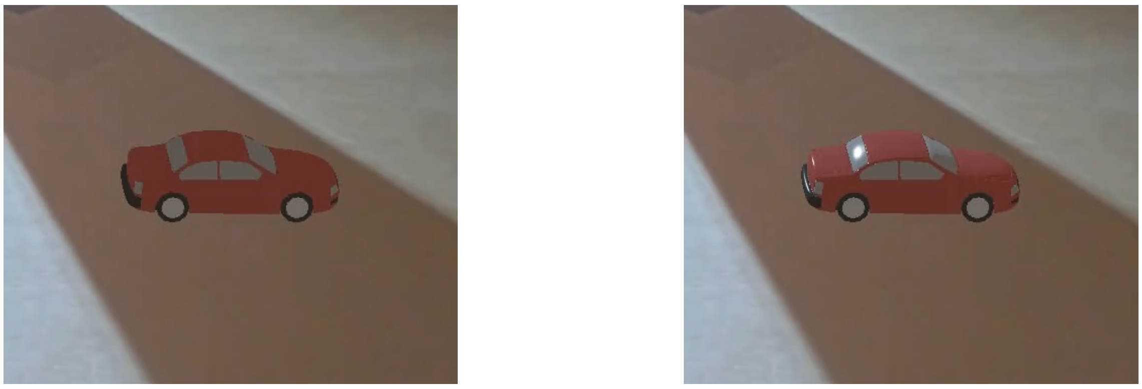
(b) Without a virtual shadow.

Figure 3.5: A virtual car with and without a shadow.

3.3.3 Real shadows on virtual objects

The opposite problem is how to make the cast shadows from the real scene affect the virtual car. In other words, if the car is driving through a real shadow, then the car should become darker. For this, we needed to determine where these real cast shadows should be on the car. To solve this, we used the result of the shadow segmentation, a black-and-white image that segmented the scene into the two areas, shadow and light. Each pixel of the virtual ground plane was then mapped to the corresponding pixel in this binary shadow segmentation image, thus classifying each pixel of the virtual ground plane as either shadow or light. For each pixel of the virtual car, a line was traced from the light source through the vertex connected to the pixel, onwards until it met the virtual ground plane. If the point on the ground was marked as shadow from the shadow segmentation, that meant that the light was unable to reach that point because a real object in the scene blocked it. This meant that the light would not have been able to reach the virtual car either, had it been real. As such, that pixel on the car was marked as shadow and the shader for the car colored that area using only the contribution from the ambient light.

This type of shadow is referred to as an *object shadow* in this report. An example can be seen in Figure 3.6.



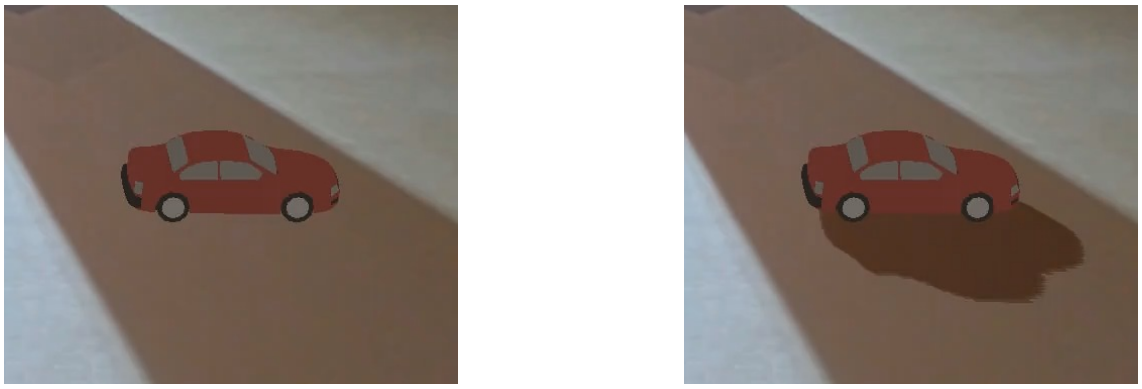
(a) Object shadow.

(b) No object shadow.

Figure 3.6: A virtual car with and without an object shadow. Since the car is in shadow, the left image shows the "correct" lighting.

3.3.4 Double shadows

In most cases a cast shadow from the virtual object is needed to make it look more realistic. Still, there are cases where a shadow can instead have the opposite effect. This happens when the object is placed in an area of the scene where there already is a cast shadow on the ground from a real object in the scene. Rendering a virtual shadow on top of the real shadow will create a so called *double shadow*. An example can be seen in Figure 3.7. Given that the scene is lit mainly by one light source, this double shadow is not very realistic. In other words, this means that it is not enough to just determine what color a shadow from the virtual object should have, but also where it is suitable to render it. Also, if the virtual object's shadow is only partially covered by the real shadow then the two shadows should ideally be the same color and blend seamlessly. This problem was also handled by using the results of the shadow segmentation. Since the segmentation produced an image with light and shadow shown as white and black, it was simple to map the position of the shadow cast by the virtual object and see whether it overlapped with a shadow area in the scene or not. For good results, note that this method required the shadows to be very sharp and that the shadow segmentation was very accurate.



(a) No double shadow.

(b) Double shadow.

Figure 3.7: A virtual car without and with a double shadow. Since the car is in shadow, the left image shows the "correct" lighting.

3.3.5 Creating the shader for the car

From the shadow segmentation it was possible to get an estimated ratio between the shadow areas and non-shadow areas. We could also obtain the mean value for each color channel in shadow and non-shadow areas. The shader for the virtual car was based on the Phong reflection model, and it was assumed that the lighting of the real scene followed this model as well. This meant the shadow areas would correspond to the areas that were (ideally) only affected by ambient lighting. However, the ambient component of the Phong reflection model consists of two factors, k_a and the L_a, where the values in k_a depend on the object itself and L_a depends on the lighting. The mean value of the shadow areas in the scene was therefore not only affected by the ambient light, but also the material properties of the objects in the scene. To estimate L_a from the mean value in the shadows, we therefore needed to make an assumption about k_a. Taking inspiration from the gray world assumption [21], it was assumed that the intrinsic colors of the objects in the scene are gray on average. The ambient light L_a was then found as follows,

$$L_a = \left(\frac{R_{shadow}}{0.5}, \frac{G_{shadow}}{0.5}, \frac{B_{shadow}}{0.5} \right),$$

where R_{shadow}, G_{shadow} and B_{shadow} are the mean values for each color channel in the shadow areas of the scene. The ambient color of the car was then calculated by multiplying k_a from the car with L_a. The diffuse component of the shader for the car could then be found using the ratio between shadow and non-shadows in the following way,

$$\frac{L_a \cdot k_a}{L_a \cdot k_a + L_d \cdot k_d} = r \iff L_d \cdot k_d = \frac{L_a \cdot k_a(1-r)}{r},$$

where r is the ratio between shadow and non-shadow areas. Note that multiplication was done by element between vectors in this case, since we were looking at each color channel independently from each other. The color of the specular highlights was set to $(1, 1, 1)$ for all scenes.

After trying this shader on a few scenes, it became apparent that the virtual car was a lot more saturated in color compared to the real car. The lower saturation of the real car could possibly occur due to having a low light scene, in combination with the type of camera that was used. To try and prevent oversaturation of the virtual car the ambient part of the shader, $L_a \cdot k_a$, was desaturated by a factor s. This factor was found by experimentation on three different scenes in different lighting conditions. For each scene, five different strengths of ambient light were used. In each scene and light situation two images were taken, one image with a panel of 12 different colors placed in shadow and one image without the panel. Additionally, an image was taken of the panel in a well lit situation with white light. This image was also slightly manipulated to better reflect how the colors looked like in real life, since the goal of this image was to acquire an approximate k_a for each color. The scenes were then analyzed to calculate r and L_a for each lighting condition. Together with the information about k_a taken from the reference image described before, the estimated shadow color was found for the colors on the panel. To calculate s for the scene, the saturation of this shadow color was divided by the saturation of the true shadow color, taken from the image which included the color panel in the scene. The final value for s was then found through linear regression. Since the mean shadow color was already a part of the model, we decided that it would not be appropriate to include this as a parameter in the linear regression. The parameters that were tested were the shadow ratio and a constant.

3.4 Post-processing

When looking at the ground truth, the virtual car seemed to show more details whereas the real car was a bit more unfocused. To imitate this, a post-processing effect that would blur the car was created. To apply blur to only the car, we rendered the car to a blank texture of the same size of the screen, blurred it and then rendered the texture to the screen. Since we used blur on a texture, it was useful to modify the kernel slightly. We based the blur on the kernel in Equation (2.3). However, since texture coordinates range from 0 to 1, moving "one pixel" in the convolution was not trivial, and instead we had to move by a float number. We used $\frac{1}{300}$ as our offset, meaning that when the kernel was applied in the convolution, it looked at the pixels offset from the middle pixel by $\frac{1}{300}$ of the screen resolution. In this way, an object would become equally blurred regardless of the screen resolution. To make it more clear, if the screen resolution had been 300 by 300 pixels then the kernel was as shown in equation (3.6). If the screen resolution had been 600 by 600 pixels then the kernel was as shown in equation (3.7). That is to say, pixels two steps away from the middle pixel were used instead of those one pixel away in that case.

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \tag{3.6}$$

$$\frac{1}{9}\begin{pmatrix}1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 1\end{pmatrix} \tag{3.7}$$

3.5 Methods for evaluation

There needed to be a way to evaluate the results, that is the videos of the virtual car, and determine whether they were good or bad, and where improvements were needed. The specifics of how that was done is described here.

First, a statement about what was to be evaluated. As previously stated, the goal was to create a virtual car object that resembled a real car as much as possible, mainly from the perspective of an ANN but also visually. As such, it was logical to compare two videos taken in the same environment. One with a real car and one with a virtual car, the second adjusted to match the first one as well as possible in movement. The evaluation was divided into three separate parts. The first part was a visual assessment, where the real and virtual car were compared by human eyes. This assessment compared for instance lighting, shadows and reflections. The humans in question were ourselves, to save time during the development process, and since this evaluation method was mainly a complement to the other two methods. The second part was also a visual assessment, but was in contrast to the first performed as a direct pixelwise comparison of the same frame in the two videos by calculating PSNR, peak signal-to-noise ratio. That is, a computer made a strict comparison to see how well the visuals matched. The third part was performed by the ANN. It too compared the two videos, but because it was an ANN, it was more difficult to determine what exactly it examined. In any case, the ANN produced bounding boxes that showed where it detected a car. The position of these bounding boxes were output, along with a score that represented how certain the ANN was. The score ranged from 0 to 100, while the position, in x and y coordinates, ranged from -1 to 1. In general, a high score meant that the ANN deemed that the detected object was very likely to be a car, and a lower score meant that the ANN was more uncertain. If the ANN at some point did not detect the car, it was counted as a score of 0. In this report, these three comparisons will be referred to as *human comparison*, *pixelwise comparison* and *ANN comparison*, respectively.

One reason for having multiple ways of comparison was to avoid some ambiguity. The most important of the three comparisons could have been argued to be the ANN comparison, but the score did not communicate what exactly made the ANN react. That is, it could very well be that two cars that looked very different could have gotten very similar scores, at least in some situations. Since the aim of this project was to create a virtual car that gave the same results as a real car in all situations, it was reasonable to ensure that the cars looked similar. If the cars looked similar and the ANN reacted in the same way, it was easier to argue that the goal had been reached than if it was only the ANN.

3.5.1 ANN comparison

In the ANN comparison, both the score and the bounding box positions were compared. For the bounding boxes, the *Jaccard* or *Tanimoto* index [6][18] was used, meaning we calculated intersection over union. If the bounding boxes are denoted A and B and their areas are denoted as $|A|$ and $|B|$, then the Jaccard index can be written as

$$\frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

This method always results in a ratio between 0 and 1, where 0 means the bounding boxes do not overlap at all and 1 means they overlap completely. When comparing two moving objects in different videos, it was important that the movement of the objects could be closely matched to each other. A mismatch could easily lead to an invalid decrease in the Jaccard index.

Regarding the scores in the ANN comparison, there were not as many alternatives when comparing as for the bounding boxes. Since the score was just a single number between 0 and 100, the easiest way to compare them was to calculate the absolute difference, and that is what was chosen. The relative difference or square error would not have been unfeasible choices either.

These output parameters were obtained at an unknown, varying rate, and to easier compare two different setups where the parameters of the virtual car had changed, their means were calculated. For some cases, the entire sequence of values was plotted, for easier visualization and for added clearness regarding where in the sequence the likeness was better and worse. Timestamps were then used to match the sequences in time. Since the output parameters were not acquired at a fixed time interval, the output values were unlikely to match perfectly in time when comparing two videos. To make comparisons between videos possible, interpolation was used to estimate the score and bounding box position at a specific time. However, interpolation would not always be suitable in some situations, since there could be periods of time when the ANN simply did not manage to detect the car. It was decided that if the time difference between two observation was longer than one second, then no interpolated values would be created between them. Thus when comparing two videos, the first observation in each video would be lined up with each other, and then the closest following observation would be found and the corresponding point in the second video would be found by interpolation. If it was impossible to interpolate a point in the second video, the resulting Jaccard index would be 0 for that observation, and the score of the second video would be 0. This was repeated for all observations in the videos. This meant that in instances where both cars were not detected by the ANN, the mean Jaccard index and mean score difference were not affected.

3.5.2 Pixelwise comparison

For the pixelwise comparison, a single frame of the car in approximately the same position was taken from an image with the real car and the virtual car to calculate PSNR between them. The image was cut to focus only on the car, to avoid noise from other parts of the image negatively impacting the PSNR. We wanted the focus to be only on the similarity of the cars, rather than than how similar the videos were as a whole. The PSNR was calculated in the RGB color space.

As mentioned in Section 2.4, one downside of RGB is that it is *not perceptually uniform*, unlike CIELAB. One might wonder, then, why RGB was chosen and not LAB. The reason for that is that there were already two other ways of comparison, one of which was the ANN comparison and the other of which was the human comparison. Since the focus was ultimately on the ANN, and there already was a human comparison, there was no need to ensure that the pixelwise comparison actually used a color space that matched human perception. Also, because the end goal of the comparison was only to be able to present differences between two images in a way could be understood and compared to other pairs of images, the most important thing was just to use the same metric for everything. RGB was hence chosen, for simplicity. However, it is not unthinkable that comparisons made in other color spaces could result in more accurate numbers in some aspect.

3.5.3 Basic and advanced shader

Having stated that the comparison was made between the real car and the virtual car, there was no way to get any perspective just from that comparison. Any results would be ungrounded and without a point of reference. To create a baseline, we used two different shaders, one *basic* and one *advanced*. The basic shader is a shader that only returns k_d of the object, whereas the advanced shader uses the Phong reflection model and takes the output from the scene analysis into consideration. Less formally, the basic shader represented the car model in the state it was before we begun this project and the advanced shader represented the car model's state after we were done. As such, the basic shader could be used as a baseline. Both shaders were used in the comparisons, with the goal of showing that the advanced shader made the virtual car more similar to the real car than the basic shader did.

3.5.4 Scenes, situations and settings

Apart from using two different shaders for the virtual car, we also made use of several different options to create variation and a larger sample size. We used five different scenes, which could vary in placement and strength of light source, ground material and placement of objects. A scene together with a car moving in a certain way was called a *situation*. Mostly, each scene only contributed to one situation, but one scene had three different situations. Within each situation, several different *settings* were also defined. Apart from variation, these settings were meant to be used to investigate and highlight how small changes in the car or shadows impacted the results. We decided to use a setting called 'Regular' as the base setting. Another setting was called 'No virtual shadows', where we did not render any virtual cast shadow for the car. A third was 'Blur', where post-processing was used to blur the car. These settings were tested for the virtual car both with the advanced and the basic shader. A fourth and fifth option were only used in situations where the car passed through shadows in the scene. The fourth was 'Double shadow' and meant that double shadows were not removed. The fifth setting was only used for the advanced shader, and was called 'No object shadow'. This setting had the car not changing appearance from going through shadows. Comparisons between the settings were done in two ways. The first way was to choose one setting and compare the results for that setting with the basic and advanced shader. The second way was to choose either the basic or the advanced shader and then compare the results of two different settings for that shader.

3.5.5 Handling false positives and outliers

When retrieving output data from the ANN, it was clear that some output values were caused by other sources than the car. That is, the ANN sometimes reacted to other things in the scene and treated them as cars. These false positives were not of interest, since the focus was only on how the ANN reacted to the real and virtual car. In most cases, the bounding boxes of these false positives were placed very differently from the bounding boxes of the car, making them easy to identify and remove. In two videos of the ground truth, there were instances when the ANN detected an object with a bounding box that was close to that of the car, but with a different size. These values were kept, since it was possible that these outputs could have been caused by the car and therefore ideally should be recreated by the virtual car as well.

Chapter 4

Results

This section presents the results from the desaturation experiment and the two different shadow segmentations that were tested, as well as comparisons between different versions of the virtual cars in different scenes.

4.1 Desaturation factor

After performing linear regression, the coefficient for the shadow ratio was not shown to be statistically significant, and was excluded from the model. As such, only the constant desaturation factor s remained. The results of the experiments to determine s can be seen in Figure 4.1. The estimated value of s was 0.3413, calculated as the mean of all values in Figure 4.1.

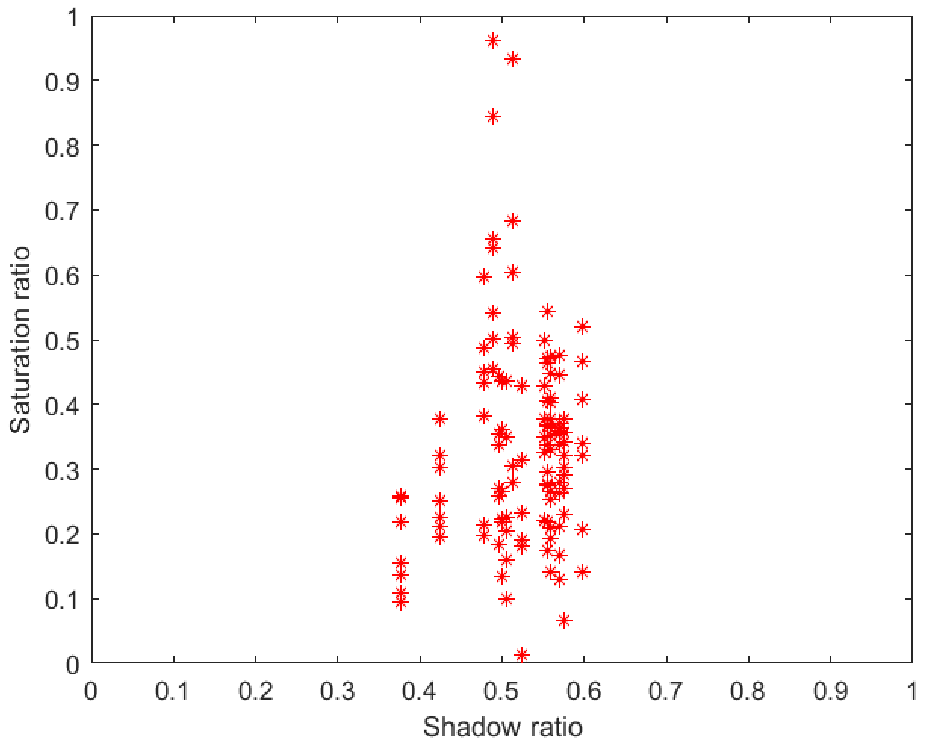
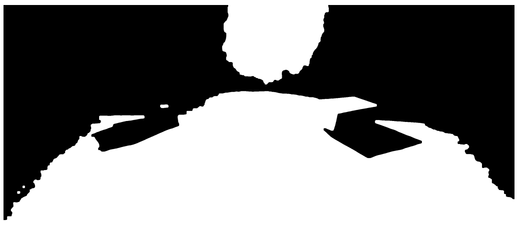
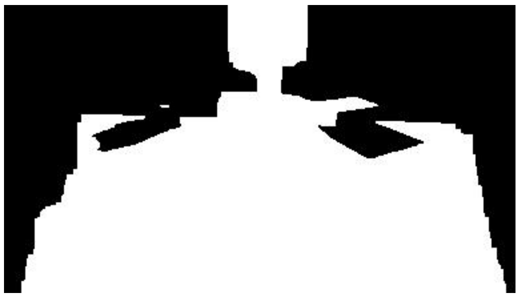


Figure 4.1: Saturation ratio plotted against shadow ratio. The shadow ratio has been calculated as a mean over the shadow ratios of the three RGB color channels.

4.2 Shadow segmentation

Two examples of the shadow segmentation using two different techniques are shown in Figures 4.2 and 4.3 below. For the first scene in Figure 4.2, both the thresholding and graph cut algorithm find the shadows from the boxes and classifies parts of the ground floor closest to the right and left edge as shadow. In Figure 4.3 the threshold technique is unable to find the shadows in the center of the scene. The graph cut algorithm is significantly better at segmenting this area. As a reminder, the shadow segmentation used when developing the shader was the graph cut algorithm.


(a) Thresholding.


(b) Graph cut.

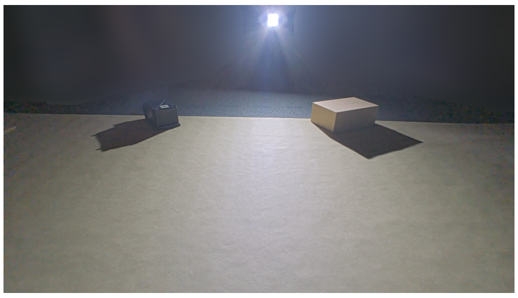
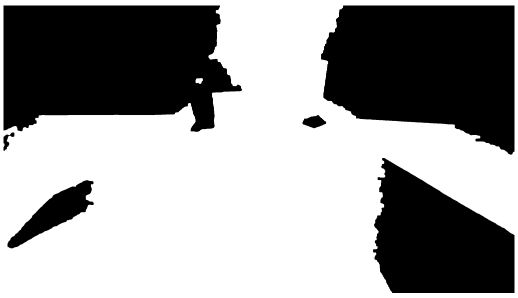
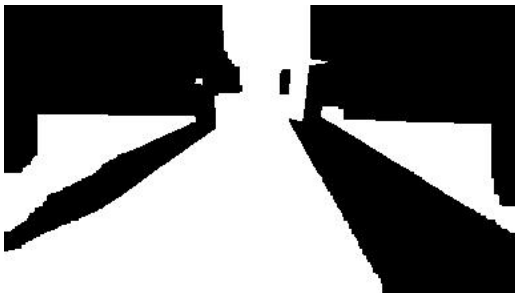

(c) Base image.

Figure 4.2: Results from two types of shadow segmentation. Subfigure **(a)** shows the shadow segmentation performed by the thresholding algorithm, while **(b)** shows the shadow segmentation performed by our graph cut algorithm.


(a) Thresholding.


(b) Graph cut.

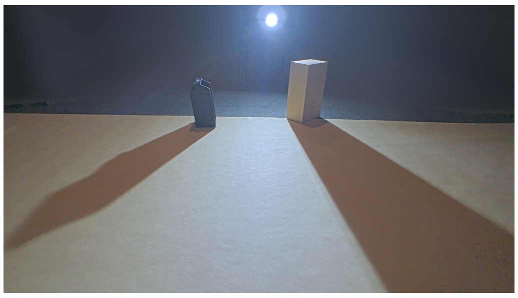

(c) Base image.

Figure 4.3: Results from two types of shadow segmentation. Subfigure **(a)** shows the shadow segmentation performed by the thresholding algorithm, while **(b)** shows the shadow segmentation performed by our graph cut algorithm.

4.3 Comparisons

In this section, we present the mean bounding box overlap (Jaccard index), mean difference in score as well as the PSNR value for the virtual car in 7 different situations, compared to a video of the real car in the same situation. Several different settings are used for both shaders in every situation. Note that parts of the background have been blurred for all images of the scenes, in order to preserve confidentiality.

Before looking at observations from individual situations, it would be best to summarize a few thing that stayed consistent between them. When post-processing was used to blur the virtual car, it was almost never detected by the ANN, and as such, was assigned a score of 0. However, the blurred car always achieved a higher PSNR value than the cars with the regular setting. The PSNR value was always higher for the advanced shader, compared to when the basic shader was used in the corresponding situation. Also, the PSNR value was almost always higher when a virtual cast shadow was rendered.

Regarding the human comparison, it was easier for us to see the virtual car in a scene compared to the real car. The virtual car always stayed separate from the background, while the real car sometimes merged with it, either in shadows or close to the light source. The virtual cast shadow was necessary for the car to look realistic for a human observer, as well as removing double shadows and rendering object shadows. The advanced shader also made the car look more realistic. Therefore, the regular advanced shader is deemed the best by human comparison. Regarding blur, there were both situations where the blur made the virtual car look less realistic and ones where it made it look more realistic. In general, the real car was less sharp than the virtual car with the regular setting, but significantly more sharp than the virtual car with the blur setting. That is, the virtual car generally looked more similar to the real car without blur. However, in the aforementioned cases where the real car merged with the background, it also became less focused. In those instances, the blurred virtual car looked considerably more realistic than the one without blur.

4.3.1 First scene

The first scene used in the tests can be seen in Figure 4.4. Three videos were filmed with this background. One where the car moved in a straight line from the left part of the scene to right, one where it moved from the back to the front, and the last video showed the car moving from the front to the scene to the back.

The comparison between the virtual car and the real car in the case where the car moved from left to right is shown in Table 4.1. The mean Jaccard index was approximately the same for both shaders when no virtual shadow was rendered. The regular advanced shader had a somewhat lower mean Jaccard index compared to the regular basic shader. In general, the mean score difference was lower when using the basic shader. In the ground truth video, the car became brighter when it drove through the middle of the scene, and darker closer to the edges.

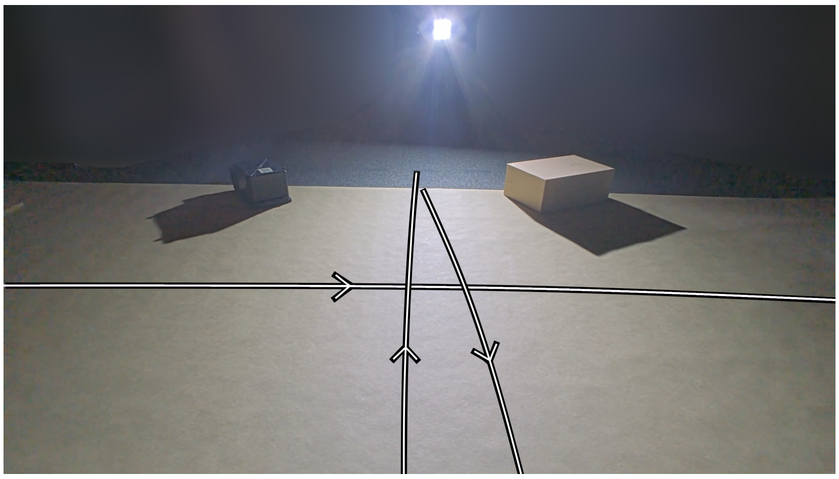


Figure 4.4: A scene with a visible light source and two objects that cast shadows. The scene was used to produce the results in Tables 4.1, 4.2, 4.3. The arrows show the paths of the car in the three videos.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3851	46.3774	15.8911
No virtual shadow	0.3774	49.6005	12.7088
Blur	—	26.9390	16.2882
Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3490	57.7820	19.0392
No virtual shadow	0.3797	53.9259	13.9505
Blur	—	26.9390	19.4050

Table 4.1: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.4, with the cars moving from left to right.

The comparison between the virtual car and the real car in the case where the car moved from the back to the front is shown in Table 4.2. The mean bounding box overlap was generally similar between the basic shader and the advanced shader, with generally low values and the advanced shader having the highest value. The score difference was the lowest when using the advanced shader with a virtual shadow and no post-processing.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.0998	26.2049	13.9402
No virtual shadow	0.1036	25.8179	11.4129
Blur	—	41.1384	14.2422
Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.0922	23.9682	15.6267
No virtual shadow	0.1499	25.4621	12.2745
Blur	—	41.5899	15.7907

Table 4.2: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.4, with the cars moving from the back of the image to the front.

The comparison between the virtual car and the real car in the case where the car moved from the front to the back is shown in Table 4.3. In general, the basic shader had a lower mean Jaccard index and higher score difference compared to the advanced shader.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.2469	21.7081	14.2052
No virtual shadow	0.2403	22.5571	11.3792
Blur	—	29.9677	14.4439
Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3798	14.2396	15.5731
No virtual shadow	0.3378	11.5926	12.0436
Blur	—	29.9677	15.7686

Table 4.3: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.4, with the cars moving from the front of the image to the back.

4.3.2 Second scene

The next scene is shown in Figure 4.5. The car moved from the left side of the scene to the right side in a straight line, and the evaluation parameters are shown in Table 4.4. The score difference was the lowest when the car was blurred. However, do note that the ANN still did not detect the car at all when blurred, so the score of the blurred virtual car was always 0.

The third lowest score difference was found when using the advanced shader without removing double shadows. Generally the mean Jaccard index was about the same for the advanced shader. The highest mean Jaccard index was found with the regular basic shader. The PSNR value was slightly higher when object shadows were not used, compared to the regular setting. When no double shadows were removed, the PSNR value was a bit lower.

Something worth to note from the human comparison is that the real car almost seems to merge with the shadows at times, which is something neither of the virtual cars can imitate. This problem is shown in Figure 4.6. Even when the virtual cars are in a dark shadow, it is easy to see them for a human eye.

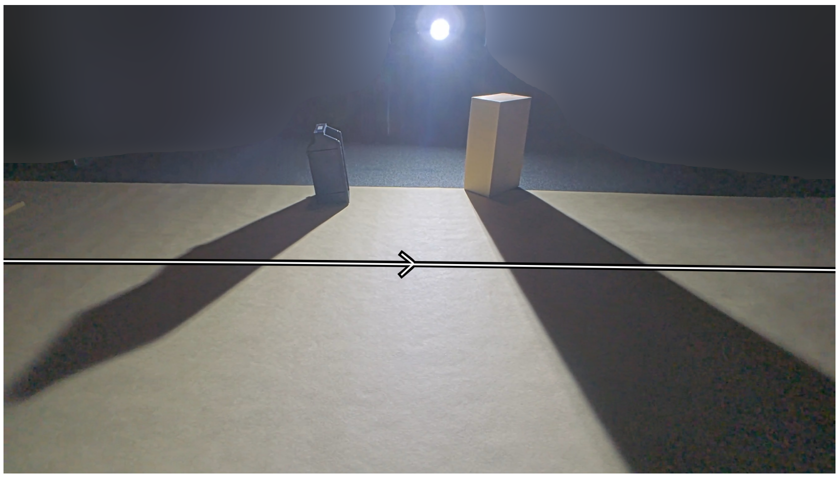


Figure 4.5: A scene with a visible light source and two objects that cast long shadows. The scene was used to produce the results in Table 4.4. The arrow shows the path of the car.

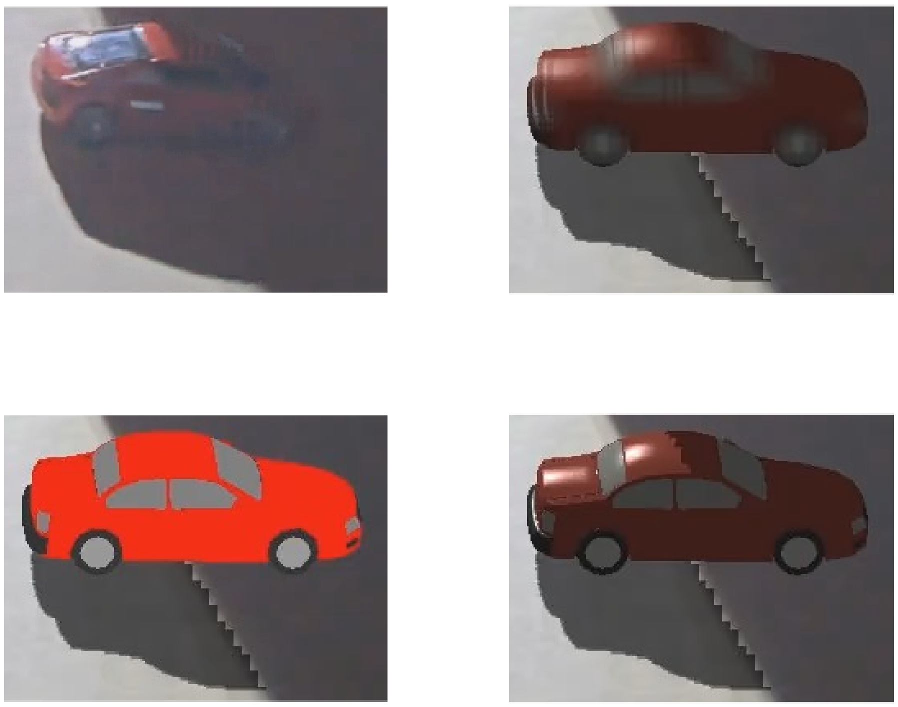


Figure 4.6: Four different cars in the same situation. From top left to bottom right, row-wise: real car, advanced virtual car with blur, basic virtual car, advanced virtual car. Note that the real car merges with the shadow while the virtual cars are easy to separate from the shadow for a human eye.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3858	54.0341	13.5639
No virtual shadow	0.3011	52.5120	11.9768
Blur	—	27.1111	14.0029
Double shadow	0.2561	44.4768	13.0534

Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3744	56.6732	17.5591
No virtual shadow	0.3774	57.7544	14.3280
Blur	—	27.1111	17.9657
Double shadow	0.3712	46.9793	16.3690
No object shadow	0.3739	59.8626	17.7048

Table 4.4: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.5, with the cars moving from left to right.

4.3.3 Third scene

The next scene is shown in Figure 4.7 and is similar to the previous one in Figure 4.5, but with a higher intensity of ambient light. The car also moved from left to right and the evaluation data is shown in Table 4.5. Here we can see that the mean Jaccard index was the highest with the basic shader without a virtual cast shadow. The mean Jaccard index is also higher for the virtual car in this scene, compared to the scene in Figure 4.5. The advanced shader had a higher PSNR value when double shadows and real shadows were taken into account.

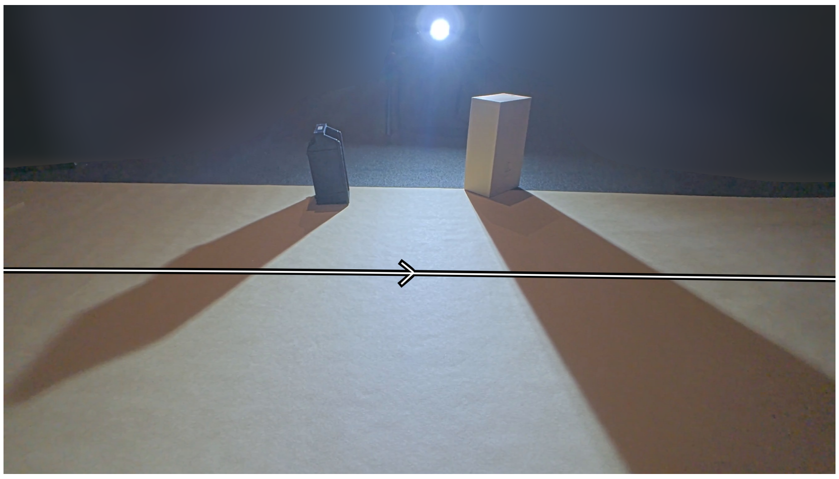


Figure 4.7: A scene with a visible light source and two objects that cast long shadows that are brighter than the ones in Figure 4.5. The scene was used to produce the results in Table 4.5. The arrow shows the path of the car.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.5671	12.5042	14.3377
No virtual shadow	0.5697	14.4897	12.8407
Blur	—	76.1615	14.8579
Double shadow	0.2546	19.1760	13.9233
Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.3916	11.7645	19.3373
No virtual shadow	0.3955	11.6870	15.7127
Blur	—	76.1615	20.0789
Double shadow	0.4976	16.6839	18.1465
No object shadow	0.4045	11.3184	19.3361

Table 4.5: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.7, with the cars moving from left to right.

For this case we elect to show a plot of the score over time, for the advanced virtual car and the real car, see Figure 4.8. More such plots can be found in Appendix B.

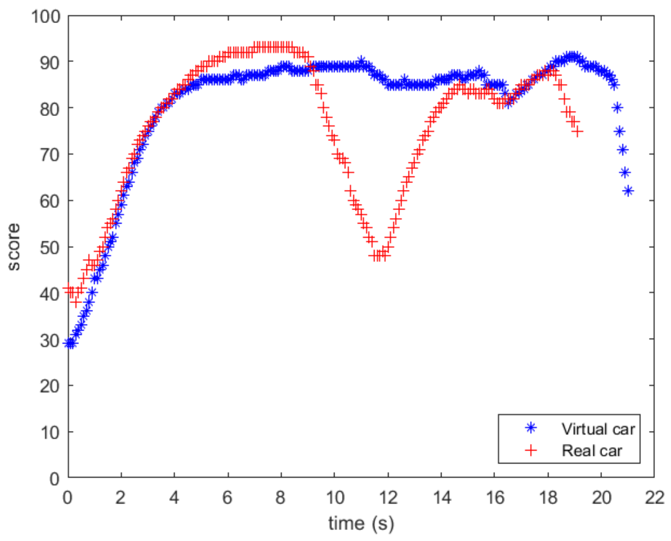


Figure 4.8: Scores on a scale from 0 to 100, for the virtual and the real car.

4.3.4 Fourth scene

In this next scene the light source was placed so that it was not visible from the camera's perspective. The scene can be seen in Figure 4.9. In this situation, the car moved from the left part of the scene to the right, and the evaluation data is displayed in Table 4.6. The mean Jaccard index was quite low for the virtual car, with the exception of the case with the regular advanced shader. The mean score difference was similar between the basic and advanced shader when no post-processing was used.

In the video of the real car, the ANN had a few observations that deviated from the rest (see Appendix B, Figures B.11b and B.12b). These occurred between 9 and 12 seconds from the first observation of the car. These observations were kept since their bounding boxes were positioned similarly to the rest, which suggested that they were observations of the car rather than something in the background. The score also decreases occasionally during this time frame when looking at the results from the virtual car with the advanced shader and basic shader with the regular setting.

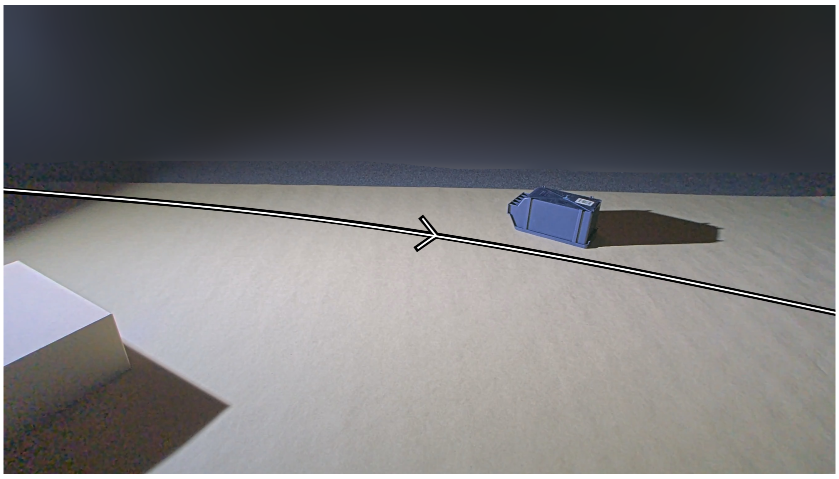


Figure 4.9: A scene with a light source outside the image, to the left, and two objects that cast shadows. The scene was used to produce the results in Table 4.6. The arrow shows the path of the car.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.0441	22.7065	12.3185
No virtual shadow	0.0616	21.4721	12.3904
Blur	—	68.0241	12.7498
Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.1866	22.3403	13.6904
No virtual shadow	0.0594	24.6612	13.7941
Blur	0	55.0890	14.1603

Table 4.6: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.9, with the cars moving from left to right.

4.3.5 Fifth scene

The last scene used is found in Figure 4.10. For this situation, the car moved from the bottom left corner of the scene to the upper right. The mean Jaccard index was approximately the same when using the basic and advanced shader, and in both cases somewhat lower when no virtual cast shadow was rendered. The mean score difference was generally somewhat higher for the advanced shader compared to the basic shader, and considerably higher when post-processing was used.

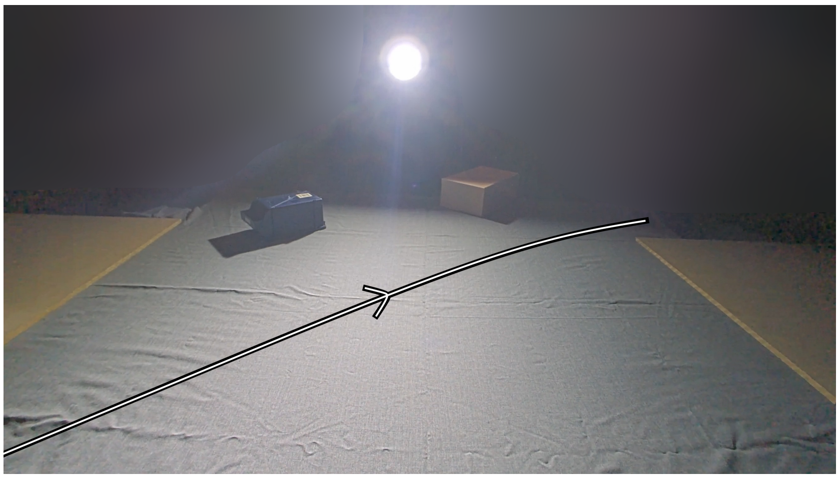


Figure 4.10: A scene with a visible light source and two objects that cast shadows. The scene was used to produce the results in Table 4.7. The arrow shows the path of the car.

Basic shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.4865	19.4164	14.0010
No virtual shadow	0.4779	20.2031	11.9958
Blur	—	54.1461	14.3587

Advanced shader	**Mean Jaccard index**	**Mean score difference**	**Image PSNR**
Regular	0.4819	25.1981	15.9602
No virtual shadow	0.4775	23.3894	13.1186
Blur	—	54.1461	16.1846

Table 4.7: Virtual car with two different types of shaders, compared to statistics from a video of a real car. The table was made using the scene shown in Figure 4.10, with the cars moving from diagonally across the scene from the lower left to the upper right.

Chapter 5
Discussion

In this section, the results are analyzed and compared.

5.1 The virtual shadows

The inclusion of the virtual cast shadows increased the PSNR value of the virtual car in almost all cases, with the exception of the videos filmed in the scene shown in Figure 4.9. It is expected that the PSNR would increase with a cast shadow, since more pixels of the background would be more similar. A reason for why the scene shown in Figure 4.9 instead gave a lower PSNR value could be the result of poor alignment of the virtual car to the ground truth. This meant that the virtual shadow covered areas of the image where the ground truth had no shadow, therefore lowering the value. Moreover, the virtual shadow made the car look more integrated with the scene for a human, and therefore more realistic. Regarding the output from the ANN, the virtual cast shadow does not seem to impact the ANN in a significant way. For both the advanced shader and the basic shader, there were cases when the virtual car had a higher Jaccard index and lower score difference with a virtual shadow, as well as cases where the Jaccard index decreased and the score difference increased. For instance, in Table 4.1 the output values from the ANN are less similar to the ground truth for the advanced shader, and in Table 4.6 the opposite was true. A reason for this could be because the ANN is mainly trained on detecting the car itself and that other details in the background, such as a cast shadow, do not contribute as much to the end result. Moreover, the vehicle detection ANN is likely also trained on images where a cast shadow from the car is not visible by the camera, such as when the main light source is behind the camera. Therefore the ANN should probably be able to detect a real car, regardless of if a cast shadow is present or not, which might explain our results.

Moving on to the impact of the double shadows, the sample size is decreased to a modest two scenes, resulting in four samples. Thus, it is difficult to draw reliable conclusions. In Table

4.4, including a double shadow decreases the mean score difference for both the basic and the advanced shader. However, in Table 4.5, both mean score differences are instead increased. For all cases, the impact of the double shadows is relatively large. The PSNR, on the other hand, seems to have less significance, but at the same time more confirmed direction. Not removing double shadows always decreases the PSNR value. This is fully logical, since a double shadow will darken a part of the image that would otherwise remain unchanged between the virtual and real video. Of course, this assumes that the shadow segmentation has been performed accurately. The Jaccard index is lowered for the basic shader but unchanged for the advanced shader. This could potentially hint at the ANN mistaking the double shadow for a part of the car when the simple shader was used, which led to a misplacement of the bounding box.

The object shadow has even less samples, only two. One case slightly increases the mean score difference for the advanced car and one slightly decreases it. However, both of these changes are small enough that they could likely have originated from misalignment when matching the videos, and as such no conclusion would have been possible to draw even had both either increased or decreased. Regarding the PSNR, the changes are small but not always positive. The problem in the scene shown in Figure 4.5 is that the real car melted into the shadow while the virtual car with the advanced shader became too dark, darker than the shadow itself. This can be seen in the top left and bottom right part of Figure 4.6. Without the object shadow, the part of the virtual car that was in the shadow became brighter and matched the real car slightly better. This only led to a big difference in Table 4.4, not in Table 4.5, since the shadows were brighter in the latter and the real car did not melt into them. The Jaccard index does not seem impacted in any way by the inclusion or exclusion of the object shadow.

5.2 Advanced and basic shader comparison

It is clear from the results of the pixelwise comparison that the advanced shader makes the virtual car look more similar to the real car than the basic shader. This is easily seen in the 'Image PSNR' column, in which the advanced shader always has a higher value than the corresponding field for the basic shader. However, the ANN comparison is more varied. There is not a clear pattern regarding which shader performs the best. For instance, in Table 4.1, the advanced shader generally has a higher mean score difference than the basic shader (worse performance), but in Table 4.3, the opposite is true.

The mean Jaccard index is also not trivial to interpret. In many cases, it is too low to make a meaningful comparison, especially since the Jaccard index is sensitive to how well the two compared videos are synced to each other. This makes it somewhat unreliable.

5.3 Real and virtual car comparison

As stated in the results, the human comparison made it clear that there were some situations where the virtual car was much more visible than the real car. We believe this is what created the large gap in score between the virtual and the real car. There are many reasons for why this could be the case. One reason could be that the image of the scene and our assumptions were not enough to accurately capture the necessary lighting parameters. As mentioned before in Section 1.5, obtaining the light information from a single image is an ill-posed problem, meaning that more information is needed to get a solution. It is therefore possible that our assumptions about the scene were not enough to get an accurate result. Furthermore, the virtual car remained fairly static in color for the duration of the video, unless it entered into areas that were marked as shadow. In the ground truth videos, the car did not always look the same in all positions of the scene. For example, in scenes with a visible light source the car became brighter when it was positioned in the middle of the scene, compared to when it was placed closer to the edges. This could have been one of the reasons for why the virtual car differed in score in comparison to the ground truth. For instance, in Figure 4.8, the real car gets a lower score after about 10 seconds, which is likely when the car is closest to the light source in the scene in Figure 4.7.

5.4 Blur

The impact of blur as a post-processing effect on the results was a great one. On one hand, the PSNR was always higher when using blur compared to other options for the same shader type. On the other hand, the ANN was unable to detect the blurred car, both when using the basic shader and the advanced shader. This is obviously not a favorable result. Of course, being able to make the virtual car look visually similar to real car is an important step, but the main focus was on making the ANN treat the virtual car the same way as it did the real one. It is difficult to know how far the blurred car is from actually being detected, but by reducing the amount of blur one should reasonably be able to find a good balance where the virtual car is detected by the ANN, but does not achieve as high score as the ones without any blur.

A behaviour that the virtual car seems to be unable to recreate is how the real car seems to almost blend into the background when entering shadow areas in low light settings. An example of this can be seen in Figure 4.6. The virtual car with the advanced shader became too dark when entering the shadow, and different parts of the car such as headlights and wheels remained detailed. In the ground truth, it almost looks as if the car is entering fog. The virtual car with the advanced shader and post-processing overall looks too blurred, but in the shadows the blurred effect is closer to the ground truth.

5.5 Future improvements

There are many things that could be done to improve the results and performance described in the earlier sections. Listed here are a few suggestions.

5.5.1 The virtual car model

The virtual 3D-model of the car and the real model car are similar, but not copies of each other. The virtual car is less detailed and does not include some elements of the real car, such as the light gray plates on the sides of the car and the dark color of the trunk cover. These differences could have affected the result, especially for the pixelwise evaluation. For future work, using a more similar 3D-model or a 3D-scan of the real car will likely give a more reliable result.

5.5.2 More advanced blur

Our results suggest that blur can greatly affect how easily the ANN can detect the car. By using a more advanced version of blur, we expect better results could be achieved. The first option is to simply try using varying amounts of blur and investigate its impact. Instead of blurring the entire car, another option is to only blur the parts of the car that are in its self-shadow, because those are the darkest and therefore hardest for a camera to see clearly. A third option is to blur the car together with the parts of the background closest to it, in order to merge the car with the background.

5.5.3 Scene variation

When producing the result, only 7 different situations were used, and three of the videos were filmed in the same scene. This is a very limited data set, and it is therefore difficult to determine what the results would be from a video taken in a new scene. The scenes in the data set are also quite simple and similar visually. In the last scene, shown in Figure 4.10, a different ground material was used but otherwise it had a lot of the same elements as the other scenes. The main reason for why so few situations were used was because of time constraints and lack of resources.

5.5.4 Gradient shadows

One problem that the graph cuts struggle with is when the scene includes gradient shadows. A gradient shadow appears when a shadow on a surface gradually turns to light along the surface, rather than having a distinct edge. The graph cut algorithm is binary, meaning it can only classify a pixel as either non-shadow or shadow, nothing in between. There are methods for obtaining a soft shadow segmentation, where the amount of shadow is indicated by a number between 0 and 1, 0 being no shadow and 1 being full shadow. One such method has been made by Madsen and Nielsen [10]. Using that would result in a better representation of the lighting in the scene, and could lead to more accurate results when rendering a shadow for the virtual car.

5.5.5 Working with a real car

As ground truth data, it would of course be more realistic to use a real car rather than a model car, but it would also be more complicated. A larger testing ground is required, and bigger equipment for controlling the lighting. Filming could only be performed when it is dark outside and when the weather permits, which depending on location and season can be more or less of a problem.

By using a real car, another possibility connected to the lighting is enabled, namely using the headlights of the car to create difficult lighting situations for the camera. It is known that a camera can be blinded if the car's headlights are aimed straight at it, which is a situation that is entirely possible, to not say unavoidable, when driving a car at the same level as the camera. Of course, such a thing would in theory be possible even with a model car, but it would most likely be difficult to find one that has strong enough lights to create the same effect.

To liken the virtual car to a real car with headlights, virtual spotlights could be added to the front of the virtual car and used to create an artificial flare effect. However, it must be noted that the virtual headlights need to illuminate elements in the real scene, such as the floor and potential objects and walls in front of car. Depending on what accuracy is sought, this could require a 3D model of the scene to solve correctly.

5.5.6 Double shadows

The method for avoiding double shadows was to use the shadow segmentation as a binary mask which showed where a virtual shadow could be rendered. This worked best when the whole rendered shadow was either surrounded by shadow pixels or non-shadow pixels. But when the virtual shadow is only partially covered by a real shadow, some double shadows occur on the edge between them. There are also some pixels on the edge that become too bright. Examples of this can be seen for the virtual cars in Figure 4.6.

A possible alternative way of solving this problem would have been to use shadow removal on the scene to make the transition between real and virtual shadows more smooth. By first finding and removing all shadows in the scene and saving the result as a separate texture, the new shadows would have been rendered by using the pixel values from this texture multiplied with the shadow ratio to hopefully give a smoother result.

5.5.7 More advanced shaders

There are several ways in which the shaders could be improved to make the virtual car look more realistic, which would probably yield better results. One is to use normal maps, specular maps and occlusion maps to easily add more details to an otherwise plain car. Phong shading has its limits. Another way is using physically based rendering (PBR), which aims to achieve photorealism and model the flow of light as in the real world. However, this could potentially be too taxing for a real-time application, especially if it is rendered on a camera with limited resources.

5.5.8 Depth detection

One of the requirements in this project was to use only a single video frame for the scene analysis, which removes several options that require a 3D model of the scene. However, one possible solution would be to utilize an algorithm that can determine depth in a single image, and from that construct a 3D model. Such an algorithm would most likely be based on AI. Since the image only shows the scene from one direction, the 3D model would be limited to just depth from that point of view, but that would still open up a lot of new possibilities. Some such possibilities are described in Section 1.5.

Chapter 6
Conclusions

In this thesis, we have investigated how to analyze an image taken in a low-light setting to acquire information about the lighting conditions, and how this can be used to improve the appearance of a virtual car. We have developed a shadow segmentation algorithm in order to extract color information from the shadows and non-shadows of the scene. The evaluation has been done by comparing a virtual car to a real model car, both visually and with PSNR, as well as by comparing output parameters from a vehicle-detecting ANN. Two different shaders were used to determine the appearance of the virtual car. One basic shader, which rendered the car without taking any light or shadow into account, and one advanced shader, which used information from the shadow segmentation. Both shaders were tested with some parts of the rendering varying. This was to see what effect for instance virtual shadows and blur had.

The following research questions were asked and answered in this project:

Will the output from the ANN be more similar to the ground truth when using the advanced shader, compared to the basic shader? No, not in any clear way.

Will the PSNR value between the ground truth and the advanced shader be higher, compared to the PSNR between the ground truth and the basic shader? Yes, the PSNR was always higher when using the advanced shader compared to the basic shader.

How do the virtual cast shadows and blur impact the PSNR value and output from the ANN of the virtual car compared with the ground truth? Virtual cast shadows generally increased the PSNR value and made the car look more realistic, but did not change the output from the ANN in any relevant way. Blur also increased the PSNR, but made it almost impossible for the ANN to classify the virtual car as a car.

Does the advanced shader make the virtual car look more similar to the ground truth, from the perspective of a human observer? Yes. The car looks much better and more similar to the ground truth when using the advanced shader than when using the basic shader. However, there are still situations where the virtual car differs noticeably from the ground truth.

In summary, the advanced shader made the car visually more similar to the real car compared to the basic shader, but from the perspective of the ANN the advanced shader did not make the car more realistic. Therefore it is not suitable to use videos of the virtual car with the advanced shader as validation data for the ANN. However, several different approaches have been identified that might lead to more desirable results in future research.

References

[1] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering, Fourth Edition*. Crc Press, fourth edition, 2019.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[3] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[4] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[5] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[6] Paul Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bull Soc Vaudoise Sci Nat*, 37:241–272, 1901.

[7] Katrien Jacobs, Cameron Angus, Celine Loscos, Jean-Daniel Nahmias, Alex Reche, and Anthony Steed. Automatic generation of consistent shadows for augmented reality. In *Proceedings of Graphics Interface (GI 2005)*, pages 113–120. Canadian Human-Computer Communications Society and School of Computer …, 2005.

[8] Salma Jiddi, Philippe Robert, and Eric Marchand. Estimation of position and intensity of dynamic light sources using cast shadows on textured real surfaces. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1063–1067. IEEE, 2018.

[9] Martin Loesdau, Sébastien Chabrier, and Alban Gabillon. Hue and saturation in the rgb color space. In *Image and Signal Processing: 6th International Conference, ICISP 2014, Cherbourg, France, June 30–July 2, 2014. Proceedings 6*, pages 203–212. Springer, 2014.

[10] Claus B Madsen and Michael Nielsen. Towards probe-less augmented reality: a position paper. In *Proceedings: GRAPP 2008*, pages 255–261. Institute for Systems and Technologies of Information, Control and Communication, 2008.

[11] Jacob Munkberg. Shading and glsl - edaf80. `https://fileadmin.cs.lth.se/cs/Education/EDA221/lectures/Lecture4_web.pdf`, 2012.

[12] Saritha Murali and VK Govindan. Shadow detection and removal from a single image using lab color space. *Cybernetics and information technologies*, 13(1):95–103, 2013.

[13] Manasa Nadipally. Chapter 2 - optimization of methods for image-texture segmentation using ant colony optimization. In *Intelligent data analysis for biomedical applications*, pages 21–47. Elsevier, 2019.

[14] George Paschos. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. *IEEE transactions on Image Processing*, 10(6):932–937, 2001.

[15] Ioannis Pitas. *Digital image processing algorithms and applications.* John Wiley & Sons, 2000.

[16] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[17] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A standard default color space for the internet-srgb, version 1.10. 1996. `http://www.w3.org/Graphics/Color/sRGB`. [see matrix at end of Part 2].

[18] Taffee T Tanimoto. *Elementary mathematical theory of classification and prediction.* International Business Machines Corp., 1958.

[19] TankStorm. Virtual car model. `https://www.cgtrader.com/products/lowpolycar-free-sample`. [Online; accessed 14-February-2023].

[20] Joey de Vries. Ortographic and perspective projection — Learn OpenGL. `https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping`, 2015. [Online; accessed 11-May-2023] `https://twitter.com/JoeyDeVriez`.

[21] John A Watlington. Gray world assumptions. `http://alumni.media.mit.edu/~wad/color/exp1/newgray/`, 1996. [Online; accessed 12-May-2023].

[22] Wikipedia. Phong reflection model — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Phong_reflection_model`, 2023. [Online; accessed 11-May-2023].

[23] Pengfeng Xiao, Min Yuan, Xueliang Zhang, Xuezhi Feng, and Yanwen Guo. Cosegmentation for object-based building change detection from high-resolution remotely sensed images. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–17, 01 2017.

Appendices

Appendix A

Screenshots from the ground truth

The following figures aim to visualize the movement of the real car in the seven videos used as ground truth. Each figure has four subfigures, which show the car at different points in each video. The order in time from first to last is top right, top left, bottom left, bottom right. Note that parts of the backgrounds have been blurred to preserve confidentiality.

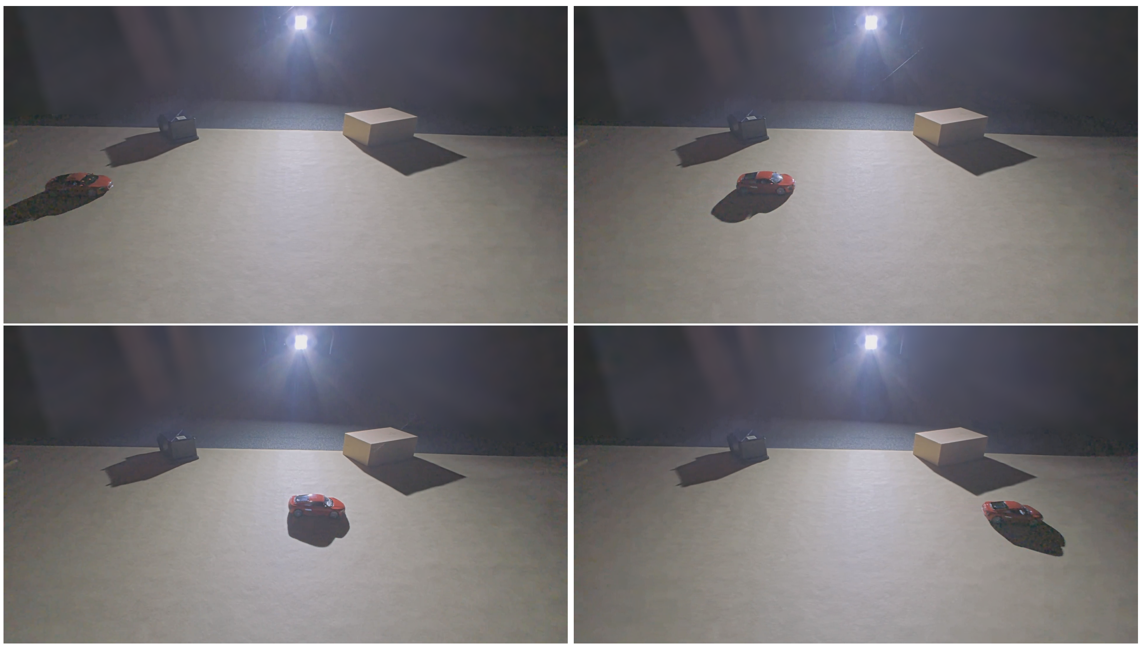


Figure A.1: Screenshots from the video with the background shown in Figure 4.4. The car follows the path of the arrow pointing to the right.

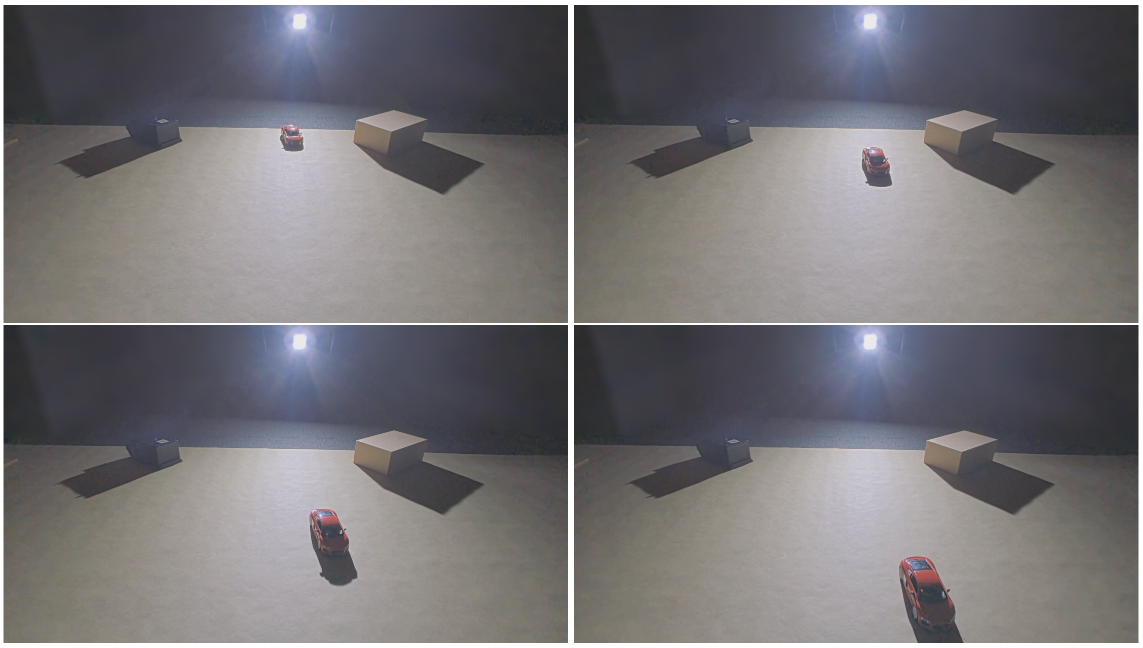


Figure A.2: Screenshots from the video with the background shown in Figure 4.4. The car follows the path of the arrow pointing to the bottom of the image.

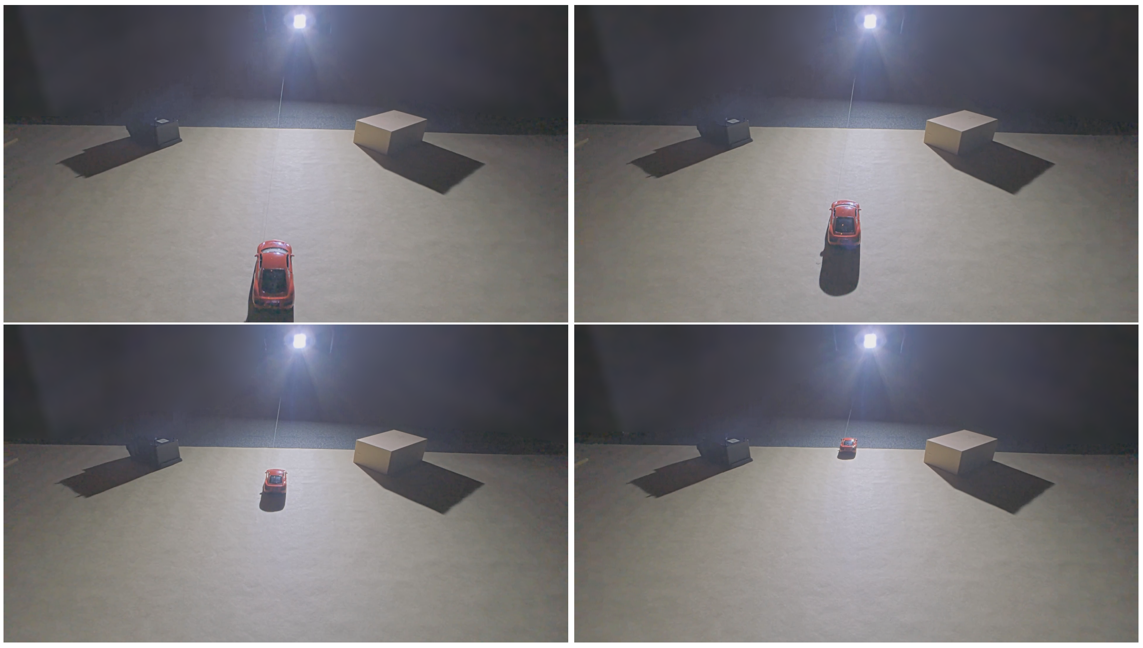


Figure A.3: Screenshots from the video with the background shown in Figure 4.4. The car follows the path of the arrow pointing to the top of the image.

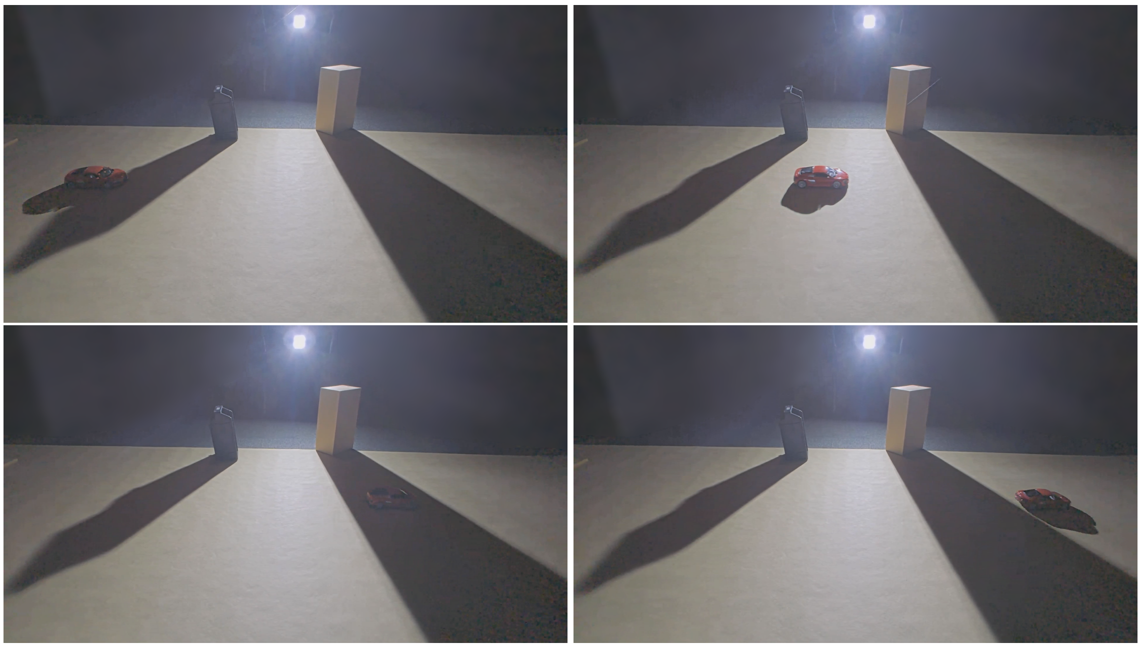


Figure A.4: Screenshots from the video with the background shown in Figure 4.5.

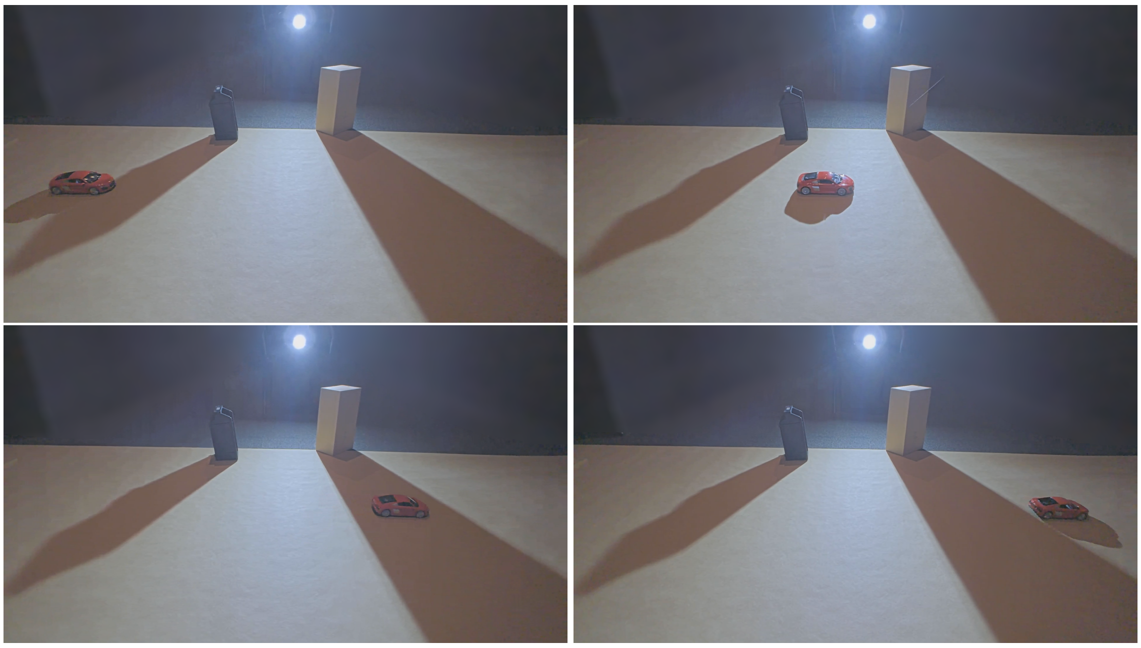


Figure A.5: Screenshots from the video with the background shown in Figure 4.7.

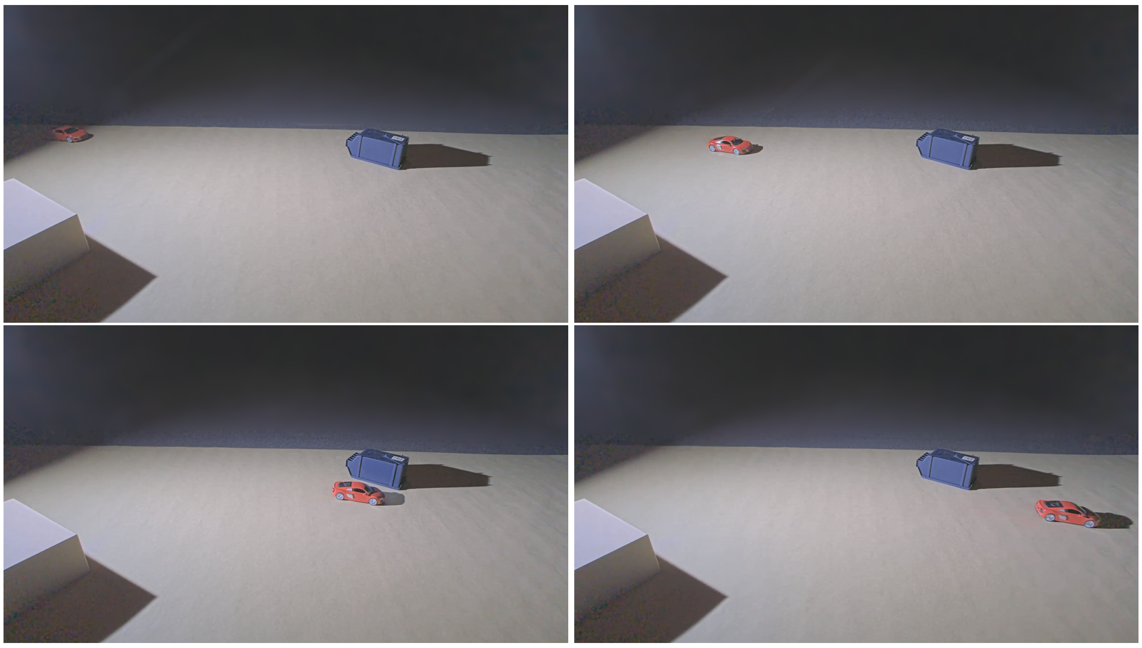


Figure A.6: Screenshots from the video with the background shown in Figure 4.9.

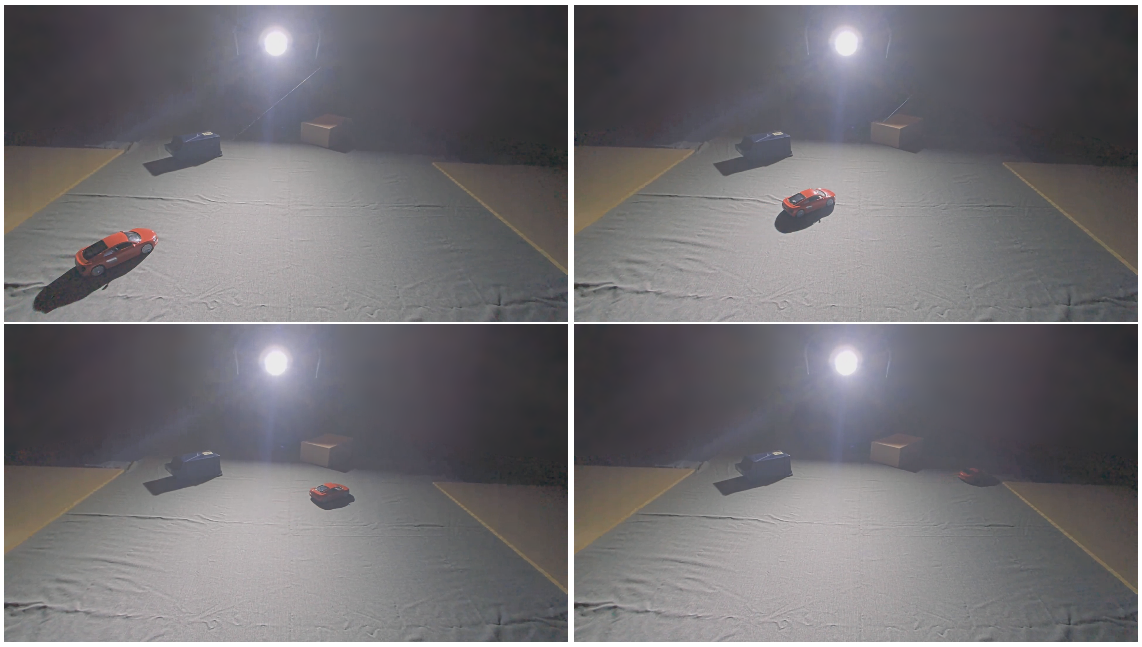
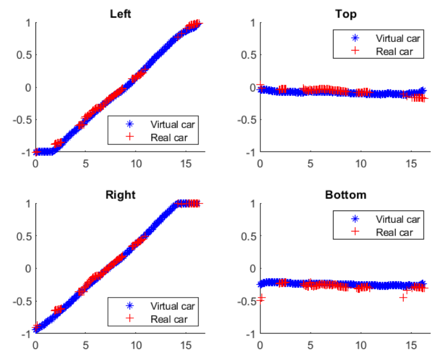


Figure A.7: Screenshots from the video with the background shown in Figure 4.10.

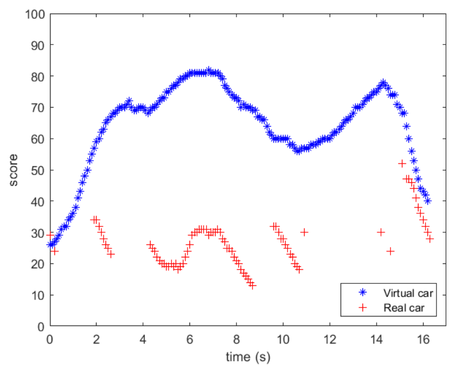
Appendix B

Bounding boxes and scores

The following figures can give more insight into the results. They are structured in pairs that each correspond to a certain situation in the results. The first figure in each pair show the results for the regular basic shader and the second for the regular advanced shader. Both are compared with the real car. Also, each figure has two subfigures that show bounding box coordinates and scores. The bounding box coordinates are shown as four different plots, corresponding to the different boundaries of the box, which are left, top, right and bottom. All coordinates range from -1 to 1, where -1 is the leftmost or lower part of the screen and 1 is the rightmost or upper part of the screen.

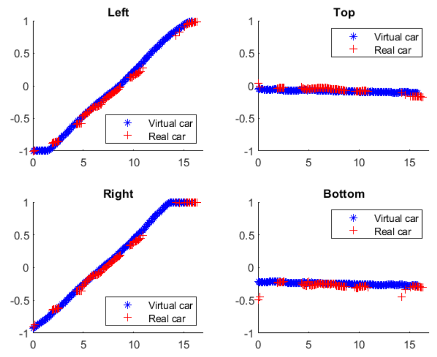


(a) Bounding box.

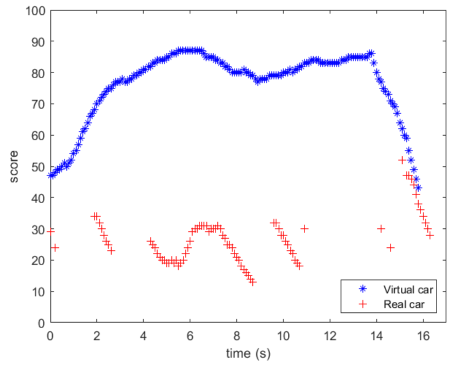


(b) Score.

Figure B.1: Detailed results of the ANN comparison for the regular basic shader in Table 4.1.

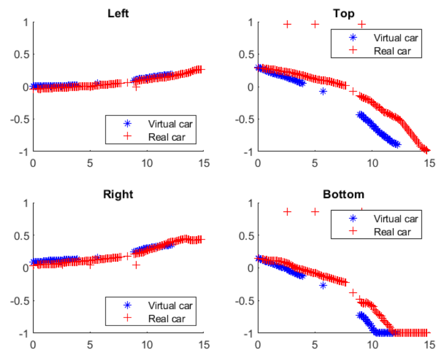


(a) Bounding box.

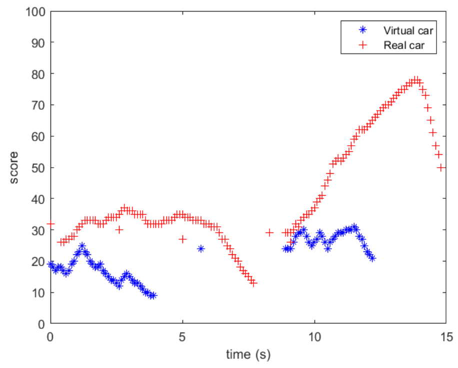


(b) Score.

Figure B.2: Detailed results of the ANN comparison for the regular advanced shader in Table 4.1.

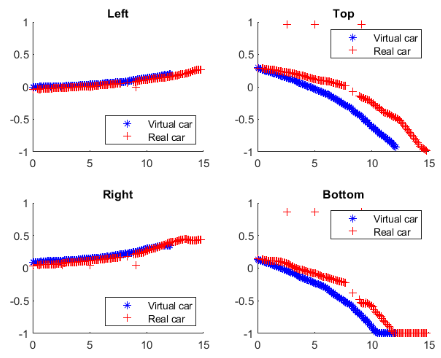


(a) Bounding box.

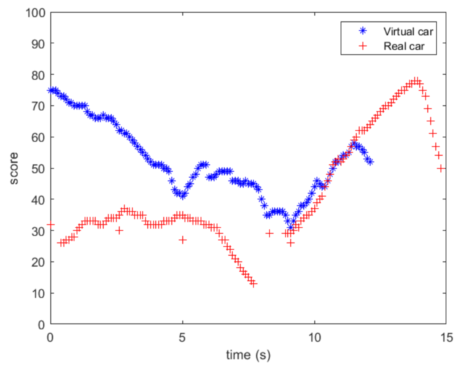


(b) Score.

Figure B.3: Detailed results of the ANN comparison for the regular basic shader in Table 4.2.

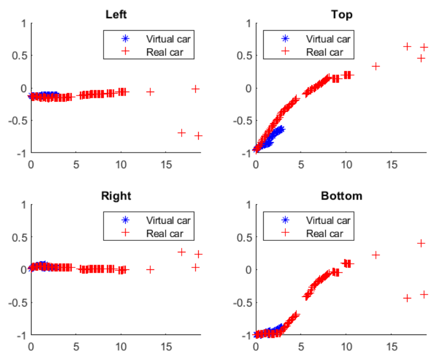


(a) Bounding box.

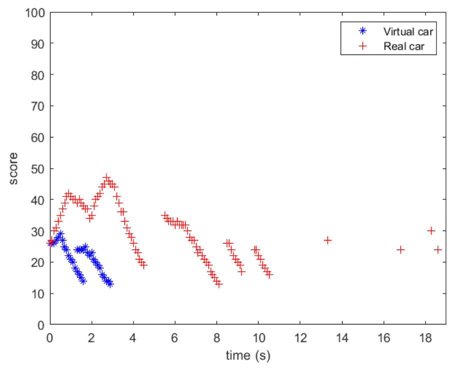


(b) Score.

Figure B.4: Detailed results of the ANN comparison for the regular advanced shader in Table 4.2.

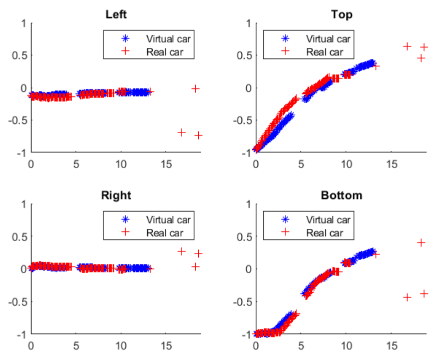


(a) Bounding box.

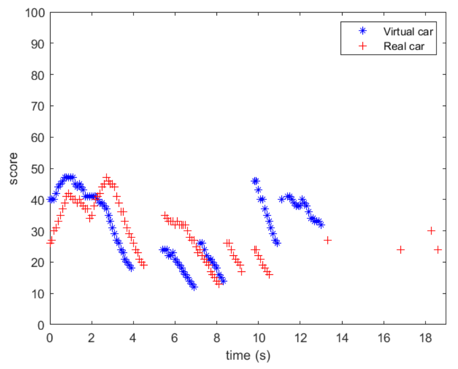


(b) Score.

Figure B.5: Detailed results of the ANN comparison for the regular basic shader in Table 4.3.

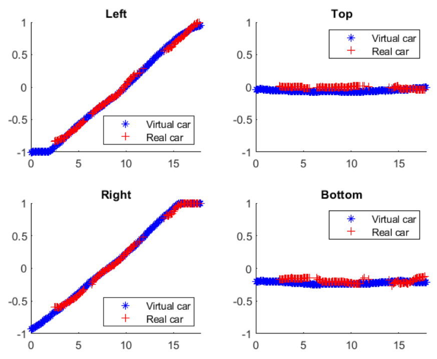


(a) Bounding box.

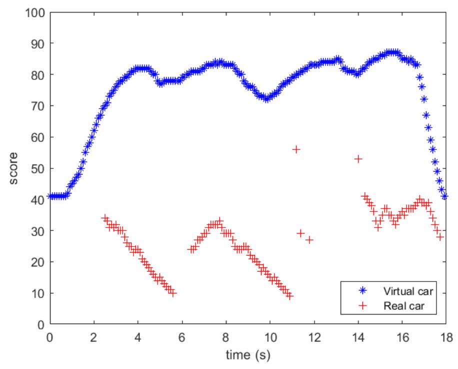


(b) Score.

Figure B.6: Detailed results of the ANN comparison for the regular advanced shader in Table 4.3.

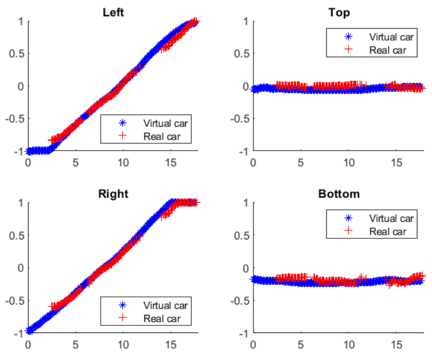


(a) Bounding box.

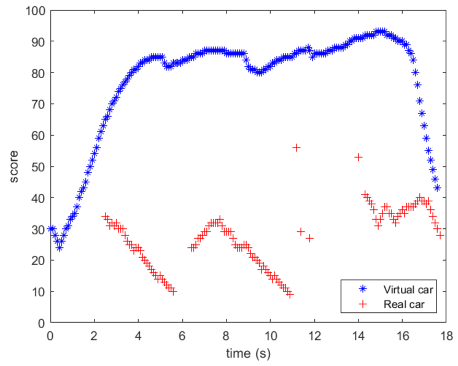


(b) Score.

Figure B.7: Detailed results of the ANN comparison for the regular basic shader in Table 4.4.

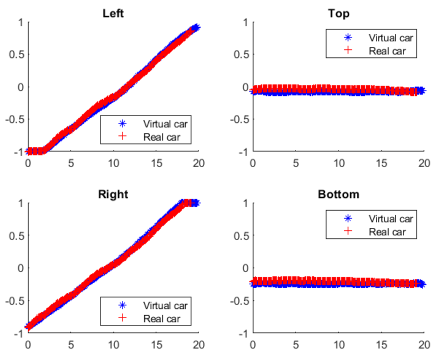


(a) Bounding box.

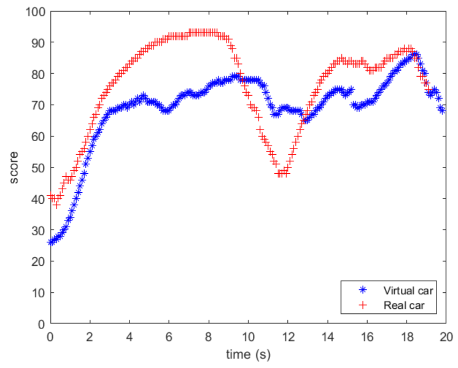


(b) Score.

Figure B.8: Detailed results of the ANN comparison for the regular advanced shader in Table 4.4.

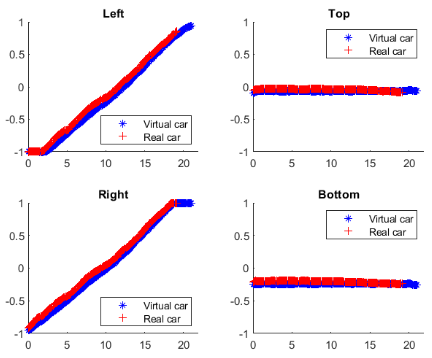


(a) Bounding box.

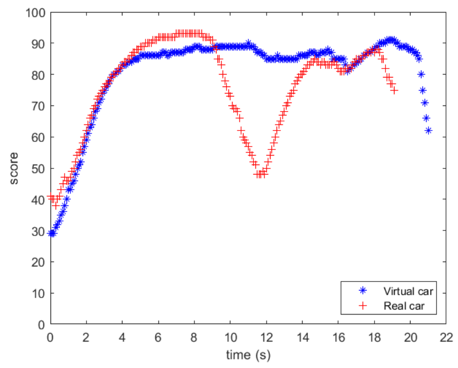


(b) Score.

Figure B.9: Detailed results of the ANN comparison for the regular basic shader in Table 4.5.

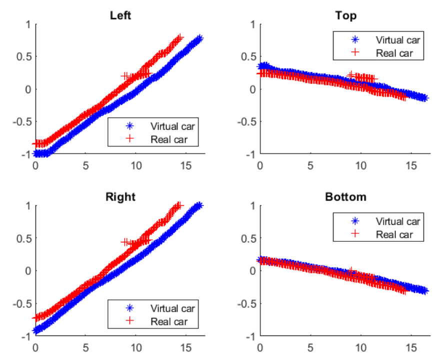


(a) Bounding box.

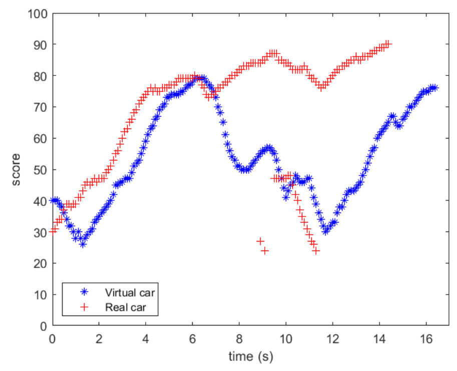


(b) Score.

Figure B.10: Detailed results of the ANN comparison for the regular advanced shader in Table 4.5.

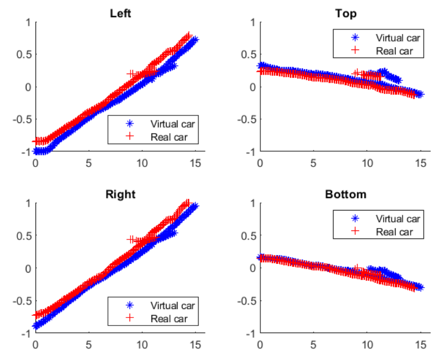


(a) Bounding box.

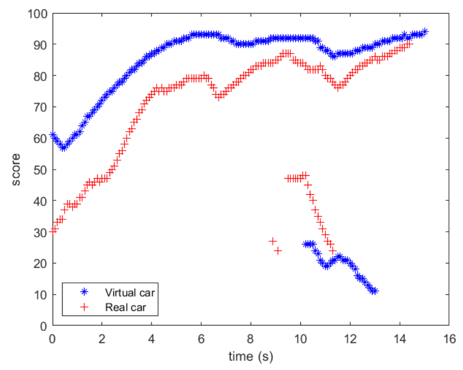


(b) Score.

Figure B.11: Detailed results of the ANN comparison for the regular basic shader in Table 4.6.

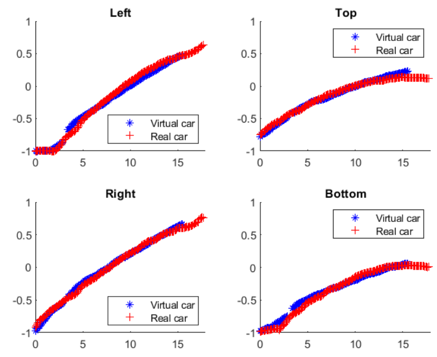


(a) Bounding box.

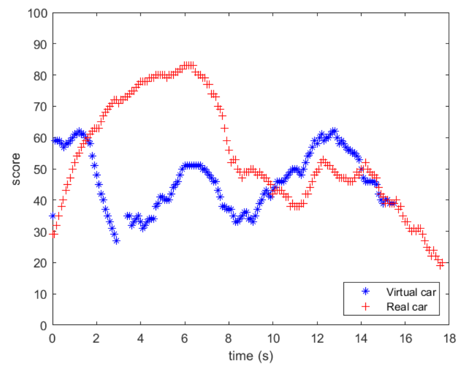


(b) Score.

Figure B.12: Detailed results of the ANN comparison for the regular advanced shader in Table 4.6.

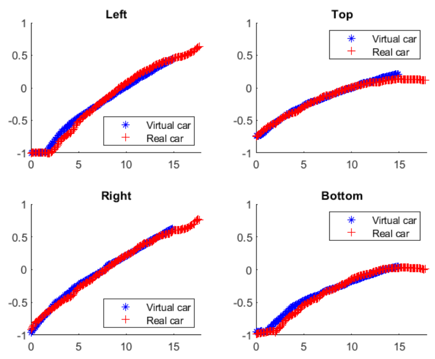


(a) Bounding box.

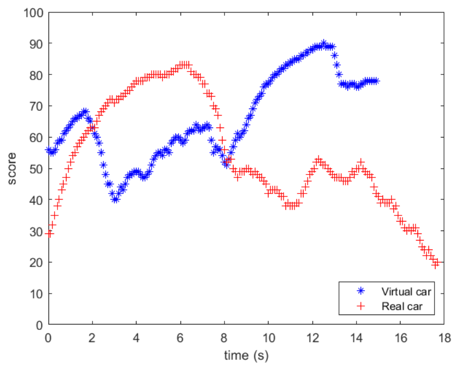


(b) Score.

Figure B.13: Detailed results of the ANN comparison for the regular basic shader in Table 4.7.



(a) Bounding box.



(b) Score.

Figure B.14: Detailed results of the ANN comparison for the regular advanced shader in Table 4.7.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-16

EXAMENSARBETE Improving artificial validation data using scene analysis
STUDENTER Gustav Klotz, Kristina Patrikson
HANDLEDARE Michael Doggett (LTH)
EXAMINATOR Flavius Gruian (LTH)

Realistisk virtuell bil för bättre validering av artificiell intelligens

POPULÄRVETENSKAPLIG SAMMANFATTNING **Gustav Klotz, Kristina Patrikson**

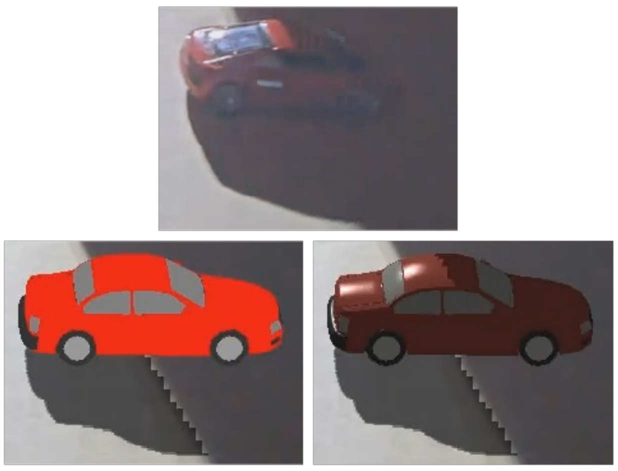
Artificiell intelligens som detekterar olika objekt blir allt vanligare. Vårt arbete gick ut på att göra en virtuell bil så lik en riktig bil som möjligt, speciellt i svåra miljöer med dåligt ljus, så att den skulle kunna användas istället för en riktig bil för att kontrollera att en AI funkar.

Idag används AI till bildanalys inom många olika företag och forskningsområden. För övervakning av trafik eller säkerhetsklassade områden pågår arbete för att konstruera AI som kan detektera och varna säkerhetspersonal när bilar befinner sig i dessa områden. När man tränar en AI behövs mycket data, men efter träningen behöver AI:n också valideras, det vill säga att man kontrollerar att den klarar sig bra på sådant den inte har tränats på. Ett potentiellt sätt att förenkla insamlingen av denna valideringsdata är att låta en virtuell bil köra framför en bild av en riktig bakgrund.

Målet var att förbättra en virtuell bil så att den såg bättre ut i mörka situationer, för annars stack den ut som alldeles för ljus. Den gamla bilen såg likadan ut oavsett bakgrund eller ljus, medan vår nya bil ändrades i utseende. Vi jämförde både den gamla och nya bilen med en riktig bil, för att kunna se om vår nya var bättre. Vi jämförde också både genom att se på hur lika bilarna var varandra, men också genom att se hur AI:n reagerade på de olika bilarna. Om vår nya bil fick liknande reaktioner från AI:n som den riktiga bilen så var det bra.

Vi delade upp bilden i skugga och icke-skugga för att räkna ut hur mörkt det var i skuggorna jämfört med i ljus. Vi kunde även utnyttja uppdelningen för att veta när den virtuella bilen skulle ha skugga på sig.

Resultatet visade att vår nya bil såg bättre ut än den gamla, men tyvärr tyckte AI:n att båda var ganska olika den riktiga bilen. Detta kan bero på att de virtuella bilarna har svårt att smälta in i skuggor, och därmed hittas lättare av AI:n, speciellt i de mörkaste bilderna. Vår virituella bil kan därför inte användas för att validera en AI.


Figur 1: Den riktiga bilen (Övre), gamla virtuella bilen (Nedre vänster) och nya (Nedre höger).