Tampere University

Samuel Schmidt

LIVING IN THE LINE WORLD

Better Lines

Bachelors degree
Faculty of Information Technology and Communication Sciences
Examiner: Prof. Joni Kämäräinen
June 2023

ABSTRACT

Samuel Schmidt: Living in the Line World
Bachelors degree
Tampere University
Degree Programme in Computing and Electrical Engineering, BSc
June 2023

This thesis continues the work of Aleksi Viljanen[1] and Anni Lignell[2], fixing problems Lignell found during her thesis and improving the usability of Viljanens application. The main goal is to create a VR application that allows us to live in the Line World, a world that consists only of the edges of objects drawn with lines, and to figure out what humans can do in such a world.

The main focuses will be to make the view much smoother as currently it can take some time for the view to completely render. This can cause nausea and is just in general inconvenient. This should make the application lighter which hopefully will also reduce battery draining. To achieve this a less calculation intensive filter will be applied.

Secondly the current implementation is a bit too strict with detected edges. There are almost no false positives but this also leads to a lot of edges missing or them being too light for the human eye to properly recognize. In perfect lighting and after a proper render time the current application works properly, but these limitations are too strict for the use of this application. Also I believe that even though with the current implementation because of the lack of details coloring the edges is a smart choice. When more details are added the coloring becomes unnecessary.

This thesis explores the world of the Line World through human eyes and improves it in a way where it is possible to live and complete most tasks in the Line World. The improvements were done considering both the usability of the application as well as the human aspect where some false positives combined with lots of extra details give a much more stable result.

The end goal of this application is to figure out what humans can do in the Line World where life might be harder with the lack of details but maybe there is some tranquility in the lack of extra noise and details that we are nowdays blasted with daily. This kind of technology can also be used elsewhere like computer vision applications where the lack of details does not cause problems. Or maybe machine learning algorithms can detect things with just edges that humans could not see. The possibilities are vast and it is impossible to imagine all of the awesome things the future holds.

Keywords: VR, application, edges, false positive, usability

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Samuel Schmidt: Living in the Line World
Kandidaatintyö
Tampereen yliopisto
Degree Programme in Computing and Electrical Engineering, BSc
Kesäkuu 2023

Tämä työ jatkaa Aleksi Viljasen[1] ja Anni Lignellin[2] työtä. Tavoitteena on korjata ja parantaa ongelmia joita Lignell löysi hänen työnsä aikana ja parantaa Viljasen applikaation käytettävyyttä. Päätavoitteena on luoda VR applikaatio joka mahdollistaa ihmisten elämisen Line Worldissä. Eli ympäristössä jossa näkyy vain esineiden ja asioiden ääriviivat ja tavoitteena on selvittää voivatko ihmiset elää sellaisessa maailmassa.

Projectin päätavoitteet ovat saada applikaatiosta paljon sulavamman, sillä nykyisellä implementaatiolla näkymällä voi mennä hetken aikaa kokonaan renderöityä. Tämä voi aiheuttaa pahaa oloa ja on yleisesti epäkäytännöllistä. Tämän pitäisi myös tehdä applikaatiosta kevyemmän ja toivottavasti vähentää applikaation akun kulutusta. Tämän saavuttamiseksi implementoimme vähemmän kalkulaatiota vaativan filtterin.

Toiseksi, nykyinen implementaatio on liian tarkka ääriviivojen tulkinnassa. Vaikka nykyimplemetaatiolla ei ole melkein yhtään false positivea, johtaa tämä usein siihen että ääriviivat jäävät liian vaaleiksi tai jättää havaitsemtatta joitain oikeita ääriviivoja. Täydellisessä valaistuksessa ja tarpeeksi pitkän processointi ajan kanssa nykyinen applikaatio toimii hyvin, mutta nämä ovat liian tarkkoja vaatimuksia applikaation käytännöllisyydelle. Uskon myös että, vaikka nyky implementaatiolla esineiden ääriviivojen värjäämisessä on järkeä, uusilla vaatimuksilla värjääminen on tarpeetonta.

Tämä työ tutkii Line Worldiä ihmissilmien läpi ja pyrkii parantamaan sitä tilanteeseen, joissa suurin osa tehtävistä pystytään suorittamaan Line Worldissä. Päivitykset applikaatioon tehdään huomioiden ottaen applikaation käytettävyyttä, jotta se olisi mahdollisimman käytännöllinen ihmisille. Kun lisäämme muutaman false positiven ja saamme paljon enemmän yksityiskohtia, voi ihminen toimia siinä ympäristössä paljon paremmin.

Applikaation tavoitteena on selvittää mitä ihmiset voivat tehdä Line Wordlissä missä elämä voi olla vaikeampaa ilman yksityiskohtia, mutta ehkä Line World sisältää eräänlaisen rauhan ilman liikaa ylimääräistä hälinää ja yksityiskohtia joita kohtaamme nykypäivänä enemmän kuin tarpeeksi. Tällaista teknologiaa voidaan myös muualla kuten computer vision applikaatioissa, joissa yksityiskohtien puuttuminen ei aiheuta ongelmia. Tai ehkä machine learning algorytmit voivat havaita asioita ääriviivoista joita ihmissilmä ei pysty havaitsemaan. Mahdollisuudet ovat laaja ja on mahdotonta tietää millaisia mahtavia asioita tulevaisuus sisältää.

Avainsanat: VR, false positive, ääriviiva, applikaatio, käytettävyys

CONTENTS

1. INTRODUCTION

First Aleksi Viljanen[1] created the Line World application. A VR application that creates an environment out of the edges it detects with the camera of a phone. Then Anni Lignell[2] proved that humans can indeed live in the Line World but there are some serious limitations with the lack of details and the problems it faces in different lightings. But the beauty of the application is that it can be run on a very basic smartphone with very limited processing power and a simple camera. While humans may never need to live in the Line World maybe experiencing a world of only edges can help humans understand the limitations and benefits of such an environment and help us develop applications that require only a small amount of processing power as well as a simple camera to do some complex tasks. This could be used in small machines with limited processing power in highly cluttered environments like forests or cities where perhaps the only necessary information can be seen in the edges.

Making the application run smoother and improving the overall usability and details of the application will lead to a better experience in the Line World but it also allows the users to see the limitations and benefits that the Line World provides more clarity. The purpose of this would be that the users would have a better understanding of the requirements and restrictions for applications that rely on edge detection based vision of the surrounding area.

This thesis will focus on firstly implementing a lighter filter that still achieves good edge detection while also making the application drain less battery and making the view much smoother and secondly, the filter will be given such parameters that it is less strict with the edges it detects. So that the end product would be as usable as possible for humans. Next, we will implement some preprocessing steps for the original image and hopefully enhance the end result. Then we will discuss some other possible improvements and implementations for the application. Lastly, we will summarize the work and will share some thoughts about the possible future and the current state of the Line World.

2. THEORY

The original application by Viljanen[1] is too heavy to run smoothly on, at the time of writing, a basic smartphone such as the Samsung Galaxy S6[3] that was used to test the application. The original application is implemented with Canny edge detection which is considered by many as an optimal edge detector[4]. The Canny edge detector is implemented in 4 steps as described in the review of the Canny edge detection[4]. To make the filtering lighter to run we either need to think of another edge detector or trim the existing steps. So the obvious approach is to use the Sobel edge detector since it is the basis of the Canny edge detector and it is a very widely used edge detector.

2.1 Sobel edge detection

Before using the Sobel edge detection we will reduce noise with a Gaussian blur filter and convert the image to grayscale to make the edge detection more effective The Sobel edge detector calculates the first derivate of pixel values in both horizontal and vertical directions with a kernel. When it detects a sum of gradients that are higher than a certain threshold it detects an edge[5]. The derivative can be visualized by a pixel intensity function and its derivative function as seen in figure 2.1[6].

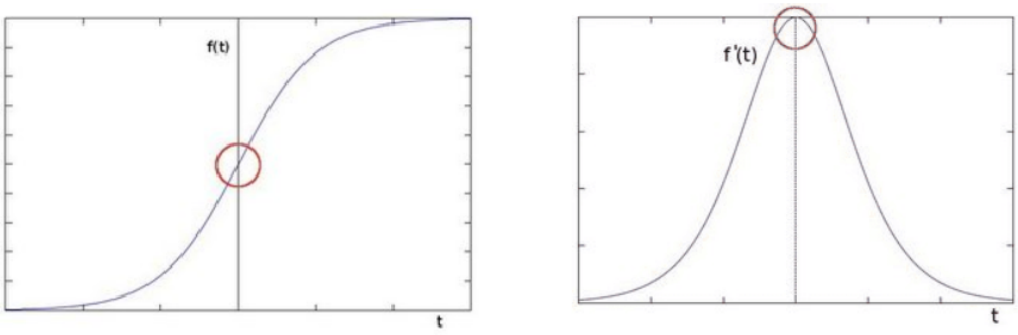

Figure 2.1. *Pixel intensity function and its first derivative*

The result from the Sobel edge detection also produces less strict edge detection so even though there are more unnecessary edges there are also plenty more details that make complex environments easier for humans to recognize and navigate. The difference

between the Canny edge detection and the Sobel edge detection can be seen in figures 2.2-2.4[7].

Figure 2.2. *Base image*

Figure 2.3. *Sobel edge image*

Figure 2.4. *Canny edge image*

And as we can see the Sobel edge detector should produce more edges although some may be unnecessary the extra detail helps the human eye recognize items in more practical situations.

2.2 Image enhancement

In addition to making the application smoother and the edge detection less strict we are trying to make the view as optimal as possible. So enhancing the image and making it as clear as possible is important. An important part of any video-based application is preprocessing and our application is no different. We already apply a basic Gaussian blur to the image and change the image to grayscale, but there are many more ways of improving the initial image. A basic problem with the original application is that it performs poorly in anything but perfect lighting, while it would be hard to make the application work perfectly in poor lighting we can at least slightly improve the situation by applying basic enhancements that improve image quality in poor lighting conditions while being light operations so that we don't undo the benefit we gain by switching to the Sobel edge detector. To this end, we will be applying basic contrast enhancement to the image to hopefully enhance the result. Commonly used processes are multiplication and addition which are called gain and bias[8]. These terms are more commonly known as contrast and brightness. By increasing the contrast we can add more lines detected since it amplifies the differences between pixels, but this also leads to more false positives. Another form of image enhancement is gamma correction which can be used to remove the non-linear mapping between input radiance and quantized pixel values[8]. While this makes images brighter and clearer to the human eye and it has a more adaptive effect as contrast enhancement it is unclear whether this will help edge detection. So the methods need to be tested to see if either makes the application better or not.

3. TESTING AND IMPLEMENTING

This chapter will be divided into three parts. First, we will simply implement the Sobel edge detector and test the results we get. Then we will apply some preprocessing steps and test the application. Finally, we will test different kernel sizes and we will try to find the best combination of our image processing steps. All the code is visible in the GitHub repository[9].

3.1 Implementing the Sobel filter

Since this thesis continues the work of Viljanen and Lignell we will continue using the open-source computer vision library OpenCV[10] and its built-in Sobel derivatives function[6] which we can use to detect edges as explained in the theory section. So we simply use the built-in function and we will also disable the edge coloring implemented in Lignells thesis[2] since the added amount of detected edges combined with colored edges makes the image too messy and light colors make the edges softer and harder to recognize. The results of this can be seen in Figures 3.1-3.3.

Figure 3.1. *Original image*

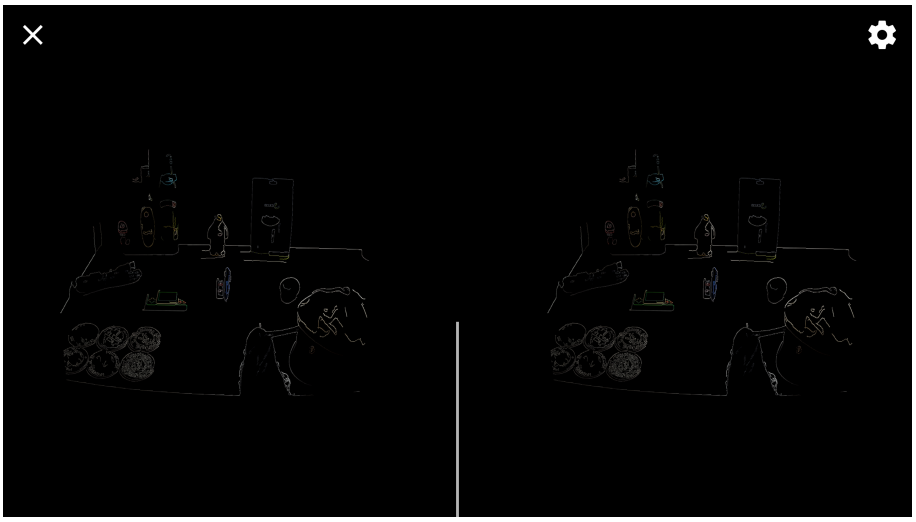

Figure 3.2. *Canny edge image*

Figure 3.3. *Sobel edge image*

The results seem already promising since the application now runs much smoother and even though without colors the items are harder to precisely recognize. The general shape is easier to recognize making the object easier for the human eye to classify into broader groups. Which is more important for life inside the Line World.

3.2 Image preprocessing

To strengthen the edges detected and hopefully remove some of the excess edges from the image trying gamma correction makes sense to enhance the contrast in the image. Also in the same sense contrast enhancement makes sense so we will try both with large enough multipliers to see what kind of changes they cause in the images so we can try and find the optimal combination of both of them. Brightness correction could be hard to implement since we would need to make such an adaptive brightness correction value that takes into account the average lighting of the image and also it should consider

whether there are large brightness differences in the image. That is why we won't implement brightness correction here since we could find good values for this situation but they likely wouldn't work in other lighting situations. The results of our gamma correction and contrast enhancement can be seen in Figures 3.4 and 3.5.

Figure 3.4. *Contrast enhanced image*

Figure 3.5. *gamma correction*

Sadly neither of the preprocessing operations seems to have the desired effect. Both make the image messier and make especially the smaller objects harder to recognize, while not really getting rid of any excess edges. So at least in this application, the original image gets better results than the processed results.

3.3 Trying different kernel sizes

We started with a 5x5 kernel which yielded a pretty good result. But we can also use a larger or smaller kernel to increase or decrease the number of edges detected. The

larger the kernel the more edges we will detect, so in this case we most likely want to either keep or lower the size of our kernel. We will still test a larger 7x7 kernel to see the effects a larger kernel has on the edge detector, but since we want to reduce the amount of unnecessary edges detected we most likely won't use the larger kernel in our final application. For the smaller kernel, we will use a 3x3 kernel, but this Sobel kernel size can produce noticeable inaccuracies. That is why the OpenCV library uses a Sharr function that is as fast but more accurate than the Sobel function would be[6]. The results are shown in Figures 3.6 and 3.7

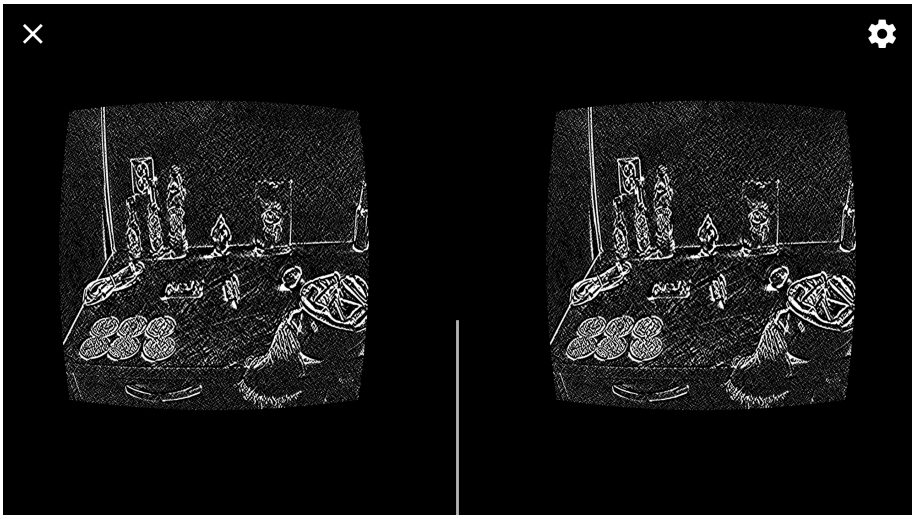

Figure 3.6. *Sobel function with 7x7 kernel*

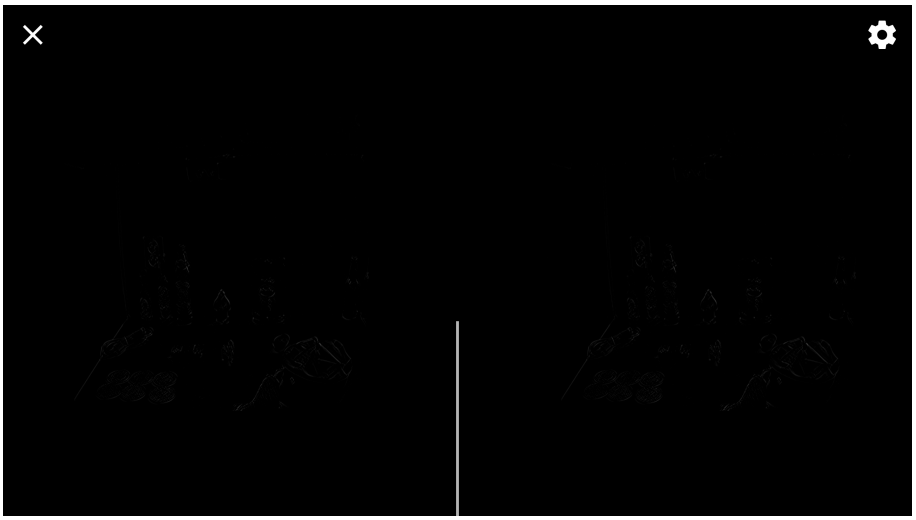

Figure 3.7. *Sobel function with 3x3 kernel*

As predicted the larger 7x7 kernel produces even more unnecessary edges as well as increases the graininess of the image. Whereas the smaller 3x3 kernel removes the graininess of the 5x5 kernel. Sadly it also makes the image way too dim and basically produces the same edges as the 5x5 kernel.

4. RESULTS

The tests were successful in finding the best combination of preprocessing methods and kernel sizes. Surprisingly the first implementation of a 5x5 kernel with no other image preprocessing then adding Gaussian blur and changing to grayscale resulted in the best result

4.1 The application now

Now that the edge detector has been swapped to the Sobel edge detector the application generates enough edges in very average lighting for humans to not bump into everything. But also a major improvement in the application is that items have more distinct edges so that we can deduce what kind of item it might be. Before the application only allowed recognizing items in very good lighting since otherwise the edges would often be too soft and the general shape of the items was hard to see. With these results, we could live our daily lives in the Line World but complex tasks or tasks that require high precision would still be very hard if not impossible to do in the Line World. Still, there is some tranquility in the reduced amount of information the brain receives in the Line World and it helps to focus on the task at hand. Also, we were able to make the edge detector less calculation intensive which makes the application run smoother and makes the user experience a lot more enjoyable. In the future or even at the time of writing, phones and VR devices can be way more powerful and more optimized for VR use than our Samsung Galaxy S6 which would allow the application to become more demanding. And maybe it could have some features that help to live in the line world like making text readable or using the Canny edge detector with more relaxed requirements but still being able to run smoothly. But with the current implementation that is not necessary and life in the Line World would probably be completely possible. The question only remains, what do you want to do in it?

4.2 Conclusion

We successfully improved the application so that it runs smoother and more consistently. We also successfully relaxed the edge detection requirements so that we have a few more false positives and unnecessary lines but also a lot more details and context for the items,

allowing us to recognize item shapes and deduce what specific item we have. Overall now life in the Line World seems possible and it would be exciting to see what comes next.

REFERENCES

[1] Viljanen, A. *Aleksi Viljanens Bachelor thesis. IMPLEMENTING VR FEATURE CAMERA ON ANDROID PLATFORM*. 2020.

[2] Lignell, A. *Anni Lignells Bachelor thesis. LIVING IN A LINE WORLD How life would be in a world where we could merely perceive edges*. 2022.

[3] Samsung. *Samsung Galaxy S6*. URL: https://www.samsung.com/us/support/mobile/phones/galaxy-s/galaxy-s6/#benefits.

[4] Vasundhara, R. R. ; S. N. ; V. *A Review on Edge detection Technique "Canny Edge Detection"*. 2019. URL: https://www.ijcaonline.org/archives/volume178/number10/ramnarayan-2019-ijca-918828.pdf.

[5] Gary, K. A. ; B. *Learning OpenCV 2nd Edition*. 2014, chapter 5. URL: https://learning.oreilly.com/library/view/learning-opencv-2nd/9781449331955/chapter-40.html.

[6] OpenCV. *Sobel Derivatives*. 2022. URL: https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[7] LearnOpenCV. *Edge Detection Using OpenCV*. URL: https://learnopencv.com/edge-detection-using-opencv/.

[8] Szeliski, R. *Computer Vision: Algorithms and Applications, 2nd ed.* 2022, pp. 110–112. URL: https://szeliski.org/Book/.

[9] Schmidt, S. *LineWorld git*. URL: https://github.com/Vesimies80/LineWorld.

[10] OpenCV. *About OpenCV*. 2023. URL: https://opencv.org/about/.