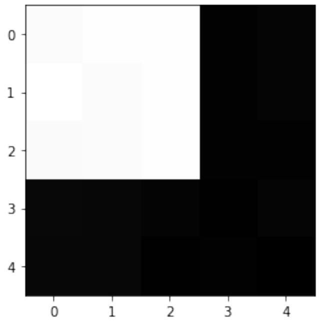
Computer Vision Assignment – 2 **Name: Ishu Goyal**

Part -A

Github link: https://github.com/ishu98goyal/Computer-Vision-CSC8830/tree/main/Assignment%202

For Part A, a patch was created.

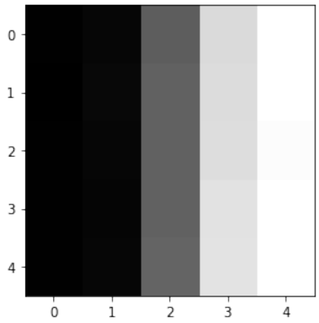
Patch creation process is shown in Jupyter Notebook: Patch_Creation.ipynb and it creates an image called corner.jpg



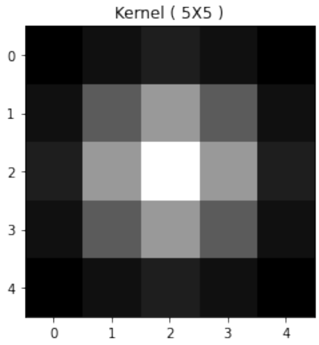
Q1: Code: GitHub repo → Part A → Canny Edge detection.ipynb

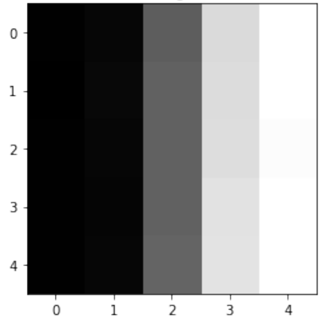
Outputs:

Image Patch:

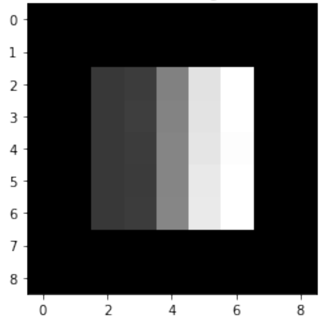


Blurred Patch:

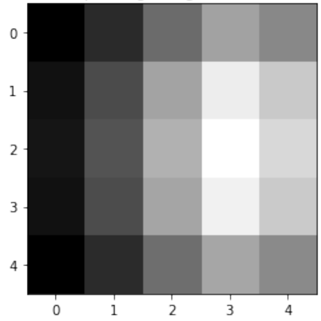




img

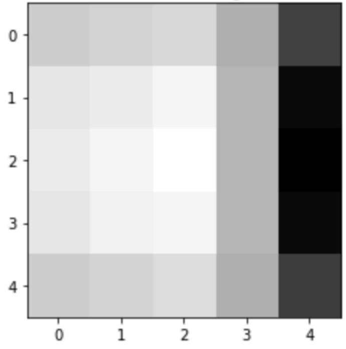


Padded img

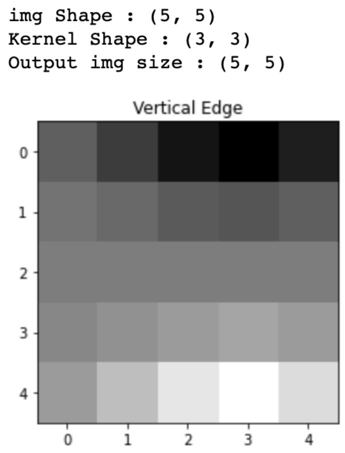


Output img using 5X5 Kernel

```
img Shape : (5, 5)
Kernel Shape : (3, 3)
Output img size : (5, 5)
```



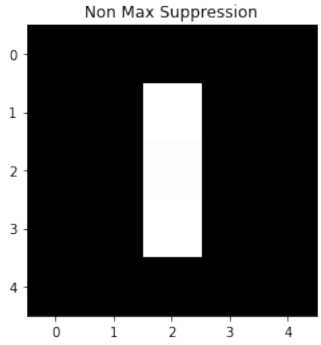
Horizontal Edge



```
img Shape : (5, 5)
Kernel Shape : (3, 3)
Output img size : (5, 5)
```

Vertical Edge

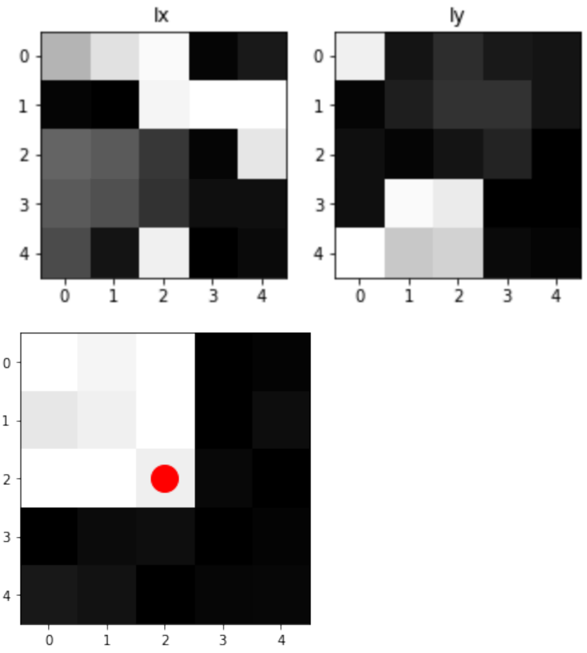
Final Image:



Non Max Suppression

Q2 Code: GitHub repo → Part A → Harris Corner Detection.ipynb

Output:



Ix

Iy

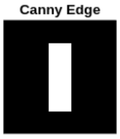
PART B

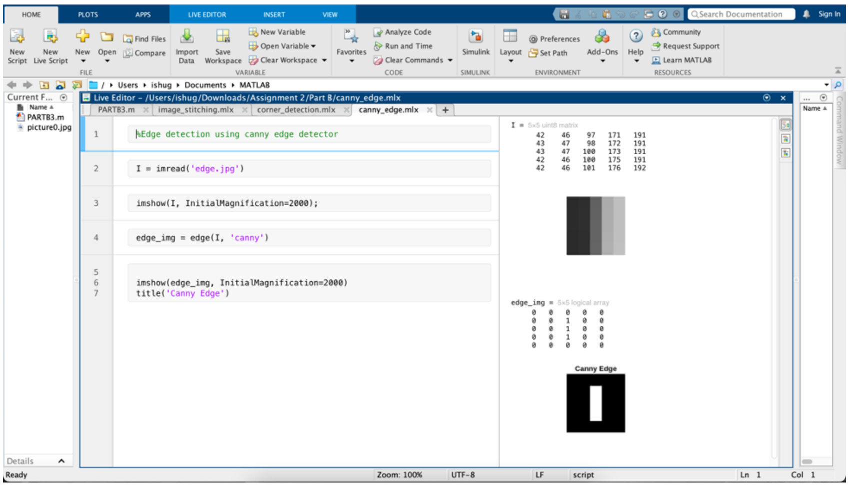
Q3: Code: GitHub repo → Part B → canny_edge.mlx (Github)

Output:

```
I = 5×5 uint8 matrix
    42    46    97   171   191
    43    47    98   172   191
    43    47   100   173   191
    42    46   100   175   191
    42    46   101   176   192
```

```
edge_img = 5×5 logical array
   0   0   0   0   0
   0   0   1   0   0
   0   0   1   0   0
   0   0   1   0   0
   0   0   0   0   0
```



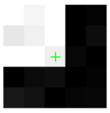


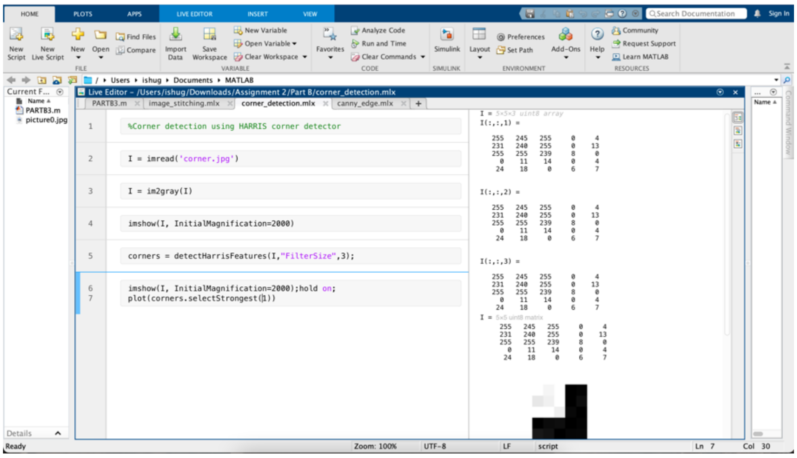
```
%Edge detection using canny edge detector

I = imread('edge.jpg')

imshow(I, InitialMagnification=2000);

edge_img = edge(I, 'canny')


imshow(edge_img, InitialMagnification=2000)
title('Canny Edge')
```

```
I = 5×5 uint8 matrix
    42    46    97   171   191
    43    47    98   172   191
    43    47   100   173   191
    42    46   100   175   191
    42    46   101   176   192

edge_img = 5×5 logical array
   0   0   0   0   0
   0   0   1   0   0
   0   0   1   0   0
   0   0   1   0   0
   0   0   0   0   0
```
Canny Edge

Q4 Code: GitHub repo → Part B → corner_detection.mlx

```
I = 5×5×3 uint8 array
I(:,:,1) =

   255   245   255     0     4
   231   240   255     0    13
   255   255   239     8     0
     0    11    14     0     4
    24    18     0     6     7


I(:,:,2) =

   255   245   255     0     4
   231   240   255     0    13
   255   255   239     8     0
     0    11    14     0     4
    24    18     0     6     7


I(:,:,3) =

   255   245   255     0     4
   231   240   255     0    13
   255   255   239     8     0
     0    11    14     0     4
    24    18     0     6     7
I = 5×5 uint8 matrix
   255   245   255     0     4
   231   240   255     0    13
   255   255   239     8     0
     0    11    14     0     4
    24    18     0     6     7
```

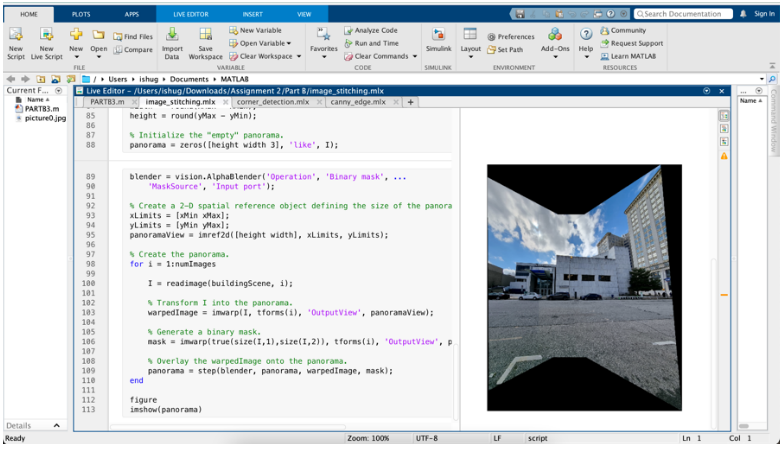




```
%Corner detection using HARRIS corner detector

I = imread('corner.jpg')

I = im2gray(I)

imshow(I, InitialMagnification=2000)

corners = detectHarrisFeatures(I,"FilterSize",3);

imshow(I, InitialMagnification=2000);hold on;
plot(corners.selectStrongest(1))
```

I = 5×5×3 uint8 array
I(:,:,1) =
255 245 255 0 4
231 240 255 0 13
255 255 239 8 0
0 11 14 0 4
24 18 0 6 7
I(:,:,2) =
255 245 255 0 4
231 240 255 0 13
255 255 239 8 0
0 11 14 0 4
24 18 0 6 7
I(:,:,3) =
255 245 255 0 4
231 240 255 0 13
255 255 239 8 0
0 11 14 0 4
24 18 0 6 7
I = 5×5 uint8 matrix
255 245 255 0 4
231 240 255 0 13
255 255 239 8 0
0 11 14 0 4
24 18 0 6 7

Q5: Code: GitHub repo → Part B → image_stitching.mlx ()

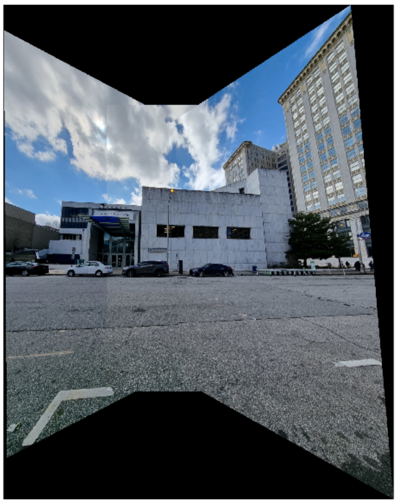


Live Editor – /Users/ishug/Downloads/Assignment 2/Part B/image_stitching.mlx

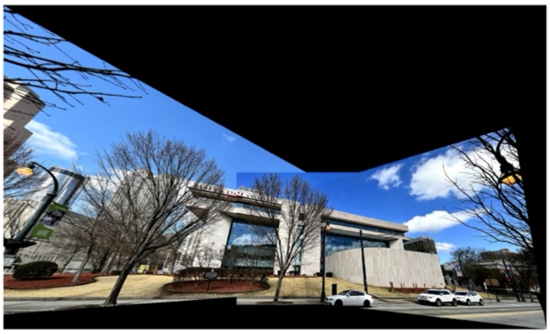
```
height = round(yMax - yMin);

% Initialize the "empty" panorama.
panorama = zeros([height width 3], 'like', I);

blender = vision.AlphaBlender('Operation', 'Binary mask', ...
    'MaskSource', 'Input port');

% Create a 2-D spatial reference object defining the size of the panora
xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

% Create the panorama.
for i = 1:numImages

    I = readimage(buildingScene, i);

    % Transform I into the panorama.
    warpedImage = imwarp(I, tforms(i), 'OutputView', panoramaView);

    % Generate a binary mask.
    mask = imwarp(true(size(I,1),size(I,2)), tforms(i), 'OutputView', p

    % Overlay the warpedImage onto the panorama.
    panorama = step(blender, panorama, warpedImage, mask);
end

figure
imshow(panorama)
```

Output:

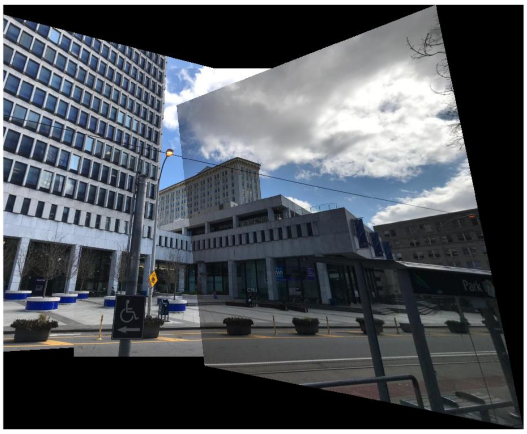
1. Arts and Humanitarian Building



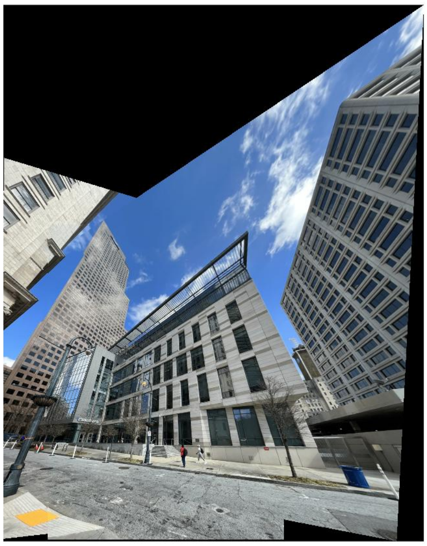
2. Normal Building



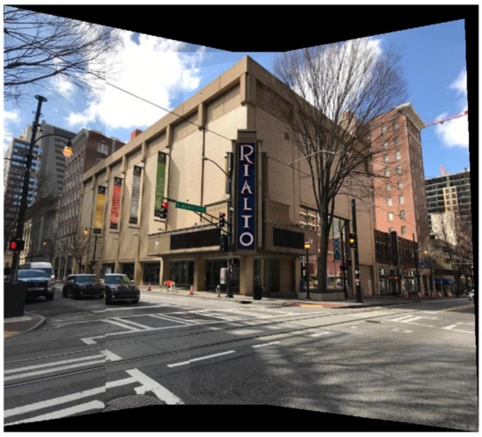
3. 25 Park Place



4. Law School Building



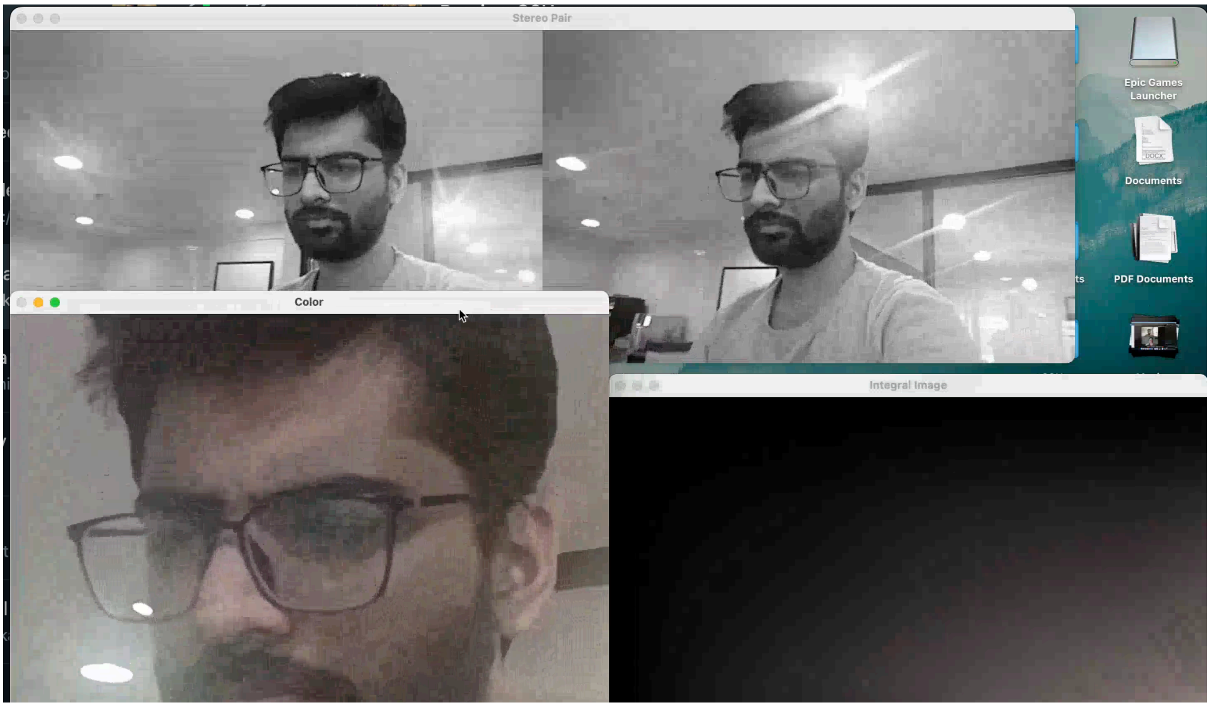
5. Rialto Building



Part C

Q6: Code + Video : GitHub repo → Part C → integral_images.py (Github)

Output:



Q7: Code + Video: GitHub repo → Part C → panaroma.py

Output: