Eric Gomez, Peyton Slater
Computer Vision
December 16th, 2023

Implementation of Edge Detection and Object Detection Method

Introduction:

The main goal of this project is to create a method for edge detection, then to identify the five largest objects of the image and display them to the user. Secondary objectives include the implementation of other edge detection methods and comparing their results with our design.

Implementation:

For our edge detection method, we took a couple of steps to ensure that the result is as good as we can possibly get. First, we decided that by smoothing the image with a 3x3 gaussian kernel would be good because it would remove a lot of the noisy edges from the final result. We then applied a simple dark-channel-prior to get rid of hazy and foggy elements that would appear in the background of the image. Then, we convert the image to grayscale to remove most of the color that would interfere with the contour detection that we used. To find the edges, we used a simple sobel operator in the x and y gradient. After finding the gradient magnitude of the sobel operators, we further filtered the gradient with a threshold of 75 , which removed a lot of the less prominent edges at the cost of losing some detail in the final image. Now that we had the edges, we used cv2's findContours operation to find the contours of the edges array (The boundaries of the objects). We then filtered out the contours with a threshold of '3', which gets rid of contours that are not thicker than 3 pixels. This removed less prominent objects. We then displayed the edge detected image to the user and sent the contour image to our object detection algorithm.

For the object detection method, we mainly utilized built-in cv2 methods to detect the objects. We first start by converting the image to a binary image. We then use cv2 to find the contours again, as well as sort those contours by area. We then just simply draw the 5 largest contours with cv2 drawContours and show them to the user.

Evaluations:

To properly evaluate our implementation, we decided it would be best to test our results on the Berkeley Segmentation Dataset 500 (BDSD500). This dataset contains 200 test images that we can test our algorithm with. We also implemented the following edge detection methods: Canny, Sobel, Kirsch, Lapplachan of Gaussian, PreWitt, and Holisted Edge Detection(HED). Note that HED is a deep-learning based method, and we decided to use a pre-trained model to save time. We used multiple algorithms to gather more qualitative results to compare our results with. For some quantitative results, we also tracked the runtime of each algorithm to see how efficiently the code runs.

Conclusion:

Overall, the results for our method shows success in certain areas and shortcomings in other areas. We saw that the edge detection struggled in images with a lot of detail. This is because our edge detection algorithm attempts to filter out less prominent edges, but when the image is almost entirely composed of those small edges as a result most of the image does not get detected by the algorithm. Another shortcoming we noticed was images with backgrounds that blend well with object boundaries. This is more a problem with the contour detection piece, because once again we attempt to get rid of less prominent object boundaries so that we could clearly discern what is or isn't an object. The issue is that when the boundary edge is similar in color to the actual background, the thickness of the edge can be too small for our algorithm to pick up.

As for successes,

References:

https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

https://github.com/s9xie/hed?tab=readme-ov-file

https://www-sciencedirect-com.aurarialibrary.idm.oclc.org/science/article/pii/S092523122200248X?via%3Dihub#s0010

A New Method for Edge Detection Under Hazy Environment in Computer Vision | SpringerLink (oclc.org)

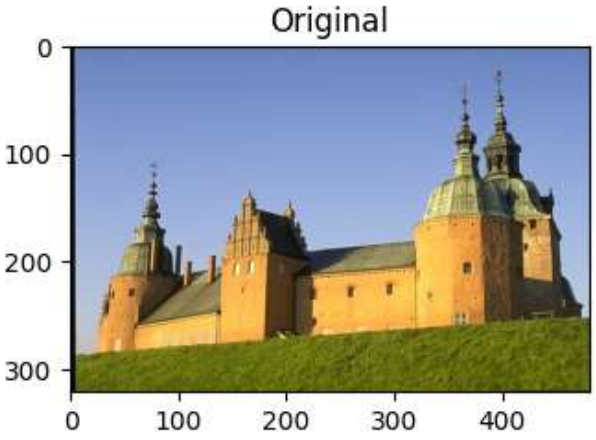
https://www.sciencedirect.com/science/article/pii/S2212017312003l2X?via%3Dihub

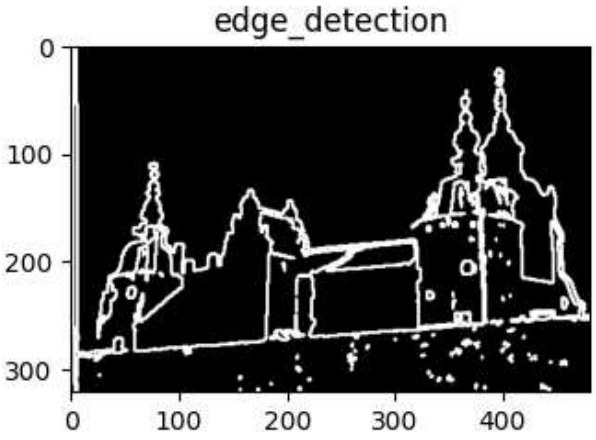
https://www-sciencedirect-com.aurarialibrary.idm.oclc.org/science/article/pii/S0925231222008141?via%3Dihub

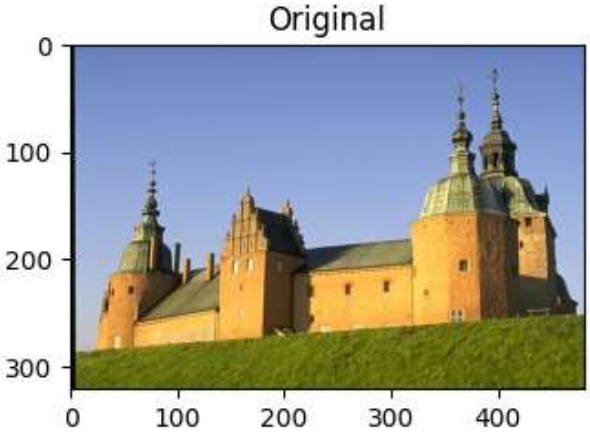
https://docs.opencv.org/4.x/

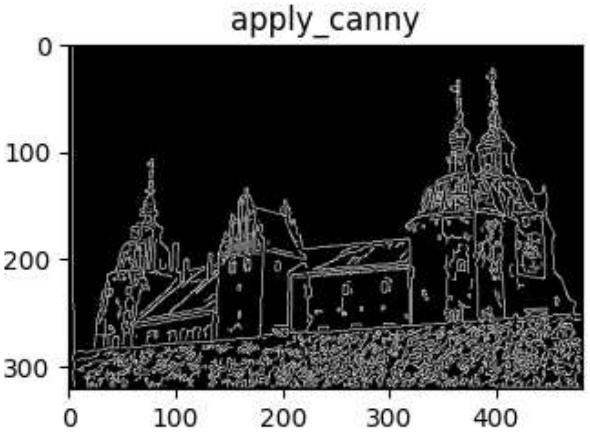
https://pillow.readthedocs.io/en/stable/

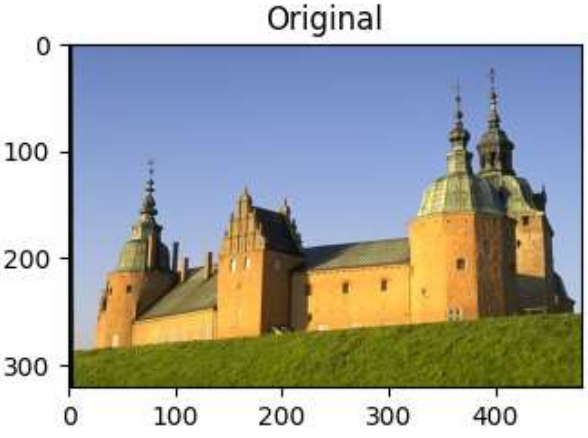
https://docs.python.org/3/library/tk.html

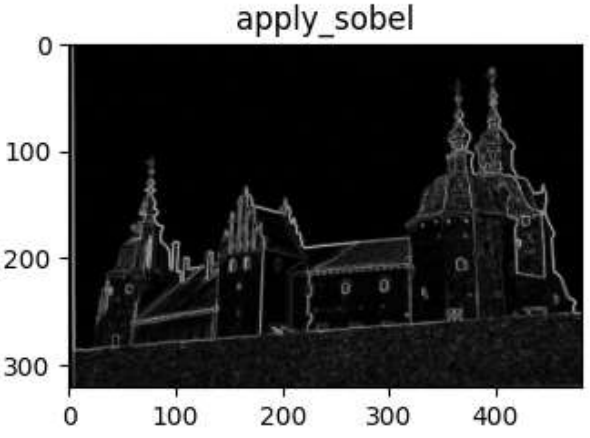

Original
0
100
200
300
0
100
200
300
400

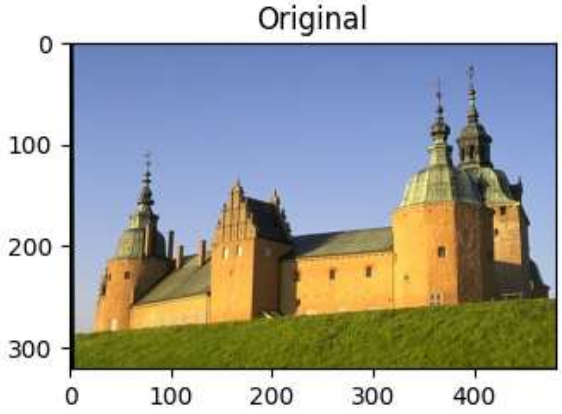

edge_detection
0
100
200
300
0
100
200
300
400

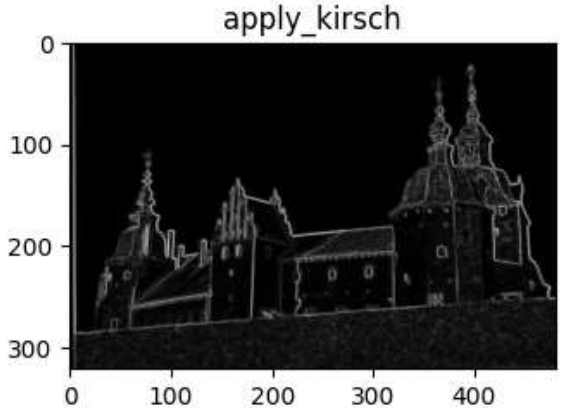

Original
0
100
200
300
0
100
200
300
400

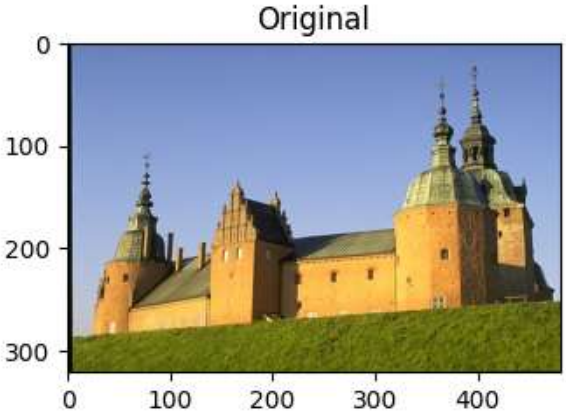

apply_canny
0
100
200
300
0
100
200
300
400

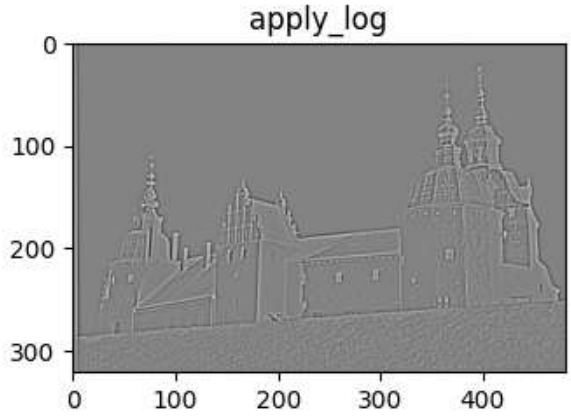

Original
0
100
200
300
0
100
200
300
400

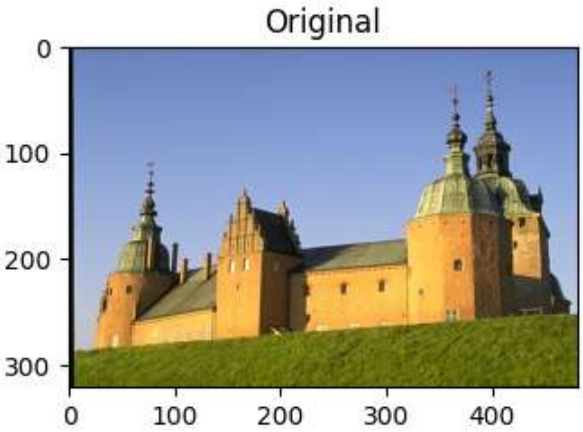

apply_sobel
0
100
200
300
0
100
200
300
400

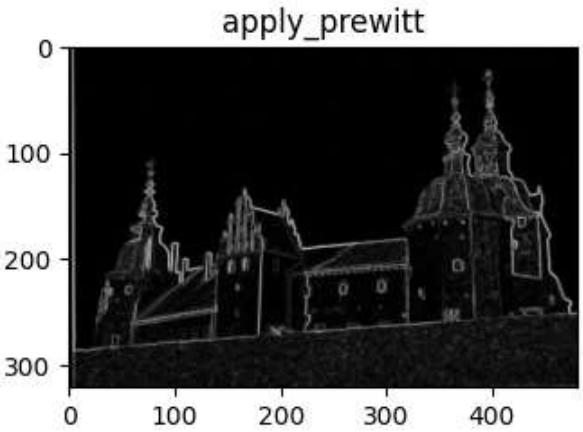

Original
0
100
200
300
0
100
200
300
400

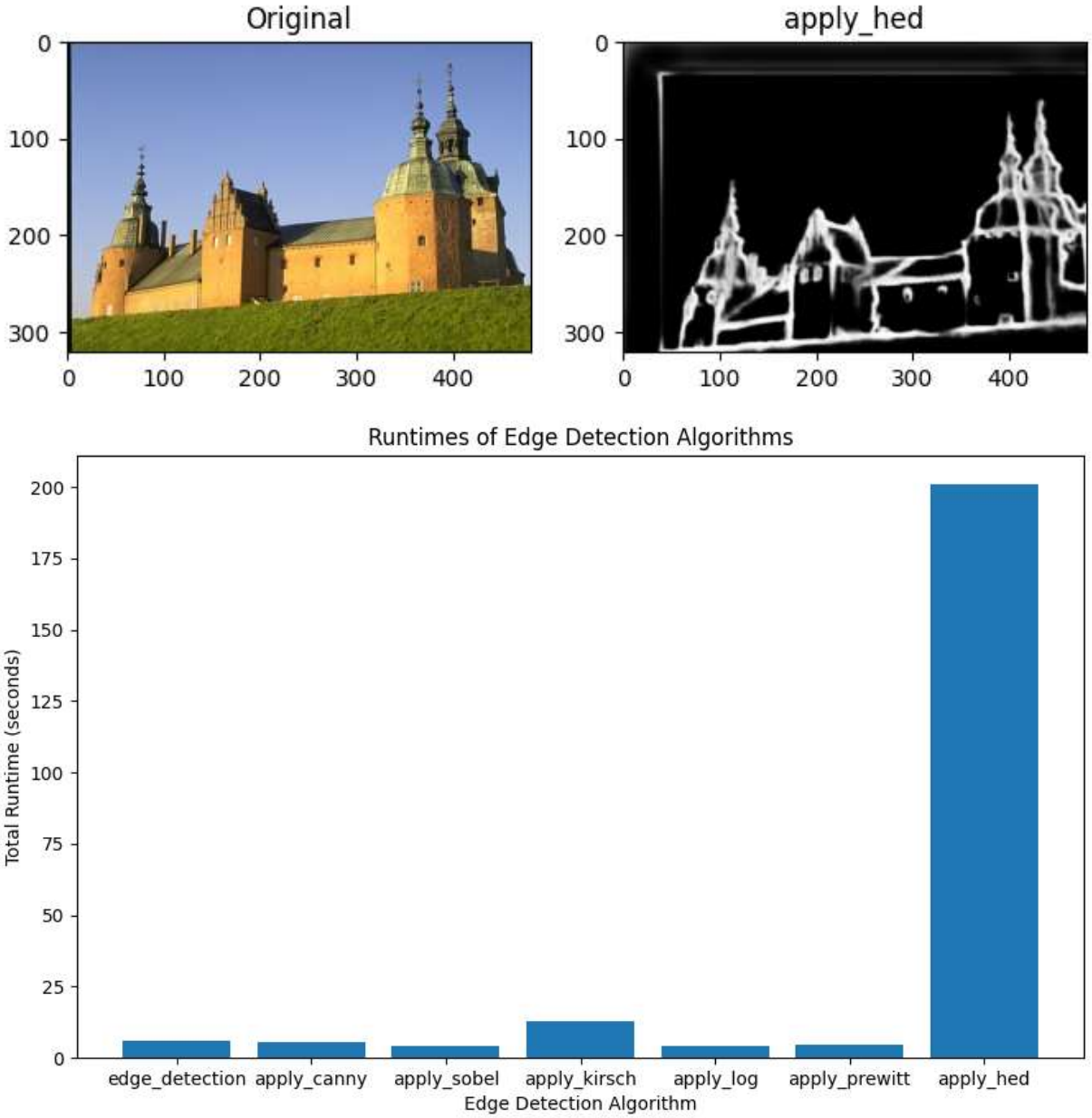

apply_kirsch
0
100
200
300
0
100
200
300
400


Original
0
100
200
300
0
100
200
300
400


apply_log
0
100
200
300
0
100
200
300
400


Original
0
100
200
300
0
100
200
300
400


apply_prewitt
0
100
200
300
0
100
200
300
400



edge_detection - Total Runtime: 5.9083 seconds
apply_canny - Total Runtime: 5.3249 seconds
apply_sobel - Total Runtime: 4.2744 seconds
apply_kirsch - Total Runtime: 12.9864 seconds
apply_log - Total Runtime: 3.9354 seconds
apply_prewitt - Total Runtime: 4.3604 seconds
apply_hed - Total Runtime: 201.0340 seconds