Eloka Obi
Intro to CV
6/1/2022
Northwestern

Intro to Computer Vision Project

Computer vision has been a major field in Artificial Intelligence over the last 60 years. Computer vision is involved mostly with acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the form of decisions. This field started to take place in the 1960s in universities. This was essentially created in order to mimic the human visual system. With this the field of robotics became more enhanced and this seemed to be the original goal with the creation of computer vision technology. This aided in assisting robots with how they specifically saw their environments and even interacted with their environments. Robots would now have a more intelligent design than before with the advent of this technology.

This most important part of computer vision or machine vision is in how it allows the machine to track where it needs to essentially go for the purposes of completing its task. Looking at the autonomous driving vehicles that Tesla is now trying to employ on a large scale, it only seems that this technology will become even more mainstream than before. Many different hardware products will use the deep learning technology in order to program robots to be more intelligent in their navigation. With this we have more dynamic machines that are able to do more in terms of their movement, ability, and functionality in general. Deep learning has really created a greater edge for integrating data more seamlessly into the total framework and giving a new feature into a technology that already is trying to mimic one body part of the human body just as deep learning tries to mimic how the human brain works and calibrates itself. This class

taught me a great deal about the history of computer vision and many of the scientists and engineers that helped to revolutionize the specific field of computer vision. I believe that I really was able to benefit from much of the history lesson that was given by the professor throughout the course and his information on many of the applications of computer vision technology in our world today. It seems that computer vision technologies will only make life more efficient. You can definitely see this in our everyday lives with the face capture on the iphone and even trying to get into parking at Northwestern where they automatically recognize who you are through the data they have compiled for their specific computer vision system. I believe that these systems will be implemented throughout more institutions like universities and public agencies in order to make them safer and less complicated. Cards and keys often get lost in the thick of things, so by having a computer vision system that is there to alleviate some of the errors in human memory and process we have technology that better makes up for this as a whole. Technology is constantly improving our lives each and every day and computer vision quite clearly showing this right now in the new modern age. In the future I want to implement more computer vision algorithms to enhance more surgical robotics and make them better than they were before. Surgical robotics is quite an imminent field that has now become more prominent in trying to assist everyday human surgeons and make their processes more efficient and easier than they were before. Minimally-invasive surgery has now been transformed by the field of robotics as there have been so many startups and companies built trying to create robots for these systems. I would like to create a startup where we try to improve on the already existing surgical robotics technology. Many of the current implementations of the computer vision technology introduced now into the surgical robotics technology have not totally experimented with the neural networks

as deeply as they should with regards to computer vision. I believe that I can aptly integrate the technology into the entire manifold with the right backing.

This specific project looked at trying to implement a computer vision algorithm in order to detect different doors and their handles across a diverse dataset of over 1000 images across the Open Images Dataset. We try to analyze door detection across three various states. These states include closed, open, semi-closed. These are crucial for our intelligent systems to be able to safely navigate in indoor spaces and environments. Usually, the task of these systems implies moving between rooms and dealing with doors. It is required to provide the robot with necessary info about the door so it can safely navigate between rooms without any problems. The bare minimum that we want to do is to distinguish a door from any other possible features.

Door state classification is useful in difficult situations. We used the Door state classification because it can be used to solve more complex robot problems. We are not going to be working directly with any robots but just the classification of which robot can potentially travel through or not. By specifying the classification, we can make the detection process seem much easier for us throughout creating our entire algorithm. By having certain boundaries on how we calculate the detection process between what is open,semi-closed, and closed, it is much easier for us to analyze the entire process for us. I believe that often we have noticed that much of the process between these encounters with doors happens at many different levels with how we perceive them to either be closed, open, or semi-closed. Computer vision technology is more detailed than the human visual system and brain system. The door detection is able to inextricably seek what it needs to do inorder to comprehensively analyze each image worth of datasets that we have chosen for our specific algorithm to analyze. Each door is unique and the algorithm and method we have chosen has been programmed in order to show this specific

change essentially. The specific algorithm chosen uses the numpy, and open cv package in order to write this algorithm for door detection. We named a class for the entire detector that we define our functions in. We defined all the parameters in the function that include threshold, shapes, templatedir, horizontal alignment threshold, the output direction, and the output detection image. Another function was then created in order to detect images and overlay a box on those images. This is where we start to use the cv2 function from the open cv library in order to change the color scale of the image to gray in order to better detect what needs to be detected in the image. We then searched for different matches in the algorithm through the open cv function match template. We then determined if the match we found was good enough according to the threshold that we set for this specific algorithm. We set the left coordinate and then we set the bottom right coordinate and set the according shape values. We then look for different shapes in an image and return whether they were found or not found. If we happen to find the different shapes, we return the top left and bottom right coordinates for this specific function. We then created a function to detect and find the current state of the door. We looked for the default setting of the state being a fail and we defaulted the detected state not having any shapes by default in our python script. If the number of shapes detected equals the number of shapes then we've detected all of the shapes in the specific image. In this specific part of the function we are looking at the horizontal differences between the shapes of triangles and rectangles. With these shapes we get defined sides for each of them as it is universally known how many sides a triangle and a rectangle have essentially. At this distance, the sides will look similar so the sides didn’t play much of a factor in the definition of the image at this specific point. Our test images were looked at in an incredible fashion from left side to the accompanying right side of the image in order to effectively find the shape of the door accurately so the algorithm could essentially have

something to easily measure. We set the horizontal threshold to 30 pixels and in this we got the horizontal difference between shapes with the vertical alignment of two shapes. We then had our output detection image based on our passed in option in order to overlay our output image. We imputed some experimental states in our algorithm. We looked at adding a machine learning component to our entire program as we thought that specific feature extraction would be engaging in this type of project. To try to implement the specific neural network was quite an intensive process. At first we thought of implementing the neural networks from Real-time 2D and 3D door detection and state classification on a low-power device. I found that it was somewhat hard to For the specific yolo model that I tried to implement for the project there were so many necessary models that were needed in order to fully implement the model. The YOLOv3 is the third version of the object detection algorithm YOLO. This specific algorithm is a convolutional neural network to detect objects in real-time. A convolutional neural network is a type of neural network that This algorithm is so popular because of how speedy it is compared to other object detection algorithms. This algorithm has often been used in various applications such as traffic signals, people, parking meters, and animals. We also used another set of code for door detection. What was different in the code we used was how we initialized the Kernel operation using a input of 3x3 and for each pixel in the image for the new code for the door detection algorithm. The Sobel edge detector performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. We typically used this in order to correspond to specific edges within whatever test image that we are trying to analyze. When previously using the sobel operators for the previous MPs the implementation was quite complicated at first trying to find the proper edges for our given test image. As we started to understand the code, the implementation of the sobel operator became

easier because we were better able to filter out the edges as we did for this specific project. In order to make our analysis better we used this specific algorithm because it has the ability to detect edges very well. The Sobel operator is very similar to the Prewitt operator. It is also a derivative mask and is used for edge detection. Like Prewitt operator sobel operator is also used to detect two kinds of edges in an image, which is the vertical direction and the horizontal direction.The major difference is that in sobel operator the coefficients of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks. I included the difference between the Prewitt operator and sobel operator because I wanted to compare algorithms or operators that are very similar to each other.The major difference is that in sobel operator the coefficients of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

By using the Sobel edge detector this led to us using the Harris Corner Detection. The Harris Corner detector is an algorithm that is used mostly for corner detection as you can obviously tell from the name of the algorithm. With this algorithm we take the definition of the corner score into account with reference to direction directly. With this specific algorithm we use shifting patches for every 45 degree angle, and has been proved to be more accurate in distinguishing between edges and corners. Not all patches are considered equal. This is what this detector has the ability to accurately measure this phenomenon when all things are essentially equal. The Harris Detector gives an entirely mathematical approach for determining which case holds. We calculated that for nearly constant patches, this will be near 0. For every distinctive patch, this will be larger. Hence… we want patches where E(u,v) is large. In this detector we first computed the x and y derivatives of the image, then we computed the products of the derivatives at every

pixel. We then looked to compute the sums of the products of derivatives at each pixel, then we defined each pixel (x,y) as the matrix. Compute the response of the detector at each pixel. This is the equation that we used to define computing the response of the detector at each pixel, R= Det(H)-k(Trace(H))^2. From this we compute the nonmax suppression. In the python code we used the Sobel operators in our function for the Harris detector in order to obtain the x and y directions for the first series of calculations of the Harris detector. After this we chose to eliminate the negative values of the derivative of X and Y. This corresponds to compute the sums of the product of the derivative at each pixel part of the algorithm essentially. It actually does go along with the multiply by negative one part of the code that was created Each of these options were entirely optional and up to whomever wants to use whatever part of the code. By using many of these different algorithms to find out the specific squares in our algorithm, we are able to gain greater diversity in our results and analysis. If we didn't implement different types of algorithms and operators into our project we wouldn't be able to be as descriptive with the algorithm that we described because we wouldn't be able to compare the algorithm to other detection algorithms. and By using the sobel operator we could find edges much better on our 2-d images.

Here are some of the images that we used for our door detection algorithm. We had more than a 1000 test images we could use and usually the more the merrier when you have an abundance of data that can be used. In the project, I only have about four test images. I chose four test images because I thought that this would be enough to accurately show the effect of my detection algorithm to the best of my ability. When we first ran the code for the algorithm we got results that we didn't really expect.Our code didn't exactly pick up on the doors that we wanted it to detect in the image. I saw that there was a defect in the code so I chose images that had clear

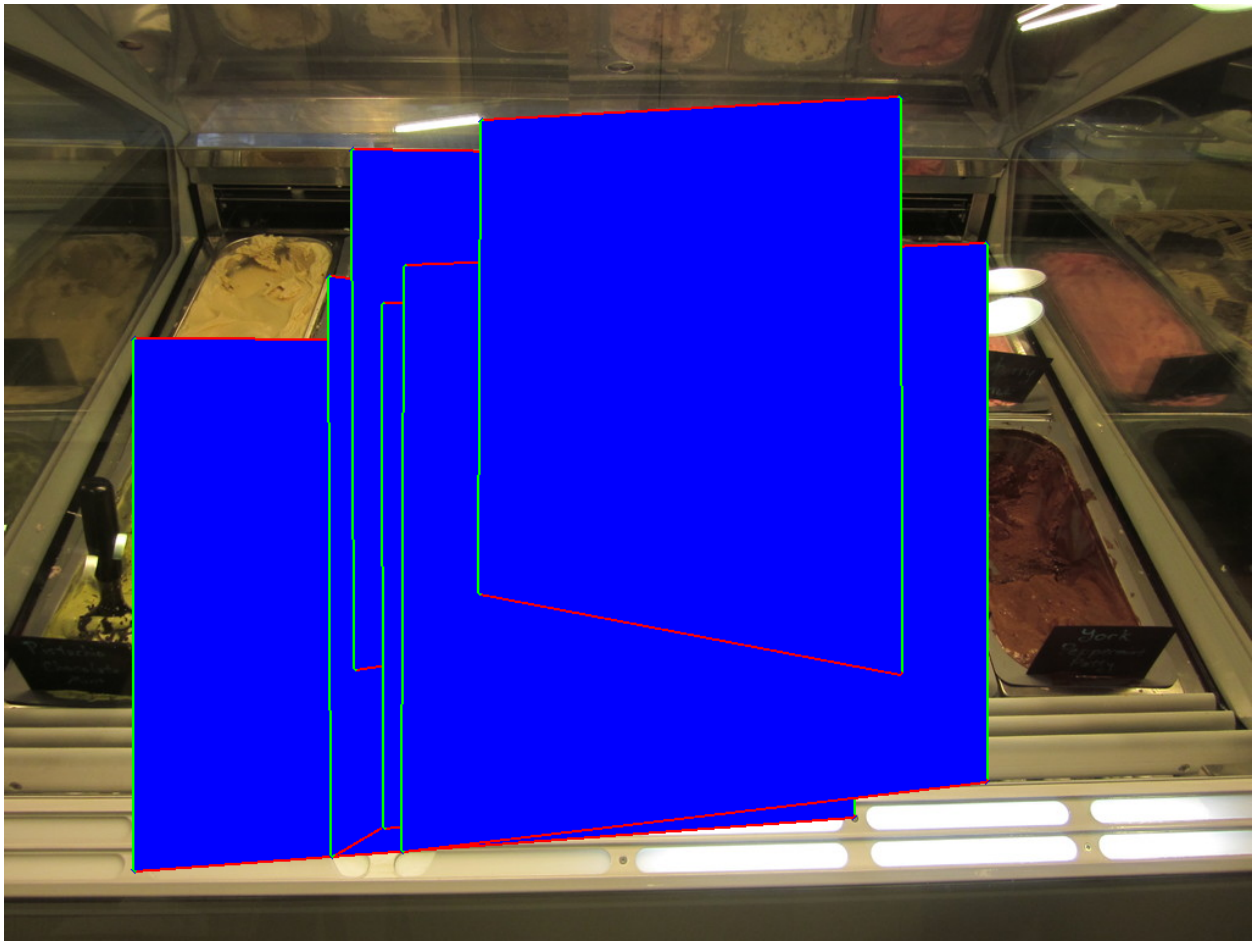
squares throughout the background. I knew that by adjusting certain lines in the code essentially adjusting the function for drawing corners and lines we were able to get a more accurate door detection model. We also resize the image because the image was just too large for the specific model that we employed. The image we first used essentially did the same thing with the results that were given in this ice cream store holder picture. The squares are quite hard to detect in the photo unfortunately. I took many long and grueling steps to try and figure out how the algorithm worked so I could properly adjust the squares appropriately. I thought by choosing pictures where the squares are larger in size the algorithm would pick it up much better. Since this first design did not work, I went to look and implement the sobel operator algorithm in order to better find the edges for my original image essentially. What we got was a regular highlighted image that pretty much lined up with our original expectations.

The Yolov3 was not totally implemented as the gpu/cpu configuration very much complicated the runtime and we were not able to get adequate results for our work. In order to not complicate the assignment we chose to withdraw the results for this specific implementation of this algorithm. We felt that the specific neural network was not altogether appropriate for this assignment. There were other neural networks from the paper that we took inspiration from this assignment from that used a machine learning component to analyze the properties of the doors in their dataset. The *PointNet* Neural network was implemented as a way to classify 3D object classification. In the specific paper they use this specific algorithm for both of the methods that they used in the entire project. This would have been an intuitive algorithm as this specific classification network would have taken parts *n points* as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification score for *m* classes. The segmentation network is an extension to the classification net. It

concatenates global and local features and outputs per point scores. *mlp stands* for multi-layer perceptron, the numbers in brackets are its layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in the classification net. This class is not some type of tutorial to add a deep learning/machine learning component to the analysis. This is why I did not delve into this realm as much. Keeping our project/report on the concepts that we mostly talked about in class was enough.

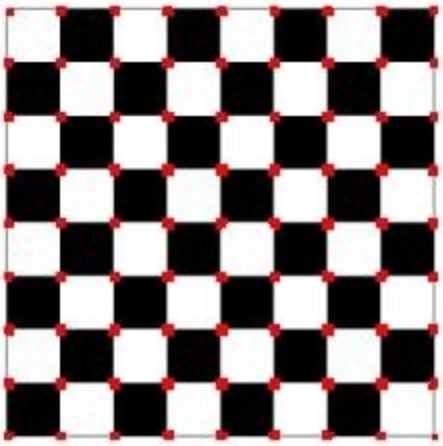
Caption: This is one of the first pictures in our dataset that we decided to use as a test image. This image was essentially thought of as not being as great of an image because it didn't have the

best defined squares that were good enough for detection. There were a few more test images that we used in order to portray a better implementation of the algorithm.

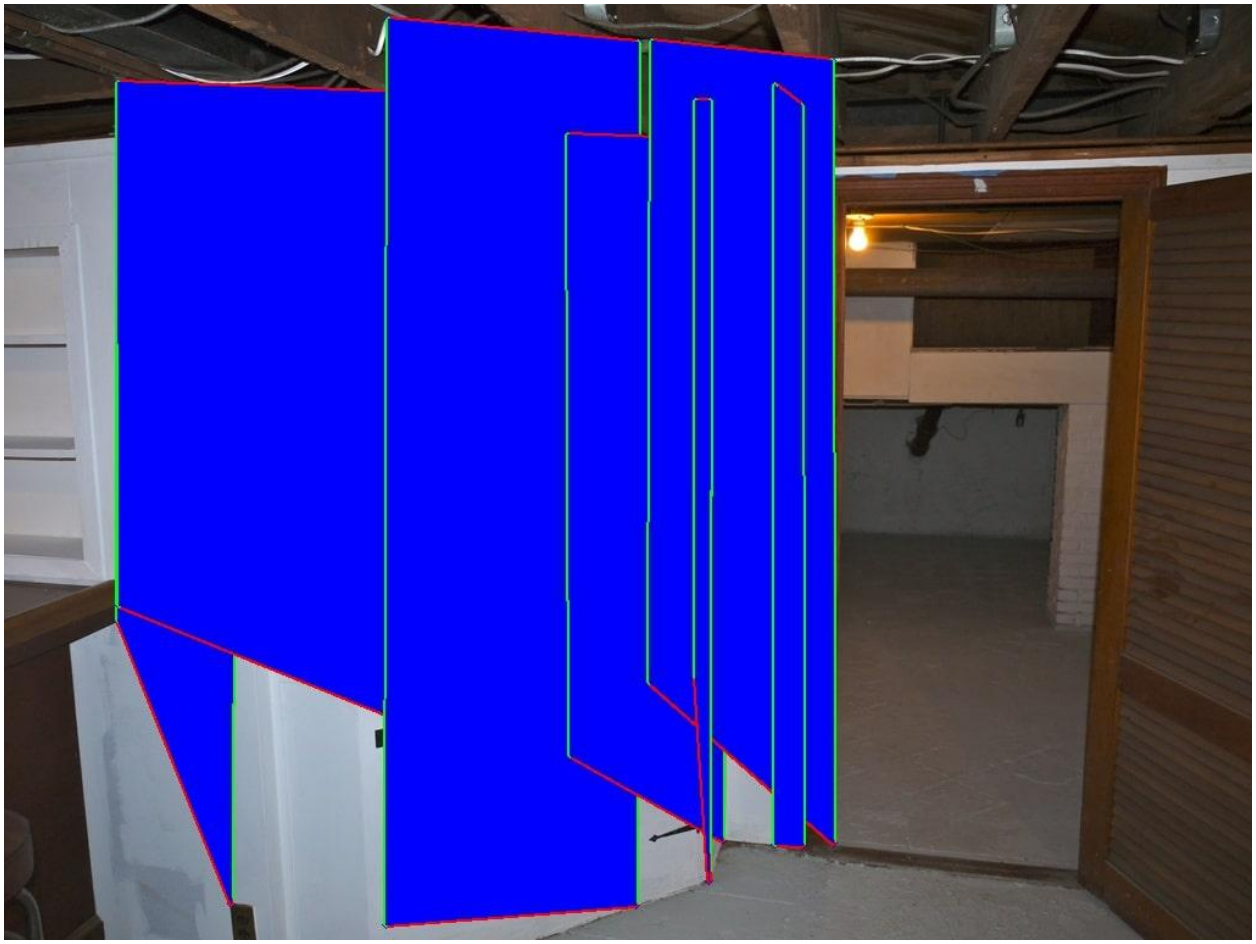


Caption: Here we find the results of implementing our own door detection algorithm. Now from what it seems the algorithm was not as accurate as it needed to be because it did not catch onto the squares that are in the image. This door detection algorithm is supposed to be very specific for certain types of images and this is what you often get when working with random images when your algorithm is supposed to specifically be for a certain dataset of images unfortunately. There was somewhat of a hard curtailing of trying to find the right code for the right type of image but one idea that was thought of was to totally scrape the images and find new ones that will exactly fit the code and the algorithm we were trying to implement through the code.

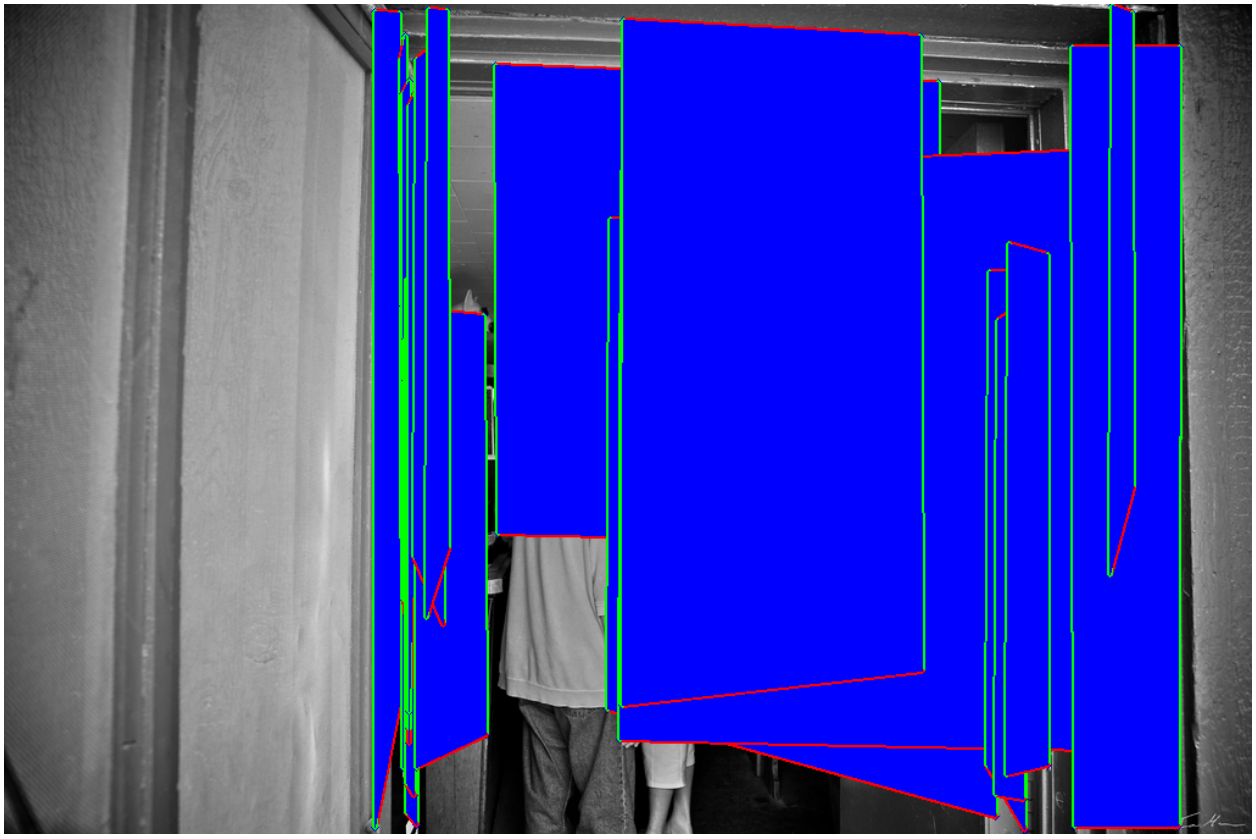
Caption:This is one of the images that we chose for the implementation of our algorithm but we had some of the same problems that we did with the previous image again unfortunately. In the image we can see how defined the squares are and the shapes in general are in the photo. This image just may not be right for the code and there are adjustments in the code that may need to be made in order to correctly detect what essentially needs to be detected. The open images dataset is probably too opaque of a dataset in general for our first door detection algorithm that we created. Even the adjustments are not enough to totally make the implementation as seamless and as proven as possible.



I ran the Harris corner detection algorithm on this image and here we get much more defined classifications for our squares compared to the other images that we ran the code on. This is because the squares are quite clearly much more defined for this specific image compared to the other images.



Caption: These are the results that we received and they again did not turn out as well as we expected them. We found some of the same figures in our current image as we did before. The sobel detection operator seemed like it would have worked much better in trying to find the real undercutting edges and lines in the particular image. The entire project would have worked better if we had taken from multiple datasets in order to ensure a better and diverse output/result essentially. This



The most important part of this project I believe needed improving was the writing of the code part. The code could have been better manipulated throughout the duration of the project much better than it should have. There are some techniques or algorithms I believe need to be properly researched for a certain amount of time before they are implemented. The class would benefit

from the ph.d. or the grad student having a TA section where he or she can help us understand many of the algorithms that are needed to be successful in computer vision. I feel that the actual implementation of the algorithms that I chose could heavily be improved on and some of the reason why the implementation may have not been as successful is because of the lack of the entire understanding of the general algorithm. This is outside of the scope of this project but I believe we should have focused more on face-tracking and much of the algorithms behind this technology. This is how many of the robots, autonomous vehicles, and machines in general learn how to navigate and see in the technology that we have today. If you wanted to really get the students prepared for industry we would be constantly trying to understand face-tracking and its algorithm. Everyone would come out of their program with computer vision engineer job offers.

This course was quite informational and engaging for both the student and I hope the professor as well. I very much learned from many of the mathematical explanations of the algorithms and I wish that there was more of an emphasis on this aspect of the class. It would be quite good for us if we delved into trying to see ways we could mathematically manipulate these algorithms and see the different outcomes we can get. This would seem to be a good assignment in order for us students to learn research for computer vision and research techniques in general for students who may want to pursue a ph.d. Or the researcher route. I feel that research is half of the computer vision field since much of this field is so new and much of the applications of this field have still yet to be discovered.The class would benefit from the ph.d. or the grad student having a TA section where he or she can help us understand many of the

References

http://stanford.edu/~rqi/pointnet/

https://link.springer.com/article/10.1007/s42452-021-04588-3

https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

https://automaticaddison.com/how-the-sobel-operator-works/

https://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

http://www.cs.cmu.edu/~16385/s17/Slides/6.2_Harris_Corner_Detector.pdf