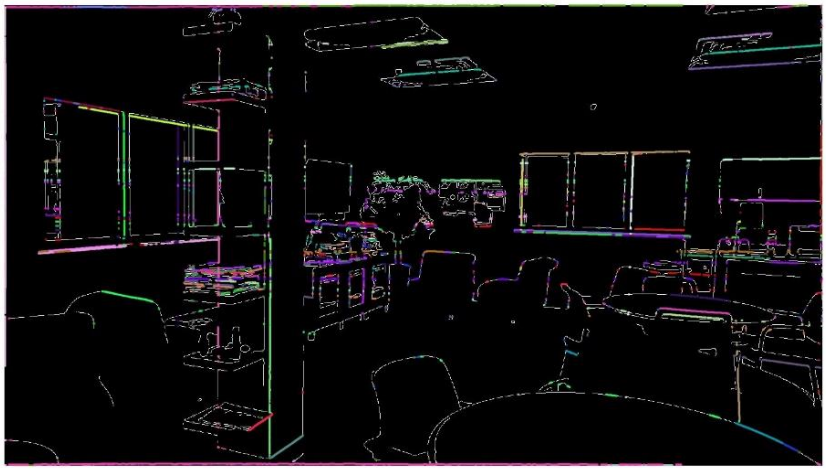
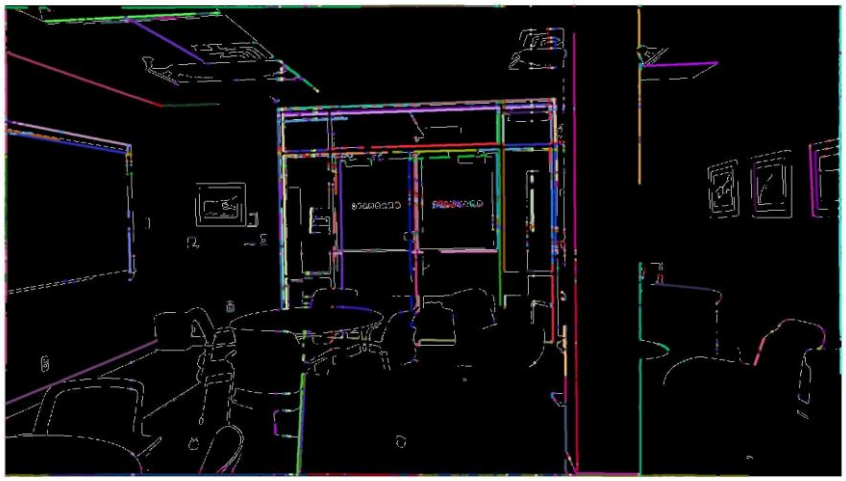
MANJU SHIVDASAN – U1133278
manju.shivdasan@utah.edu

1. The objective of this question was to obtain an image that only highlights the edges of the objects within the image.
   The function CV_1.m outputs the image for this question.
   Algorithm Logic:
   The algorithm is designed to pick 2 random points P & Q that are that are less than 100 distances apart. Then it picks another random point c, to check if it lies on the line PQ. I used a combination of cross product to find the distance of from PQ and then and used dot product to check if that point lies between the point PQ. If the point c lies on line PQ and if the algorithm were able to find 50 such points then the algorithm plots a line, else ignores it. This logic is iterated for almost 2000 times to get sufficient number of edges that is plotted in the output using random colors.

Matlab file:
The heart of the matlab file lies in the while loop that does most of the calculation to get edges in the LINE_SET. The initial for loop is used to extract the EDGE_SET from the images that is used as the base to construct the output image.

Output images for all 3 inputs:

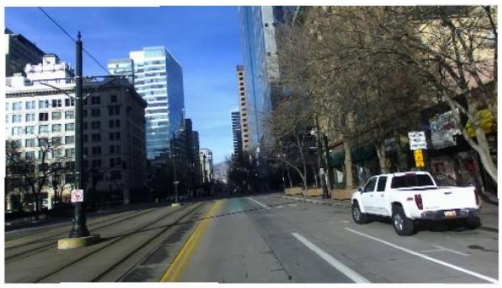




2. The objective of this question was to make the sky look white and everything else black. We had to use 3 different images to test our algorithm.
   The function CV_1.m outputs the image for this question.

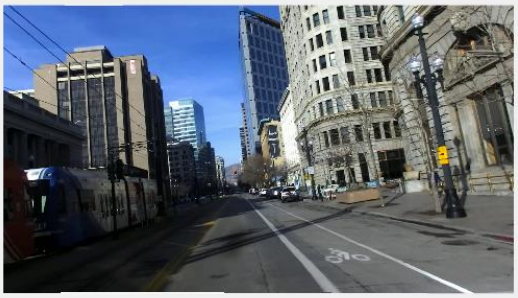
   Algorithm Logic:
   The algorithm logic is pretty simple. If the R, G, B component of the given image lies within a particular value, those pixels are given a white color and the rest of them are assigned black color. Although the algorithm works pretty well on all images, each image has its own specific r,g,b value that corresponds to the element sky. Thus, to get the desired output, the r,g,b threshold values are independently define for each input image.
   Matlab Logic:
   The main part of the matlab code is the for loops that changes the pixels either to white or black. Also the components min and max values are equally important as they make or break the code.

a.



b.

c.



3. The objective of this question was to get the disparity map when the left and right images were given.
   The function CV_1.m outputs the image for this question.

   Algorithm Logic:
   The pixels in left image is always shifted by a small value of 'd' when compared to the right image. This change in 'd' is only applicable on the x-axis and the y coordinate remains constant. I have written an algorithm that utilizes the concept of NCC to get the final disparity image. 2 pixels Patch 1 and Patch 2 are considered as perfect match if the NCC score is 1 and the worst match if the NCC score is 0. This is achieved by varying the 'disp' value between 1 to 50. For all the values from 1 to 50, the highest value is chosen as the bestDisparity. This is done on each and every pixel value the get the final disparity image.
   Matlab Logic:
   The heart of this code lies in the for loop of the code where all the calculations are done, and the disparity image is formed. It utilizes the NCC formula given in the question to find the currNCC and check which disp value will give the bestNCC value.

a.

b.

c.

Citation:

1. Mathematical logics from https://www.wikipedia.org/ .
2. Matlab and syntax functions from https://www.mathworks.com/.