Comparing the Effectiveness of Color Spaces in Identifying Traffic Lights Using OpenCV

Word Count: 4263

ABSTRACT

Autonomous vehicles are a quickly developing industry around the world. As companies compete to find the best way to become fully autonomous, many options can be overlooked. Traffic lights and intersections account for many crashes and deaths worldwide, so accounting for them is essential for safe autonomous travel. The detection of traffic lights is still at a very primitive phase, being a developmental feature for the extremely prominent company, Tesla. Using color thresholding to detect traffic light state could be an option, but then the question of which color space is introduced. Color spaces determine how a computer will process a pixel. With the variation of color spaces due to light or surroundings, a compliant but accurate color space would be needed. By comparing RGB, HSV, and YCbCr, this research was conducted to determine if there is a superior color space available for tracking traffic lights. The best threshold ranges were found with 90 different sample photos for each color space, finding whether the code created to perform my experiment could properly determine the color of the light. Using a Chi-Square Test, I determined whether differences in accuracy between the color spaces. After this experiment, there was found to be no statistically significant difference between the color spaces. All color spaces would be equally applicable to autonomous cars and other computer vision applications.

I. INTRODUCTION

Technology is evolving in our world at an unimaginable rate. From mobile technology to quantum computers, developments in electronics and computers have led the world to where it is today. One major accomplishment is the development of computer vision. “If a creature-either biological or mechanical-is to interact effectively with its environment, it needs to know what objects are where. Computer vision provides a primary method for understanding how to make

intelligent decisions about an environment, on the basis of sensory inputs" (Grimson and Mundy 1994). By emulating human vision and processing, simple jobs can be taken over, potentially with more accuracy than humans. This can be used in countless applications, including autonomous vehicles. "Uber's self-driving prototypes use sixty-four laser beams, along with other sensors, to construct their internal map; Google's prototypes have, at various stages, used lasers, radar, high-powered cameras, and sonar" (Union 2018). Although cameras are referenced, the way computer vision is used is unknown. Computer vision has a variety of types and methods, each with its own level of intensity, process and level of computing power. In a competing economy for success, the best method is still unknown to the world-and potentially to the companies themselves. Companies to find a method that is accurate, fast, and not in need of a lot of computing power. Autonomous vehicles need to be able to react to an unexpected change in the road, such as a crash up ahead. Also, computers with extreme processing power cannot be used due to the amount of money and power they will consume. Traffic lights are an aspect of the road that autonomous vehicles will need to master to be viable on the road. Color detection could be a viable option to this issue but finding a viable color space to use is necessary. With the plethora of different color spaces, each with their method of detecting and reading colors, developers would need to pick one that would give the most accurate results. By comparing three common color spaces (RGB, HSV, and YCbCr), a conclusion can be made on which color space is optimal for traffic light detection.

II. REVIEW OF LITERATURE

Computer Vision refers to the use of computers to process images in order to retrieve-data from them using various algorithms. From an engineering standpoint , computer vision is defined by T. S. Huang (n.d.) as a process that "aims to build autonomous systems which could perform

some of the tasks which the human visual system can perform (and even surpass it in many cases)". The evolution of the technology has allowed it to surpass human vision, but it can be underdeveloped in certain regions. Computers will often try and make predictions of what can happen and what would be needed, and not every situation can be predicted. If a computer cannot process what is happening quickly and accurately, in the case of autonomous cars, a crash could occur.

There are a variety of color spaces that can be used, each providing their own advantage in different scenarios. A color space is a set of numbers representing a color on a screen, or the

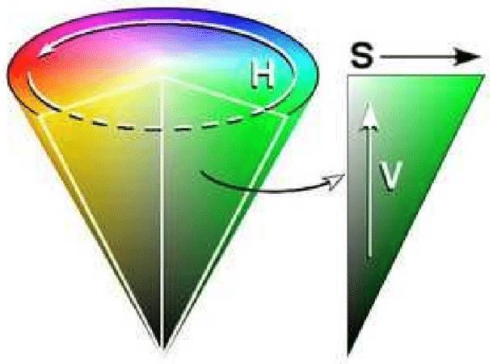

Figure 1. Visual diagram of the HSV color space and the effect of manipulating each of the components (van den Broek 2005).

metaphorical "languages" that a computer associates with for a given color. Each number changes its own aspect within the color, such as brightness, overall color, or intensity. For example, in HSV, H stands for hue, S is saturation, and V is value (Figure 1). The hue manipulates the color represented, shown at the top of the cone. Once a hue is selected, a triangle showing the saturation and value is made to better represent them. As value increases, the color

goes from dark to light. As saturation increases, the color goes from white to its true color. Another commonly used color space is RGB, representing red, green, and blue. The mixing of lights on a computer screen is different from mixing paints, making red, green, and blue the metaphorical “primary colors” on many monitors (Image-1 n.d.). In many monitors, a pixel is made of a red, green and blue light, and the intensity of each determines the overall color. In this format, the overall color and light of a certain color is impacted by all 3 aspects of the color space directly by changing these small lights in the display. Lastly, YCbCr represents a more complex color space, where the parameters Y, Cb, and Cr, representing luma, chroma-red and chroma-blue, respectively. Chroma-red and blue represent the difference between the red or blue value and a given reference value. Commonly used in digital video, the brightness is primarily controlled by Y, and the color is controlled by Cb and Cr (Basilio et al. 2011). Each of these color spaces has a unique way of defining different colors, causing different forms of change from one color to another.

With the abundance of options and applications of color spaces, many combinations are yet to be tested. A notable exemplar is “Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation” from the International Journal of Emerging Technology and Advanced Engineering. This research compared the color spaces by converting sample images into the respective color spaces, performing the color image segmentation algorithm, and comparing the final image outputs. This resulted in HSV performing better than L*A*B* (Bora 2015). While this research found HSV to be best, this might not always be the case.

Color image segmentation is one of many methods of computer vision. A more advanced version of computer vision, deep learning provides its own benefits and disadvantages. “Deep

learning allows computational models of multiple processing layers to learn and represent data with multiple levels of abstraction mimicking how the brain perceives and understands multimodal information, thus implicitly capturing intricate structures of large-scale data" (Voulodimos et. al 2018). Deep learning uses hundreds of samples to develop a pattern and method to find targets on its own. Once established, it is highly effective. The downside is the processing power and time needed for such a task. As much as this would be a powerful method to use, it is not practical to put a computer with the capacity to run deep learning applications in a vehicle. The various methods, along with color spaces, provide a range of needed research in the computer vision field.

In today's age, computer vision is constantly being implemented into devices for consumers. Commonly as a security feature for phones or homes, computers are able to spot specific faces or people to perform certain actions. Apple, the developers of the famous iPhone line of cell phones, have started using computer vision in 2010 with their operating system iOS 10. Their machine learning journal explains their development of "an algorithm for a deep neural network for face detection that was feasible for on-device execution" (Computer Vision Machine Learning Team 2017). As this form of software becomes more and more popular, it can continue to be improved so it can be used in more advanced applications. Another mobile application has been Google's translation through a camera. Released in 2019, Google added a feature to their Google Translate app that allows the user to point at a "flyer or sign and get results in your native tongue even if you don't know what language you're reading" (Johnson 2019). This development is a major milestone for those travelling, as they no longer need to spend time typing into a translator or finding someone who can translate for them. Instant digital translation

is now possible, expanding the possibilities for many. These two examples only mark the beginning of where computer vision can improve society.

One of the developing technologies for computer vision is autonomous vehicles. Autonomous vehicles have become a topic of interest for the future of transportation globally. The public is anxiously awaiting the day where everyone will no longer have to drive, but this day may still be far into the future. Deficiencies in their automation results in a need for human intervention in unpredictable moments, delaying the progress to full autonomy (Bayern 2019). Along with the mechanical and electrical limitations stopping autonomous vehicles, the software behind a vehicle's actions is a major component to success. The number of indicators, signs, and warnings on the road are massive, making a computer's job of processing and responding to all of them extremely challenging. In this technological revolution, completely self-driving cars were expected to be here. Despite this, "proponents in the industry, including Tesla CEO Elon Musk, Waymo CEO John Krafcik and Cruise CEO Dan Ammann, touted an aggressive timeline but missed and reset their goals" (Kolodny 2019). The failure of these key players in the industry to achieve their goals indicate a major stopping block in development. Pinpointing these issues and developing solutions efficiently can provide the public with the future's form of transportation faster and safer.

One of the most important signals are traffic lights at intersections. The U. S. Department of Transportation reported that in 2019, over 50% of fatal or injury crashes occurred at intersections. If these crashes could be prevented by an accurate light detection software, it should be the priority of autonomous vehicle developers. Even if everyone doesn't own a fully autonomous car, implementing software that can prevent a driver from running a red light could create a dramatic decrease in this statistic. Fortunately, some companies, such as Tesla, have

made primitive versions of this detection functional. YouTube videos users discovering the feature, that was accidentally released early, detecting and stopping at traffic lights (Greentheonly 2019). As it was not an official release, the video shows what Tesla has developed so far in traffic light detection. Although it was able to detect some lights, the software was still very inaccurate and not always able to detect the traffic light. Improving the accuracy rate of these software could depend on how traffic lights are detected. I believed that by using OpenCV (an open-source computer vision library in Java) I would be able to compare the HSV, RGB, and YCbCr color spaces, a combination not yet seen, let alone not with a focus on traffic lights (About 2019). Through analysis of various papers, I did not come across any research focusing on the varying conditions of traffic lights and the feasibility of different computer vision options to solve this issue. I compared their effectiveness of accurately displaying the state of the light in an attempt to find a statistically significant difference to declare an effective means of detecting lights in autonomous vehicles. Doing so may accelerate the development of autonomous vehicles to provide safer travel.

III. METHOD

My method was a comparative analysis of the HSV, RGB, and YCbCr color spaces. A comparative analysis is a common method for computer vision research, so I chose to follow the trends of the field. Comparative analysis displays the similarities and differences between groups in an experiment. The groups were examined and compared for effectiveness. In my research, the aspect being compared was the percent effectiveness of finding traffic lights. A code was created in the same manner, using each of the spaces with the same intent of detecting traffic lights. After the code was executed, I compared the results to the actual photo and determined the percent of pictures the code was able to get correctly.

The backbone of my research was the OpenCV Library. This library provided the tools needed to detect the traffic lights by color. By programming commands that check each pixel of a photo, it turned the image from colorful to pure black and white. White indicated the pixel fitting the criteria provided, such as having a high concentration of red. This criterion can be set to find and separate red, yellow, and green spots. These groups were then organized into what would be believed to be traffic lights, providing my output. I chose this form of computer vision because it was relatively less intensive on hardware compared to others, such as deep learning. Efficient processing is necessary, as a split-second delay in processing could determine life or death in a fast-moving car. The use of OpenCV is justified by the numerous companies that rely on it for their processes, such as Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, and Toyota (About 2019). The reputations of these companies signify that OpenCV is reliable in computer vision. These companies control most of today's technology, so knowing they use this shows that it will not have data-corrupting bugs or errors.

OpenCV is available in a variety of programming languages. The language chosen to program in for this research was Java. The language should have had no effect on the results, as the library's functions were consistent between all of them. I preferred the Java library because it was the language I had the most experience in, allowing me to execute this research as flawlessly as possible. To write this code, I decided to use the IntelliJ IDEA IDE. As a free software, IntelliJ allowed me to write the code in an environment intended to debug and run software. The IDE has capabilities that made the code creation process simpler, such as suggesting what it believed I would write next, allowing me to quickly fill the line with what I needed through their "Deep Intelligence" (IntelliJ n.d.). This made the creation process simple and very efficient. While

finding the functions that best assisted my task, the IDE directed me to what a specific function was doing and what it required so I could write accordingly.

The function of the code was to create a color threshold for each spot on the traffic light and group them so that they showed the 3 different lights. Each color space had a set of values, which may change in different ways depending on the conditions. The color of each part of the light and which is illuminated can both have varying effects. My goal was to create a range within each color space that correlates to each color of the traffic light and its current phase that would consistently track its target in a variety of images. Finding these ranges was through trial and error, where I took a few sample images and develop an effective range that I used on a larger set of different images. The images used in the smaller and larger set were randomly selected from the full set of images, and the same ones were used for all color spaces. The randomization prevented bias of images that are easier to detect, allowing challenging images to be input into the testing. These challenging images helped indicate the best color space in my results. If all color spaces are able to get the more simplistic samples, the more challenging ones will be the deciding factor on which is declared the most effective.

The method I used for finding these values was selectively constraining variables. First, I set the code to accept the full range of colors, turning everything white. I then slowly adjust the maximums and minimums of one value until I got a close range where the maximum and minimum were only about 20 apart. Depending on the color space, these values could have a range of 0 to 100 or 0 to 255. I then repeated this for the other 2 parameters of the color space. If the code was still not able to detect the blob of valid pixels or had too many pixels reading as valid, I returned to the values and attempt to make reasonable changes.

Using OpenCV, there was a process that each photo had gone through to determine the state of the light. Each color space range was saved in variables that were referenced in loops throughout the code. The first step was using a threshold on the file. This checked each pixel of the file to see if it qualified in the given range. Qualifying pixels were turned white, and the rest were turned black. The processing steps all worked in binary images exclusively, so white represented positive and black represented negative. This output was then sent into a dilation, which expanded any white sections to make them more prominent in the image. The amount dilated was equal to all files, so it would not create error in my data. Following the dilation, I blurred the image to fill any holes the white section may have. Holes can easily disturb the detection process, so eliminating them was needed. The blur selected was a median blur a radius of 10. Median blurs take the median of the given area and use that to determine the pixel status. This final image was sent into the blob detection portion. This searched for pieces of white that were somewhat circular and had a minimum area of 30. The minimum area prevented miscellaneous pixels from being detected in the tracker. The blobs were then recorded in number and the size of the largest for each light state was logged. The color state with the largest sized blob in the image was the predicted state. A blob is a circular collection of white pixels, so the largest one is the largest group of pixels, which I assumed to be the light. This process was performed on every photo in my data set, and the recorded results were kept in a generated standard text file (See Figure 2).

```
File: file1.jpg
Number of Red Blobs: 0
RedMax: 0.0
Number of Yellow Blobs: 0
YellowMax: 0.0
Number of Green Blobs: 1
GreenMax: 91.4503173828125
Predicted Status: Green
```

Figure 2. Sample output of file named “file1.jpg”, with 0 red nor yellow blobs, and one green blob with an area of approximately 91.45.

To obtain these samples images, I decided to use a set of images from past research found on the open source code-development website GitHub. The images used by these researchers, under the name “level5-engineers”, proved to have a variety of locations and lighting conditions, helping test my code for all intended purposes (level5-engineers 2017). The 4499 available images all followed a consistent file format and were organized into respective light status folders, which allowed me to conveniently select the necessary amounts from each. The consistent file format ensured that photo quality or format would be not another independent variable in my study that would have impacted the results obtained.

With my programs completed testing, I had a collection of outputs, stating each scanned photo and the predicted amount and phases of traffic lights in the photo. I compared this data visually to the actual picture and recording if the code for that color space in that photo was correct or incorrect. For statistically accurate results, I used 90 photos, as this number is sufficiently large. There was 30 of each light status: red, yellow, and green. With each color space, I developed a proportion of correct readings. With the 3 proportions, I found an average proportion and considered this part of my null hypothesis. I conducted a Chi-Square Test to determine if the differences between the color spaces were statistically significant.

IV. RESULTS AND ANALYSIS

To begin my data collection, I first found the values for each traffic light color that the color spaces are able to track most accurately. These ranges were kept small to prevent false positives from being detected by the code. After randomly selecting and using one photo as the basis of finding these values, I came up with the minimum and maximum values that allowed each color space to accurately find red, yellow, and green lights, as seen in Tables 1, 2, and 3.

Table 1

Threshold Parameters for RGB Color Space

Parameter	Red Light	Yellow Light	Green Light
R	200, 255	0, 10	80, 125
G	200, 255	200, 255	0, 100
B	0, 70	165, 255	70, 150

Note. All values are displayed in the format (Minimum Value, Maximum Value)

Table 2

Threshold Parameters for HSV Color Space

Parameter	Red Light	Yellow Light	Green Light
H	0, 10	225, 255	250,255
S	20, 45	200, 255	242, 255
V	60, 100	215, 255	110, 255

Note. All values are displayed in the format (Minimum Value, Maximum Value)

Table 3

Threshold Parameters for YCbCr Color Space

Parameter	Red Light	Yellow Light	Green Light
Y	80, 125	245, 255	80, 100
Cb	200, 255	100, 200	0, 50
Cr	150, 255	25, 125	25, 125

Note. All values are displayed in the format (Minimum Value, Maximum Value)

With these values established, I performed my tests on the 90 photos I randomly selected. The code was programmed to automatically run through the photos and display the results in a file, so all of the processing wasn't visually displayed. I have pulled the output stages from one photo from the code to display the computer process in Figure 3.

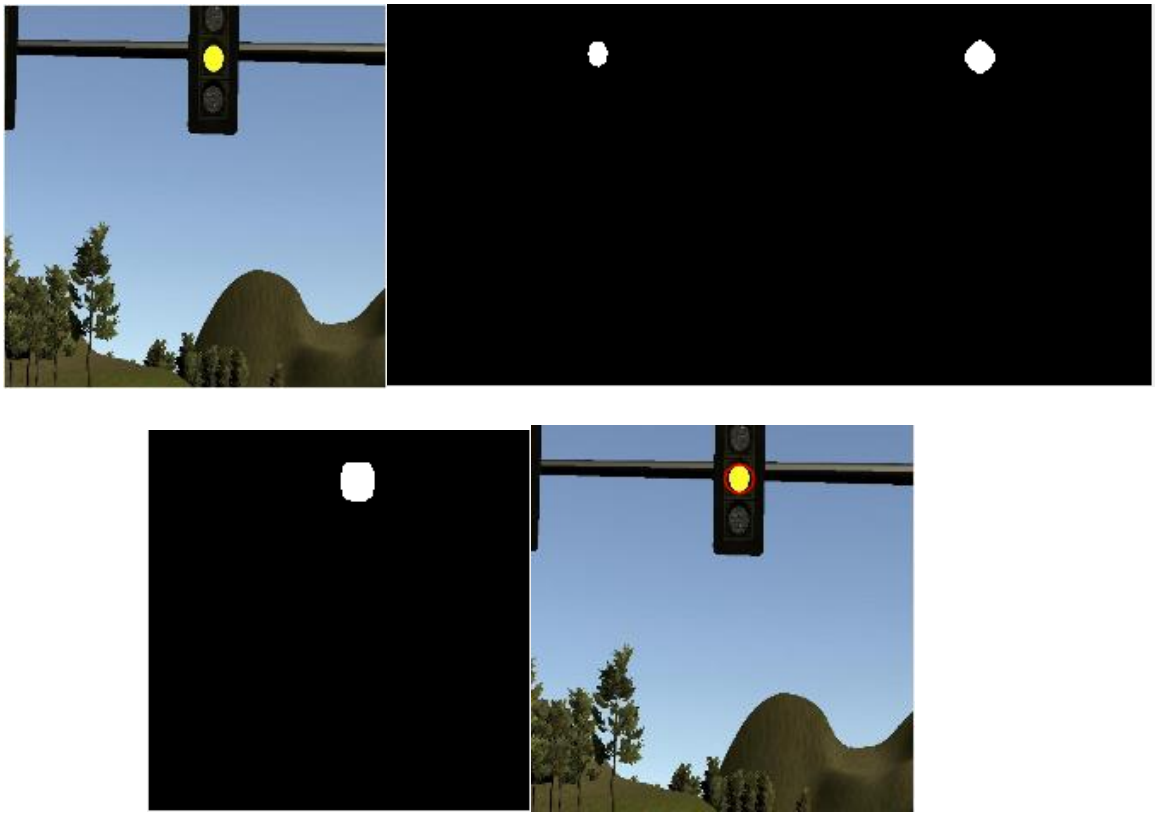

Figure 3. Images of the computer vision process, ordered from beginning to end process

Now with my data, I compared the color spaces and their ability to successfully detect the traffic lights. I collected the number of successful detections of the red, yellow, and green lights along with an overall success count. To perform a statistical analysis, I created the null hypothesis that all color spaces were equally effective. I decided to make my null mean the average number of successful reads of all 3 color spaces. Using that, I performed a Chi-Square Test of Independence to determine any significant difference between the color spaces.

Table 4

Number of Traffic Lights Correctly Identified by RGB, HSV, and YCbCr

Color Space	Number of Correct Red Lights (Out of 30)	Number of Correct Yellow Lights (Out of 30)	Number of Correct Green Lights (Out of 30)	Number of Correct Total Lights (Out of 90)	Expected Value
RGB	19	23	21	63	50.66666667
HSV	4	22	18	44	50.66666667
YCbCr	0	21	24	45	50.66666667
Results			Chi-Square = 4.513157895	df = 2	p = 0.104708

As seen by the data in Table 2, the relation between color space and overall effectiveness was not significant, X^2 (2, N = 90) = 4.513, p = 0.1047. I wanted to look further into the individual light statuses because the red-light detection count for YCbCr appeared to be an outlier, as all red photos failed to be detected. For red traffic lights, there was a significant difference between the traffic lights, X^2 (2, N = 30) = 26.174, p < 0.05. With an average correct red-light detections of 7.67, the RGB detector was the only one above average, indicating it was the best performing compared to the other two. The opposite result came from the yellow and green lights, as both showed non-significant differences, X^2 (2, N = 30) = 0.091, p = 0.956, X^2 (2, N = 30) = 0.857, p = 0.651. Overall, the color spaces proved to be equally effective, but RGB prevailed in-individual analysis because of its effectiveness in the red-light images.

V. LIMITATIONS

Although my hypothesis proved to be false, there were some limitations that should be discussed. First, the sample of photos I used could not have shown all forms of traffic lights on the roads. The variety of shape, color, and type of lights are only a few factors that vary between towns or even just individual roads. If a traffic light is brighter or makes it appear whiter to a camera, the program with the parameters I found may not work as well as in my sample. In this situation, one color space may be able to pull ahead if the color and concentration of white are independent parameters. While I attempted to find samples that were as variable as possible, not every traffic light has a picture available online. Secondly, single photos may have different results than a moving camera feed. The autonomous cars my research is intended to benefit would be moving, so the image feed would be constantly changing. I had planned on originally running my code on a camera feed on a Raspberry Pi microcomputer, but financial and safety factors led me away from that idea. The resources to make a portable computing system and put it onto a car was something unobtainable for me. Finally, my ability to select the best parameters could have been a source of error that could affect my data. As much as I tried to ensure that the parameters set were as precise as needed for the sample photo, slight variations could provide better results, potentially solving the issue seen in the YCbCr data.

VI. CONCLUSIONS AND DISCUSSIONS

I have concluded through my research that between RGB, HSV, and YCbCr, the choice of color space has no difference on overall ability to detect traffic lights. This opens the road for developers to freely use any color space available or preferred by them. Although the overall success rate is likely not a level safe for cars, it lays the foundation for improvement by other researchers with a higher expertise in this field.

Although my overall data had no statistically significant difference, the individual light categories showed RGB being significantly better that detecting red lights. Although its number of detected lights was significantly higher, I believe an error could be a cause due to YCbCr surprisingly detecting zero red lights, as seen in Table 4. I could not find the reasoning for this error, as it happened in numerous attempts of fixing it. A diagnosis of this issue and a recreation of my research could be a topic to be investigated in the future.

Although my data was intended for autonomous vehicles, this data can be used in other fields where color-based computer vision is needed. The constant need for autonomy in industry, simple color-based vision could help companies move products or keep inventory. For tech giants such as Amazon, these robots are essential for getting their products to their customers as fast as possible (Simon 2019). As more companies take on an approach similar to Amazon, knowing that all color spaces would be viable in a color-based system would make development simpler for the programmers. As more companies take upon an automated system, customers will be able to get their products faster and for cheaper shipping costs.

Despite a non-significant result, more research must be done to make it usable for real vehicles. Primarily, using similar code in a moving vehicle with the current equipment being used in a semi-autonomous vehicle would be essential before making it part of a retailed vehicle's software. The type of camera, position, and the vehicles moving at over 60 miles per hour could have significant effects on performance. Some cameras make everything much brighter than they are, while others do the opposite. These changes could alter the perception of a light to fall outside of a consistent threshold range that I have used. The speed I believe will be the biggest challenge because of blurry images. If a camera is not recording at a sufficiently fast rate, blurry images will try to be processed and could result in a poor reading. Another topic of

future research could be the color spaces analyzed. The three I selected were only a few of the many available color spaces available. Perhaps one of these color spaces could prove more effective than the others, as each has parameters representing different aspects. Research such as this has the potential to help bring autonomous vehicles to the world and change transportation forever.

References

About. (n. d.) OpenCV. Retrieved from https://opencv.org/about/.

Basilio, J., Torres, G., Sanchez-Perez, G., Medina, L. Perez-Meana, H. (2011). Explicit image detection using YCbCr space color model as skin detection. *ResearchGate*. Retrieved from https://www.researchgate.net/publication/262371199_Explicit_image_detection_using_YCbCr_space_color_model_as_skin_detection

Bayern, M. (2019, September 6). TomTom steers toward progress with autonomous test vehicle. *TechRepublic*. Retrieved from https://www.techrepublic.com/article/tomtom-steers-toward-progress-with-autonomous-test-vehicle/

Bora, D. J. (2015, February). Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation [PDF file]. *International Journal of Emerging Technology and Advanced Engineering, 5(2)*. Retrieved from https://arxiv.org/ftp/arxiv/papers/1506/1506.01472.pdf.

Computer Vision Machine Learning Team. (2017, November). An On-device Deep Neural Network for Face Detection. *Apple*. Retrieved from https://machinelearning.apple.com/2017/11/16/face-detection.html

Greentheonly. (2019, March). Tesla autopilot stopping at red lights all by itself [Video file]. Retrieved from https://www.youtube.com/watch?time_continue=55&v=umzFDfJvhLQ

Grimson, W. E. L., & Mundy, J. L. (1994, March). Computer vision applications. *Communications of the ACM*, 37(3), 45+. Retrieved

from https://link.gale.com/apps/doc/A15061339/CDB?u=nysl_li_seaford&sid=CDB&xid=12031e80

Huang, T. S. (n.d.). Computer Vision: Evolution and Promise [PDF file]. *Imaging Science and Technology - Evolution and Promise.* Retrieved from https://cds.cern.ch/record/400313/files/p21.pdf

Image-1 Introduction to Digital Images [Lecture Notes]. (n.d.). Stanford University. Retrieved from https://web.stanford.edu/class/cs101/image-1-introduction.html

IntelliJ IDEA. (n.d.) Jetbrains. Retrieved from https://www.jetbrains.com/idea/

Intersection Safety. (2019). U.S. Department of Transportation. Retrieved from https://highways.dot.gov/research/research-programs/safety/intersection-safety.

Johnson, Khari. (2019, July). Google Translate's camera can now automatically detect languages. VentureBeat. Retrieved from https://venturebeat.com/2019/07/10/google-translates-camera-can-now-automatically-detect-languages/

Kolodny, Lora (2019, November). Self-driving cars were supposed to be here already — here's why they aren't and when they should arrive. *CNBC.* Retrieved from https://www.cnbc.com/2019/11/30/self-driving-cars-were-supposed-to-be-here-already-heres-whats-next.html

level5-engineers (2017, November). Traffic Light Image Classification. *Github*. Retrieved from https://github.com/level5-engineers/traffic-light-classification

Simon, Mathew. (2019, June). Inside the Amazon Warehouse Where Humans and Machines Become One. *Wired*. Retrieved from https://www.wired.com/story/amazon-warehouse-robots/.

Union of Concerned Scientists. (2018, February). Self-Driving Cars Explained. *Union of Concerned Scientists*. Retrieved from https://www.ucsusa.org/resources/self-driving-cars-101

van den Broek, Egon L. (2005). Human-Centered Content-Based Image Retrieval. Neuroscience Research Communications - NEUROSCI RES COMMUN [PDF file]. Retrieved from https://www.researchgate.net/publication/228719004_Human-centered_content-based_image_retrieval

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018. Retrieved from https://link.gale.com/apps/doc/A585448443/CDB?u=nysl_li_seaford&sid=CDB&xid=8ea5ee4f