2D to 3D image visualization and filter application

Alexander Raytchev

100938966

Abstract

With no more than a look, living beings can process the 3D world around them without second thought. This is not the case for computers, and is what computer vision looks to explore, exactly how can we translate real world information for a machine to accurately understand. This project will aim to see how easily a 2D image can be processed such that it would be able to receive a 3D representation. Furthermore, various filters will be added to explore if altering an image could lead a path towards an easier conversion.

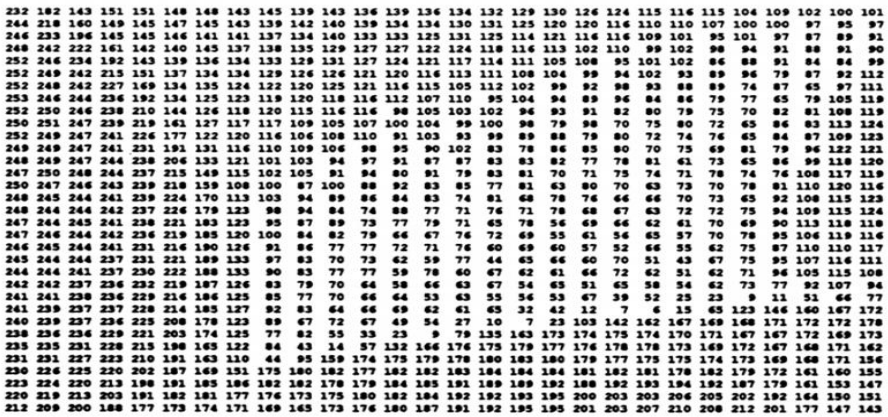
Introduction

The application is a GUI that intakes any photo that's in the current directory and from there lets the user apply any of the 10+ filters that were implemented. Once the photo has been altered to the users liking the image can then be opened as a 3d graph with either matplotlib or the more sophisticated plotly. The first challenge was implementing a fully functional GUI that could have its state altered as the user goes through the program. What was challenging about this is that the state of the image had to be continuously preserved and passed along from function to function so that if the user wanted to graph the image with modifications they could do so. Secondly it was getting all dozen implemented filter algorithms (blur, edge detections, canny etc.) to be compatible with each other. The issue here was that the numbers in the array had to be consistent, it had to first always maintain the same size. Secondly the types RGB to BGR to grayscale that OpenCV naturally posses made it so that that I had to find a way of keeping all arrays consistent with each other. Lastly, and probably the biggest challenge was making the project proposal with no idea if it would work. I had never worked with or heard of computer vision before January and the proposal was based of the first two lectures. There was no guarantee that I would have been able to store an image into an array with x,y,z values that could later be plotted. The challenge in that sense was figuring out everything for myself.

Background

The background of this project comes from the very first lecture.

Digital Image



Seeing this image instantly gave me the idea that this could have potential to be something more. Something that other students and other people in general would see and be able to recognize that it is indeed the building blocks of an image. Another inspiration that came to me was that the application could be used as a teaching tool for computer vision. Because I struggled in the beginning of this course having a tool that would show you in real time what each filter does seemed like a good idea for myself and for potential future students who take the course. For example, I have an image called saltandpeper.jpg, if you select the image then press on the median filter the noise vanishes, this could be very beneficial to other visual learners. That is how my project had two focuses, create a 3d representation and get a lot of filters working. Then later investigate if the filters help make better 3d images.

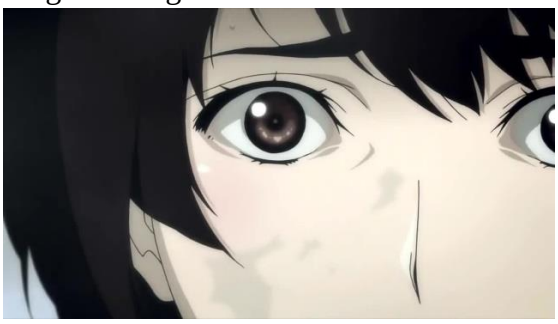
Approach

The first step that needed to be done in order to get the approach to work was to resize every image with cv2.resize() and get them all into the same grayscale format. Otherwise there were issues with the rgb bgr gray defaults that are present. For example, an image that was not in grayscale format could not receive a lot of the edge detection filters.

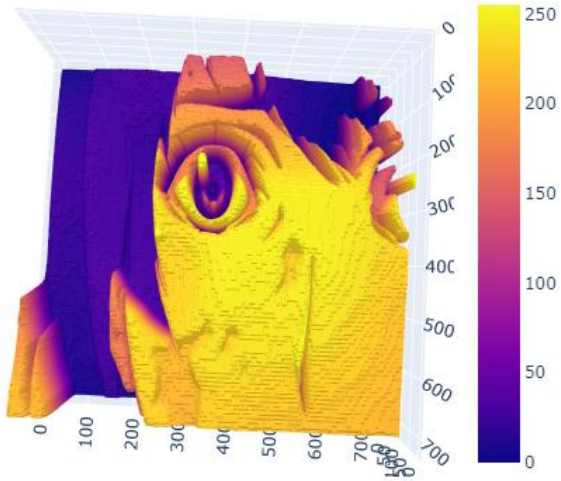
Next up were all the filters. The data of the image had to be passed into the functions which then called OpenCV prebuilt functions. Besides, Gray scaling, resizing, reading and passing the matrices no other algorithms were used. The application is more of a fun learning GUI that tends to borrow already prebuild filters from the OpenCV library. It uses average blur, Gaussian Blur, median blur, bilateral blur, laplacian, canny, sobel horizontal and sobel vertical edge detection, adaptive mean threshold and finally adaptive Gaussian threshold.

Results

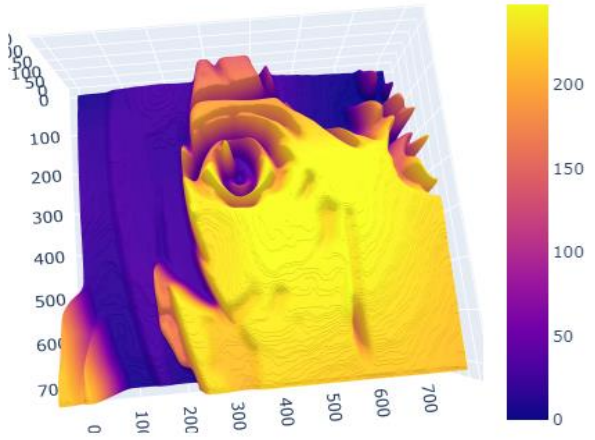
Original image



3D plot with plotly



3D plot with plotly after smoothing filters are applied



Here we can see in the results the difference in the intensity of the pixels when a smoothing filter is applied. In a normal view the image looks blurry but with the software you can clearly see the difference and how the computer would interpret the two images. And this point is key, computer vision is about what the computer can see. The only limitation was that I was not able to properly export the data the way I wanted to. Originally, I wanted to first get the data into excel, however it became apparent that each value in image was a very large array. This hindered plans to potentially port the image vectors anywhere else. If I was to recreate this project, I would have prioritized this feature and implement my work from the start to have a chance at exporting the values.

List of Work:

All Work was done by me as I was alone and did not have a group.

GitHub repository: https://github.com/raytchev3004/comp4102

References

OpenCV official documentation for basic algorithms
https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

Programming manual provided on cuLearn for various functions with PIL, array, and cv2
Programming with python, Jan Erik Solem

Mathplotlib documentation for graph
https://matplotlib.org/3.2.1/contents.html

Plotly documentation for graph
https://plotly.com/python/3d-axes/

Tkinter documentation for GUI help
https://docs.python.org/3/library/tk.html?

stackoverflow for fixing code that would not compile