CSc 8830: CV Assignment-2 Solutions Report

Capture a 10 sec video footage using a camera of your choice. The footage should be taken with the camera in hand and you need to pan the camera slightly from left-right or right-left during the 10 sec duration. For all the images, operate at grayscale unless otherwise specified:

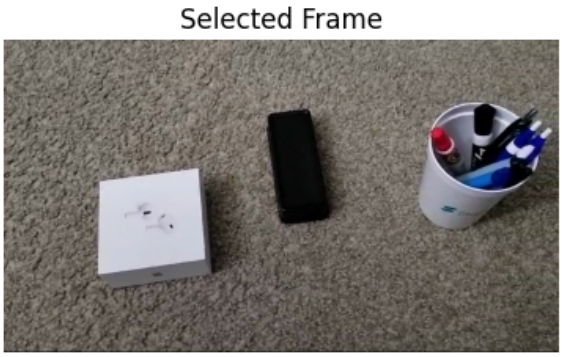
A video of 10 sec was recorded from a mobile camera with 1030p resolution and 30FPS frame rate.

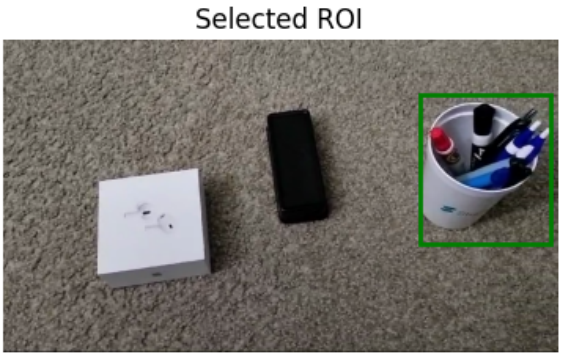
1. Pick any image frame from the 10 sec video footage. Pick a region of interest in the image making sure there is an EDGE in that region. Pick a 5 x 5 image patch in that region that constitutes the edge. Perform the steps of CANNY EDGE DETECTION manually and note the pixels that correspond to the EDGE. Compare the outcome with MATLAB or OpenCV or DepthAI's Canny edge detection function.

Steps Involved:

1. **Pick an Image Frame:** Select a frame from the video footage that contains the region of interest with an edge.
2. **Select Region of Interest (ROI):** Identify the area in the image where the edge is located. Ensure that this region contains a clear edge that you want to detect.
3. **Pick a 5x5 Image Patch:** Within the ROI, select a 5x5 image patch that includes the edge. This patch will be used for manual edge detection.
4. **Convert to Grayscale:** Convert the selected image patch to grayscale. Edge detection algorithms typically work on grayscale images.
5. **Compute Image Gradient:** Use a simple gradient operator (e.g., Sobel operator) to compute the gradient magnitude and direction at each pixel in the image patch. This step highlights areas of rapid intensity change, which often correspond to edges.
6. **Apply Non-Maximum Suppression:** Suppress non-maximum pixels along the gradient direction. Only keep pixels that have the maximum gradient magnitude in their local neighborhood.
7. **Apply Hysteresis Thresholding:** Apply hysteresis thresholding to the gradient magnitude image to identify strong and weak edges. Pixels with gradient magnitudes above a high threshold are considered strong edges, while pixels between high and low thresholds are considered weak edges.
8. **Edge Tracking by Hysteresis:** Perform edge tracking by hysteresis to link weak edges to strong edges. Starting from strong edge pixels, follow the gradient direction and connect neighboring weak edge pixels that are part of the same edge.

9. **Note Detected Edge Pixels:** Note down the pixels that correspond to the detected edge in the 5x5 image patch.
10. **Compare with OpenCV:** Use OpenCV Canny edge detection function to perform edge detection on the same image patch. Compare the outcome with the manually detected edge pixels to assess the accuracy and effectiveness of the manual process.





2. Pick any image frame from the 10 sec video footage. Pick a region of interest in the image making sure there is a CORNER in that region. Pick a 5 x 5 image patch in that region that constitutes the edge. Perform the steps of HARRIS CORNER DETECTION manually and note the pixels that correspond to the CORNER. Compare the outcome with MATLAB or OpenCV or DepthAI's Harris corner detection function.

Steps Involved:

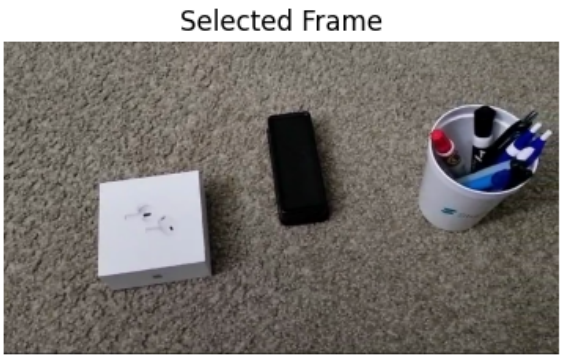
1. **Pick an Image Frame:** Select a frame from the video footage that contains the region of interest with a corner.
2. **Select Region of Interest (ROI):** Identify the area in the image where the corner is located. Ensure that this region contains a clear corner that you want to detect.
3. **Pick a 5x5 Image Patch:** Within the ROI, select a 5x5 image patch that includes the corner. This patch will be used for manual corner detection.
4. **Convert to Grayscale:** Convert the selected image patch to grayscale. Corner detection algorithms typically work on grayscale images.
5. **Compute Gradient:** Compute the gradient of the grayscale image patch using a gradient operator such as the Sobel operator. This step helps in identifying areas of rapid intensity change.
6. **Compute Harris Corner Response:** Compute the Harris corner response function for each pixel in the image patch using the formula:

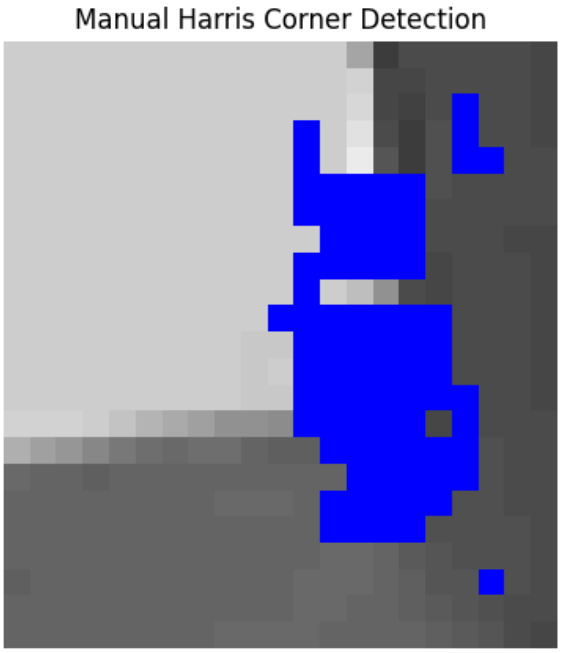
   $$R = \det(M) - k \cdot (\text{trace}(M))^2$$

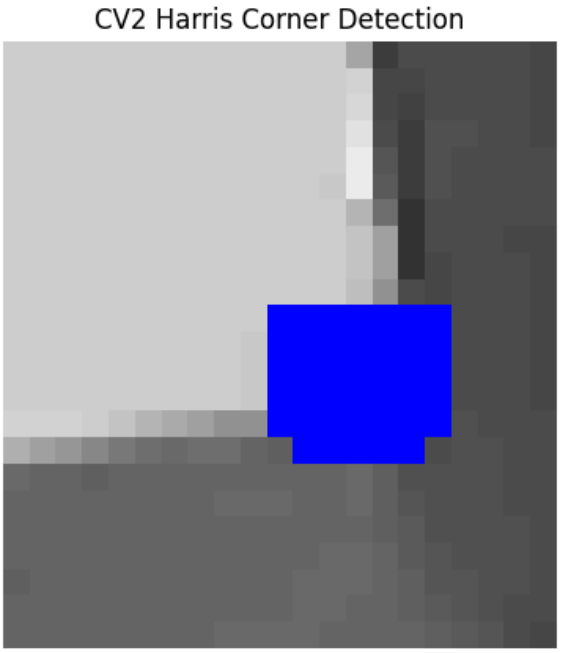
   Where:

   - R is the Harris corner response.
   - M is the gradient matrix of the image patch.
   - $\det(M)$ is the determinant of M.
   - $\text{trace}(M)$ is the trace of M.
   - k is an empirical constant typically set to a small value like 0.04.

7. **Threshold Harris Response:** Apply a threshold to the computed Harris corner response to identify potential corner pixels. Pixels with a response above a certain threshold are considered potential corners.
8. **Perform Non-Maximum Suppression:** Perform non-maximum suppression to eliminate non-maximum responses and identify local maxima as corner candidates.
9. **Note Detected Corner Pixels:** Note down the pixels that correspond to the detected corners in the 5x5 image patch.
10. **Compare with OpenCV:** Use OpenCV Harris corner detection function to perform corner detection on the same image patch. Compare the outcome with the manually detected corner pixels to assess the accuracy and effectiveness of the manual process.







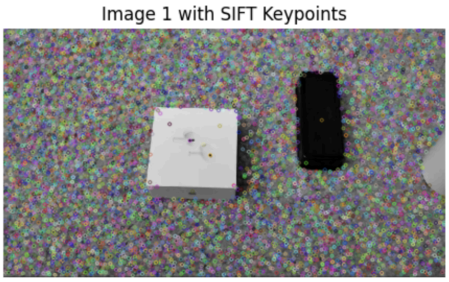
3. Consider an image pair from your footage where the images are separated by at least 2 seconds. Also ensure there is at least some overlap of scenes in the two images.

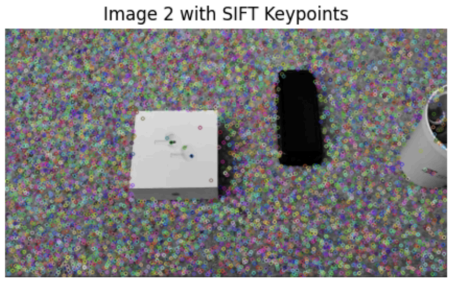
a. **Pick a pixel (super-pixel patch as discussed in class) on image 1 and a corresponding pixel ((super-pixel patch as discussed in class)) on image 2 (the pixel on image 2 that corresponds to the same object area on image 1). Compute the SIFT feature for each of these 2 patches. Compute the sum of squared difference (SSD) value between the SIFT vector for these two pixels. Use MATLAB or Python or C++ implementation -- The MATLAB code for SIFT feature extraction and matching can be downloaded from here: https://www.cs.ubc.ca/~lowe/keypoints/ (Please first read the ReadMe document in the folder to find instructions to execute the code).**
b. **Compute the Homography matrix between these two images using MATLAB or Python or C++ implementation. Compute its inverse.**

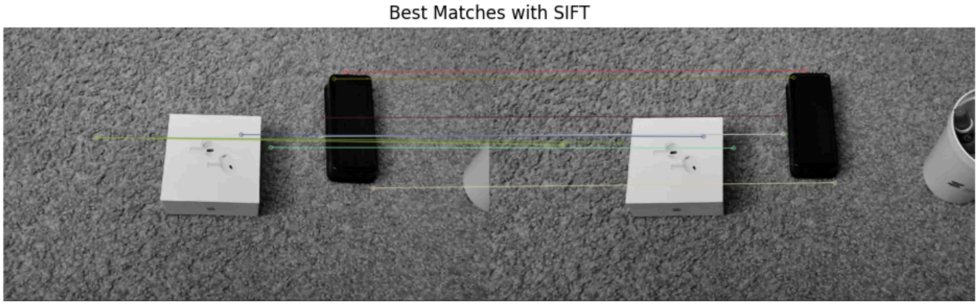
1. **Select Image Pair:** Choose two images from your footage that are separated by at least 2

seconds and have some overlap of scenes.

2. **Select Corresponding Pixels:** Pick a pixel (super-pixel patch) in the first image and its corresponding pixel in the second image. Ensure that the corresponding pixels represent the same object or area in the scene.
3. **Compute SIFT Features:** Use MATLAB, Python (with OpenCV), or C++ (with OpenCV) to compute the SIFT features for the selected super-pixel patches in both images. This involves detecting keypoints and computing descriptors using the SIFT algorithm.
4. **Calculate SSD:** Compute the sum of squared differences (SSD) between the SIFT descriptors of the two super-pixel patches. This quantifies the dissimilarity between the two patches based on their feature vectors.
5. **Compute Homography Matrix:** Use MATLAB, Python (with OpenCV), or C++ (with OpenCV) to compute the homography matrix that transforms points from the coordinate system of the first image to the coordinate system of the second image. This can be done using feature matching and the RANSAC algorithm to estimate the homography matrix robustly.
6. **Compute Inverse Homography Matrix:** Once the homography matrix is computed, calculate its inverse. This will allow you to transform points from the coordinate system of the second image back to the coordinate system of the first image.
7. **Perform the Computations:** Implement the above steps in your chosen programming language (Python). Use existing libraries or implementations for SIFT feature extraction and homography estimation if available.








```
Sum of Squared Difference (SSD) between SIFT vectors: 1425.0
```

```
Two frames with at least 2-second difference were selected.
Homography Matrix:
[[-8.87860525e+00 -1.07937893e+01  4.08755449e+03]
 [-1.21440619e+00 -1.04243070e+01  1.60691053e+03]
 [-5.97664703e-03 -1.74197323e-02  1.00000000e+00]]

Inverse of Homography Matrix:
[[-7.51990911e-02  2.58589073e-01 -1.08149122e+02]
 [ 3.59117641e-02 -6.65678901e-02 -3.98226494e+01]
 [ 1.76134892e-04  3.85900794e-04 -3.40069019e-01]]
```

4. Implement an application that will compute and display the INTEGRAL image feed along with the RGB feed. You cannot use a built-in function such as "output = integral_image(input)"

Steps Involved:

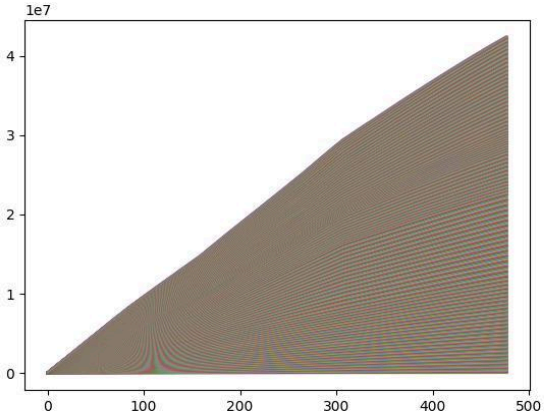
1. **Read RGB Frames:** Capture RGB frames from a camera or load a video file.
2. **Convert to Grayscale:** Convert each RGB frame to grayscale. Integral images are typically computed from grayscale images.
3. **Compute Integral Image:** For each grayscale frame, compute the integral image. The integral image at a pixel (x,y) contains the sum of all pixel values in the rectangle from (0,0) to (x,y). You can compute the integral image using the following formula:

   $$integral\_image(x,y)=grayscale(x,y)+integral\_image(x-1,y)+integral\_image(x,y-1)-integral\_image(x-1,y-1)$$

   $$integral\_image(x,y)=grayscale(x,y)+integral\_image(x-1,y)+integral\_image(x,y-1)-integral\_image(x-1,y-1)$$*Make sure to handle edge cases appropriately.*

4. **Display Integral Image Feed:** Display the computed integral image feed along with the original RGB feed. You can use any suitable visualization method to display both feeds simultaneously, such as creating a split-screen display or overlaying the integral image on top of the RGB image.

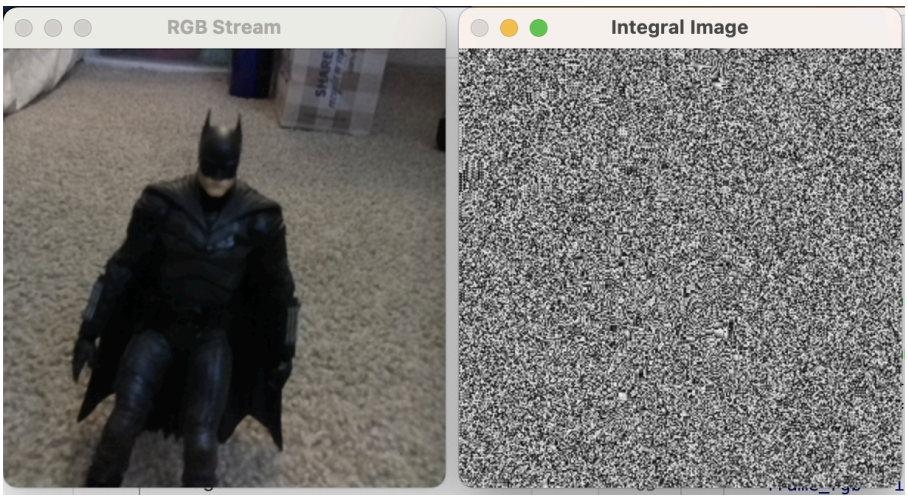
For Image:



integral_image.txt

```
102 200 294 384 472 561 656 765 897 1036 1173 1290 1402 1515 1614 1706 1800 1899 2012 2130 2253 2385
2504 2618 2730 2835 2933 3049 3190 3335 3472 3596 3723 3843 3964 4087 4220 4360 4494 4624 4745 4866
4990 5120 5249 5373 5500 5629 5758 5883 5996 6101 6204 6310 6425 6545 6668 6791 6915 7045 7177 7295
7426 7557 7689 7816 7942 8065 8172 8269 8370 8472 8576 8683 8799 8920 9047 9180 9284 9368 9422 9468
9528 9608 9710 9827 9976 10138 10284 10425 10551 10669 10795 10935 11084 11226 11353 11477 11601
11724 11848 11965 12076 12179 12278 12371 12460 12549 12640 12730 12819 12908 13010 13124 13236 13340
13447 13560 13671 13780 13884 13994 14104 14220 14330 14436 14555 14674 14793 14912 15034 15168 15290
15394 15483 15587 15680 15772 15873 15961 16051 16158 16288 16424 16547 16664 16769 16884 17002 17119
17238 17356 17475 17590 17696 17790 17867 17942 18016 18088 18170 18280 18403 18512 18610 18716 18838
18970 19115 19246 19364 19475 19578 19677 19786 19911 20029 20139 20236 20333 20441 20557 20668 20765
20894 21029 21140 21231 21322 21424 21536 21638 21720 21816 21912 22007 22115 22232 22358 22477 22607
22748 22895 23036 23172 23294 23386 23488 23577 23675 23769 23858 23942 24027 24127 24232 24358 24471
24585 24690 24786 24887 24992 25102 25223 25357 25501 25646 25779 25900 26025 26156 26287 26429 26564
26676 26768 26870 26986 27117 27246 27371 27480 27590 27702 27813 27911 27993 28095 28185 28284 28396
28512 28623 28713 28803 28905 29006 29098 29157 29224 29354 29503 29646 29783 29917 30044 30164 30279
30389 30496 30603 30704 30818 30928 31023 31107 31182 31287 31422 31561 31692 31805 31907 32003 32105
32217 32336 32454 32574 32696 32813 32918 33012 33097 33197 33319 33450 33586 33725 33862 33979 34097
34216 34336 34443 34565 34696 34833 34982 35117 35228 35326 35423 35529 35647 35773 35903 36036 36171
36308 36445 36581 36687 36776 36869 36986 37113 37252 37396 37535 37676 37805 37924 38045 38167 38283
38414 38561 38728 38874 39014 39139 39265 39406 39552 39680 39786 39888 39996 40095 40186 40281 40380
40490 40590 40684 40792 40931 41071 41215 41337 41452 41582 41734 41895 42041 42172 42328 42479 42619
42763 42893 43011 43130 43247 43365 43479 43592 43703 43815 43922 44033 44171 44303 44446 44599 44754
44900 45032 45165 45312 45454 45591 45722 45840 45950 46063 46181 46305 46431 46560 46686 46793 46873
46948 47025 47116 47222 47330 47452 47587 47731 47882 48010 48129 48266 48385 48504 48618 48747 48874
49009 49151 49283 49400 49517 49644 49769 49887 49985 50080 50181 50273 50365 50470 50592 50721 50848
50974 51098 51218 51331 51443 51546 51630 51738 51854 51978 52106 52236 52364 52491 52629 52773 52899
53001 53083 53171 53272 53374 53478 53582 53683 53781 53878 53987 54096 54216 54344 54474 54607 54738
54869 54991 55099 55219 55342 55460 55550 55639 55751 55882 56017 56139 56257 56374 56493 56604 56723
56849 56980 57119 57265 57407 57540 57641 57726 57832 57945 58049 58145 58239 58341 58453 58576 58713
58852 58986 59117 59240 59346 59450 59559 59689 59821 59938 60044 60130 60219 60315 60432 60560 60685
```

For Video Stream:



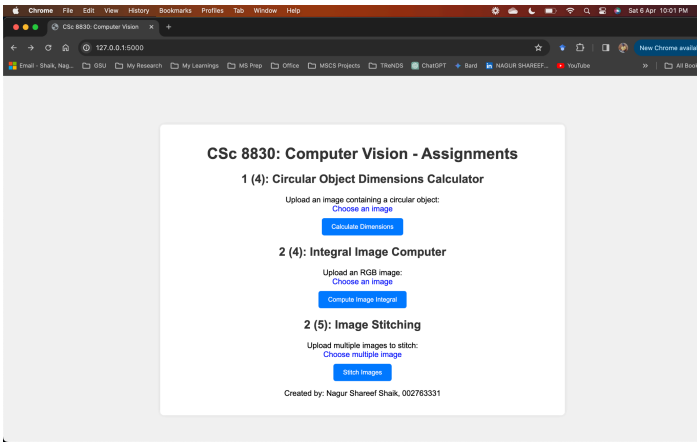
5. Implement the image stitching for a 360 degree panoramic output. This should function in real-time. You can use any type of features. You can use built-in libraries/tools provided by OpenCV or DepthAI API. You cannot use any built-in function that does output = image_stitch(image1, image2). You are supposed to implement the image_stitch() function

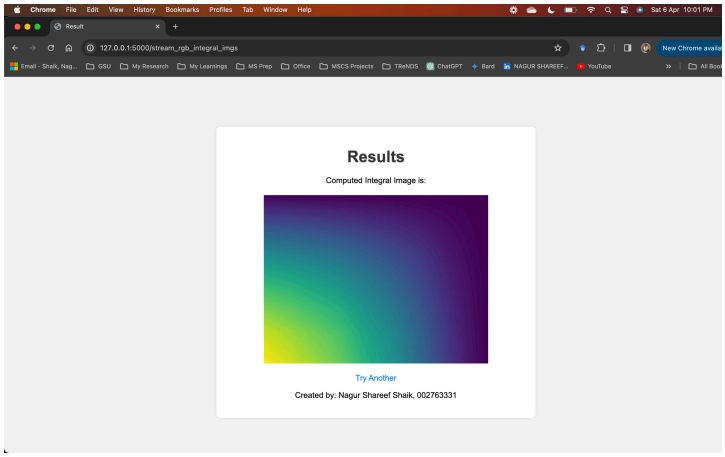
Steps Involved:

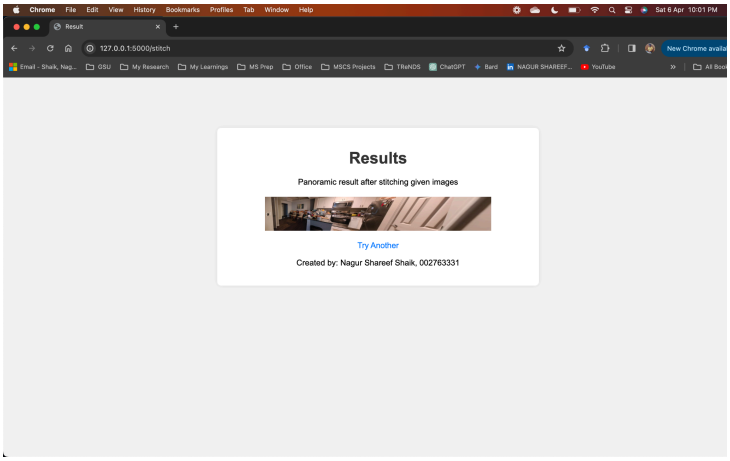
1. Feature Detection and Matching:
    - Detect keypoints and extract descriptors from each image. You can use feature detectors like SIFT, SURF, ORB, etc.
    - Match keypoints between consecutive pairs of images. You can use feature matching algorithms like brute force matching, FLANN matching, etc.
2. Find Homography:
    - Use the matched keypoints to estimate the homography transformation between consecutive pairs of images. The homography represents the perspective transformation between two images.
    - Apply RANSAC (Random Sample Consensus) algorithm for robust estimation of the homography matrix.
3. Warp Images:
    - Warp the second image (right image) onto the first image (left image) using the estimated homography matrix. This aligns the two images properly.
4. Blend Images:
    - Blend the warped image with the first image to seamlessly merge them together. You can use techniques like feathering, alpha blending, etc., to avoid visible seams.
5. Repeat Steps for All Images:
    - Repeat the above steps for all pairs of consecutive images until all images are stitched together.
6. Finalize Panorama:
    - Combine all the stitched images to create the final panorama. This can be achieved by blending overlapping regions and cropping the result to remove any black borders.

Stitched Image:

6. Integrate the applications developed for problems 4 and 5 with the web application developed in Assignment 1 problem 4*







Github Link:
https://github.com/ShaikNagurShareef/CSc8830-Computer-Vision/tree/main/Assignment-2