PART-1

Corners have been detected using Harris corner detection. This is implemented using the function **harris().** Harris is applied after converting the image into grayscale from RGB.

harris():

Inorder to get the gradients of the image along X and Y direction sobelX and sobelY is used. Then Gaussian filter is applied on this to smooth the image. Ixx, Ixy, Iyy are calculated for getting the r values (i.e for Harris getting harris corners) using the formula rval = IxxG*IyyG - IxyG*IxyG - k*(IxxG + IyyG)*(IxxG + IyyG)

Then I normalized each r value in order to get each r value between 0 and 1. Only those points are considered which are having threshold value above 0.615 inorder to detect corners which are above the threshold.

Now non maximal suppression is applied on each of the corner points obtained by taking the patch size of 9x9 grid. Non maximal suppression is when a pixel is picked if and only if the pixel is having the value greater than all the values inside the patch. These points after applying non maximal suppression is considered as corner points and are returned.

PART-2

Matching and proximity is implemented using three main functions **matching(), method1Affine(), getxybar()**

matching():

This method takes corner points of both the adjacent images. Now I am considering a blob radius of 10 around each corner point (let it be c1) in the first image then for that location in the second image I will check which points fall inside the circle (let it be c2,c3,c4) (This step ensures the proximity). Now take a patch of size 9x9 keeping c1 as center and a patch of size 9x9 keeping c2 as center. Now calculate the Sum of Square Differences (SSD) between the two. Do this process for (c1,c3) and (c1,c4) as well. Now pick the 3 minimum among these pairs based on SSD value. These will be the best 3 matching pairs.

method1Affine():

This method calculates the affine matrix by taking the best 3 matching pairs as input.

getxybar():

This method gives the transformation of a point from one coordinate system to another when applied with affine transform matrix. Here we will get

the points in frame2 that need to be fit into the stitched image corresponding in frame1 coordinate system.

PART-3

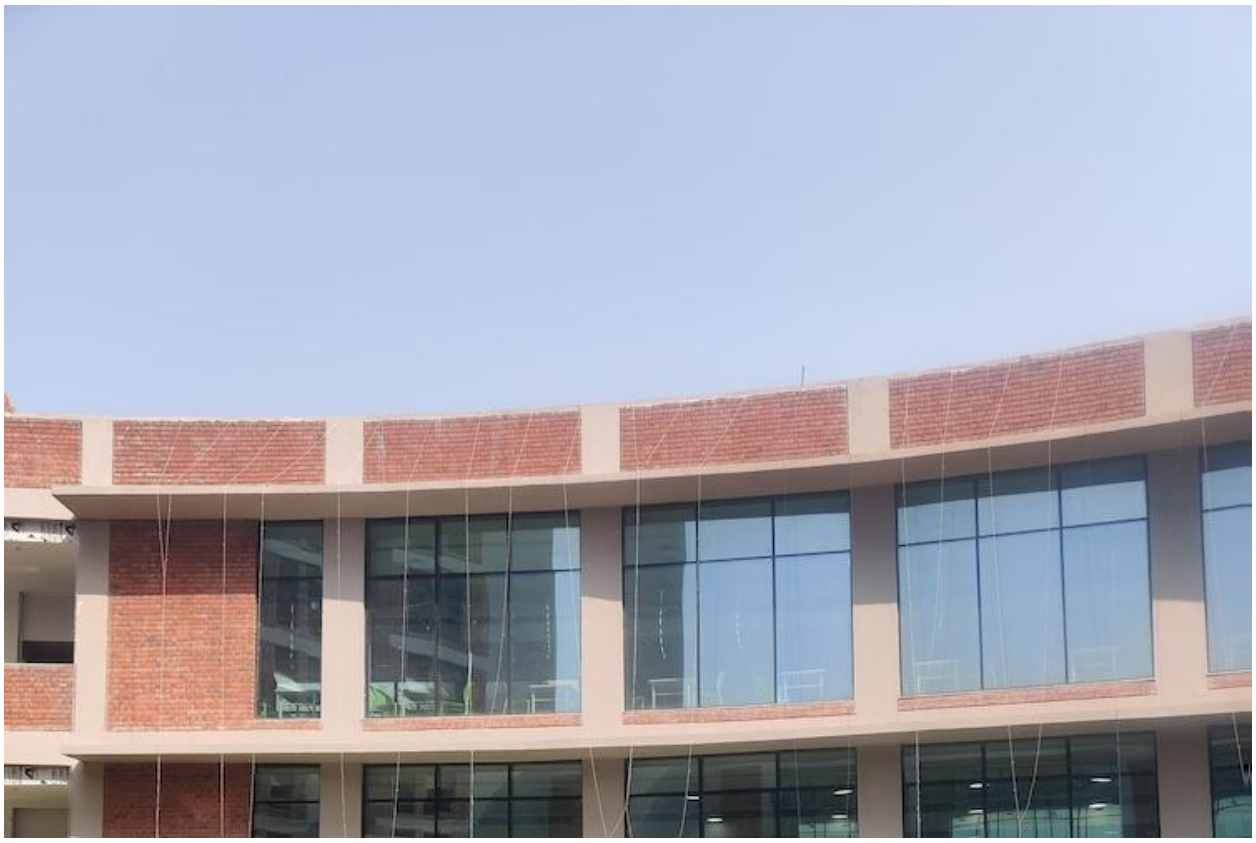
Affine model to stitch the images together is achieved by the **stitching()** function and a model for stitching the already stitched image.

stitching():

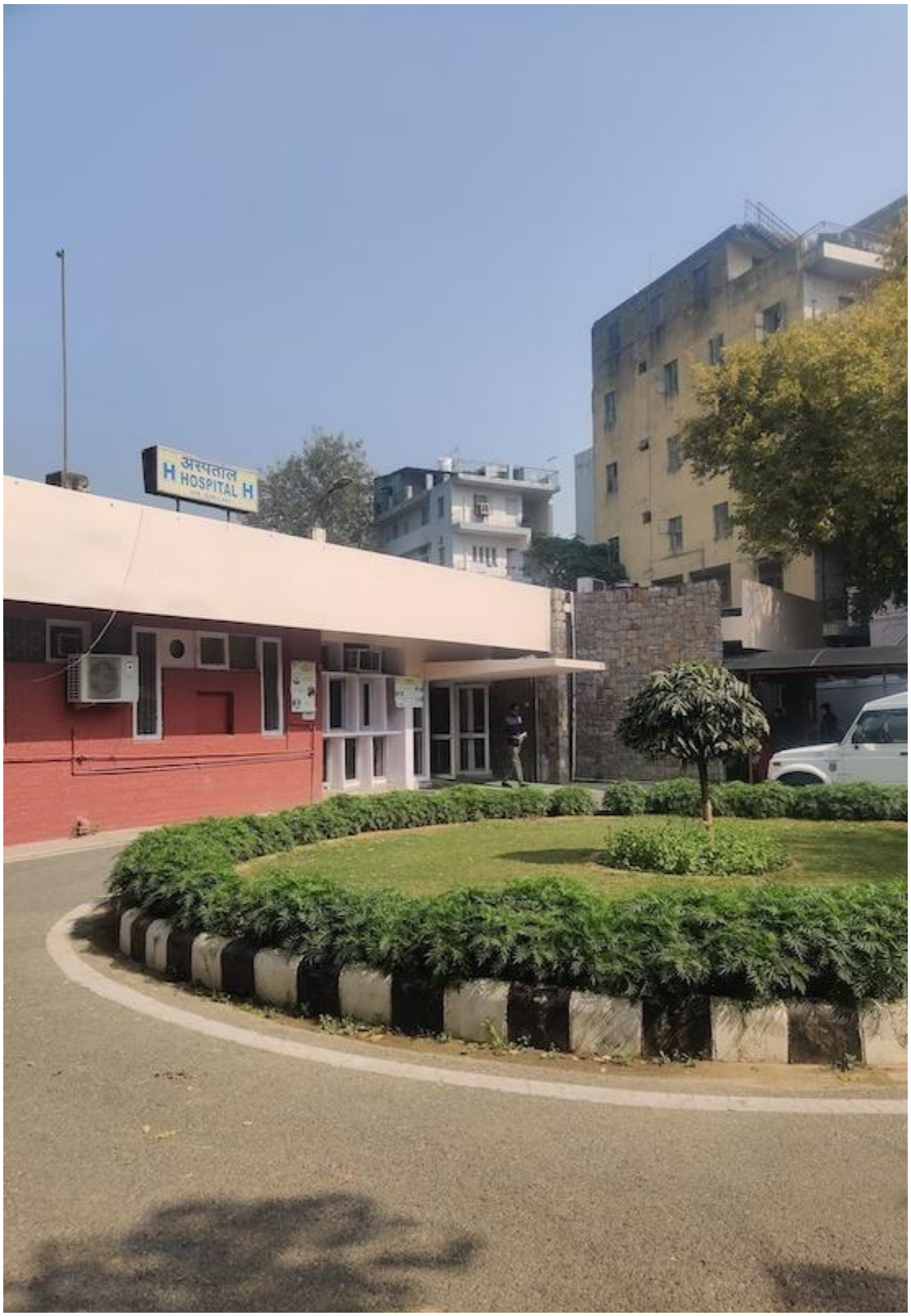
This method takes input as f1, f2 frames to be stitched and affine matrix. A new frame is created based on the translation happened in the frame2 by using topright corner. Now for each point in the frame 2 I will be getting the corresponding coordinate in the stitched image and I will update the values of frame2 accordingly. This will be the new stitched image let it be stimg. Now this stimg will be treated as a base image and new frame arrived is treated as frame2 and Affine is calculated between a point in frame2 and frame1, this will be dot product with the previous iteration Affine with the newly created Affine matrix (i.e by using best 3 matches). Now this affine matrix is used for stitching. This process is repeated till we finish all the images.

Below are the stitched images for each dataset.

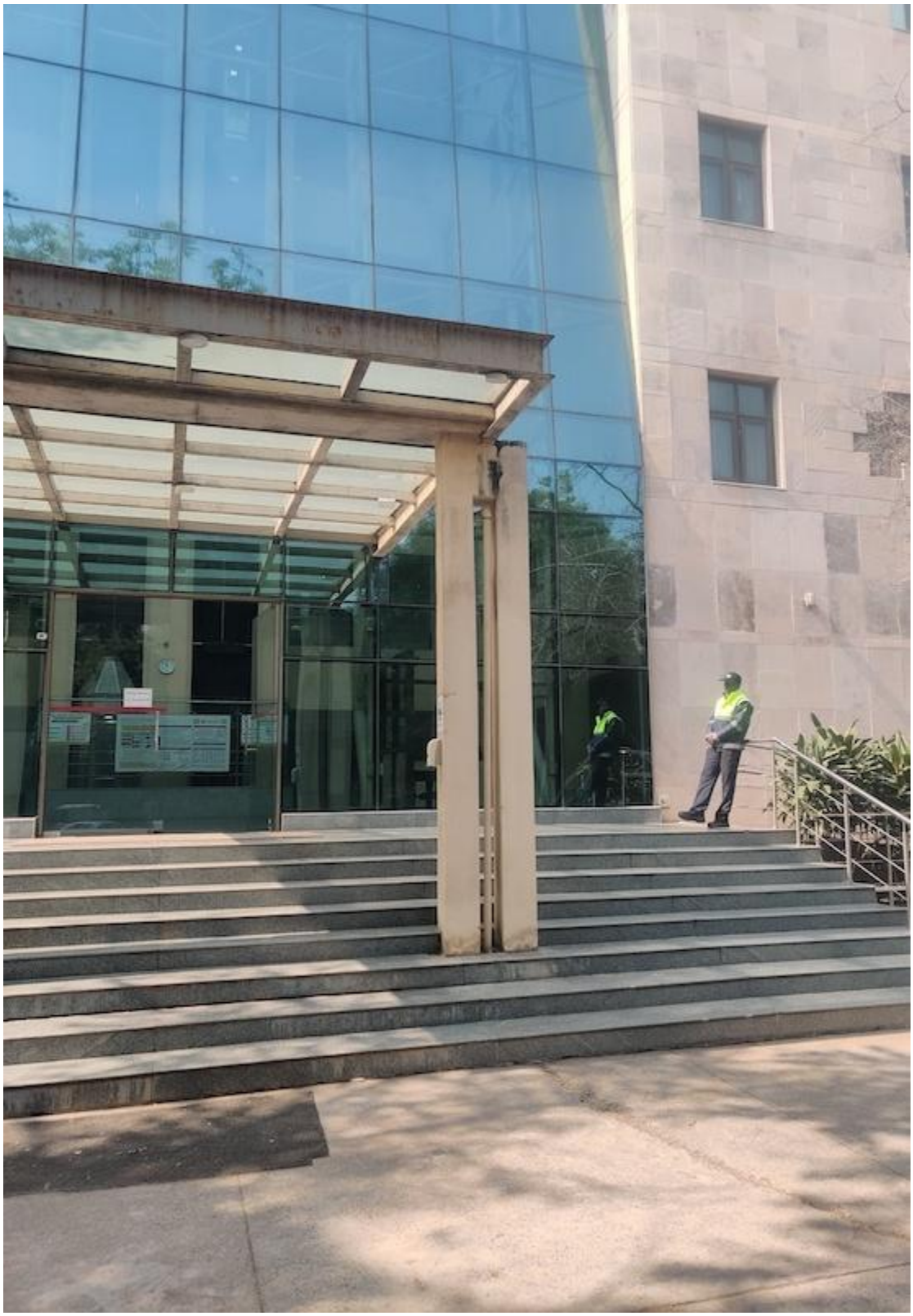
1 **k=0.12 and threshold=0.695**



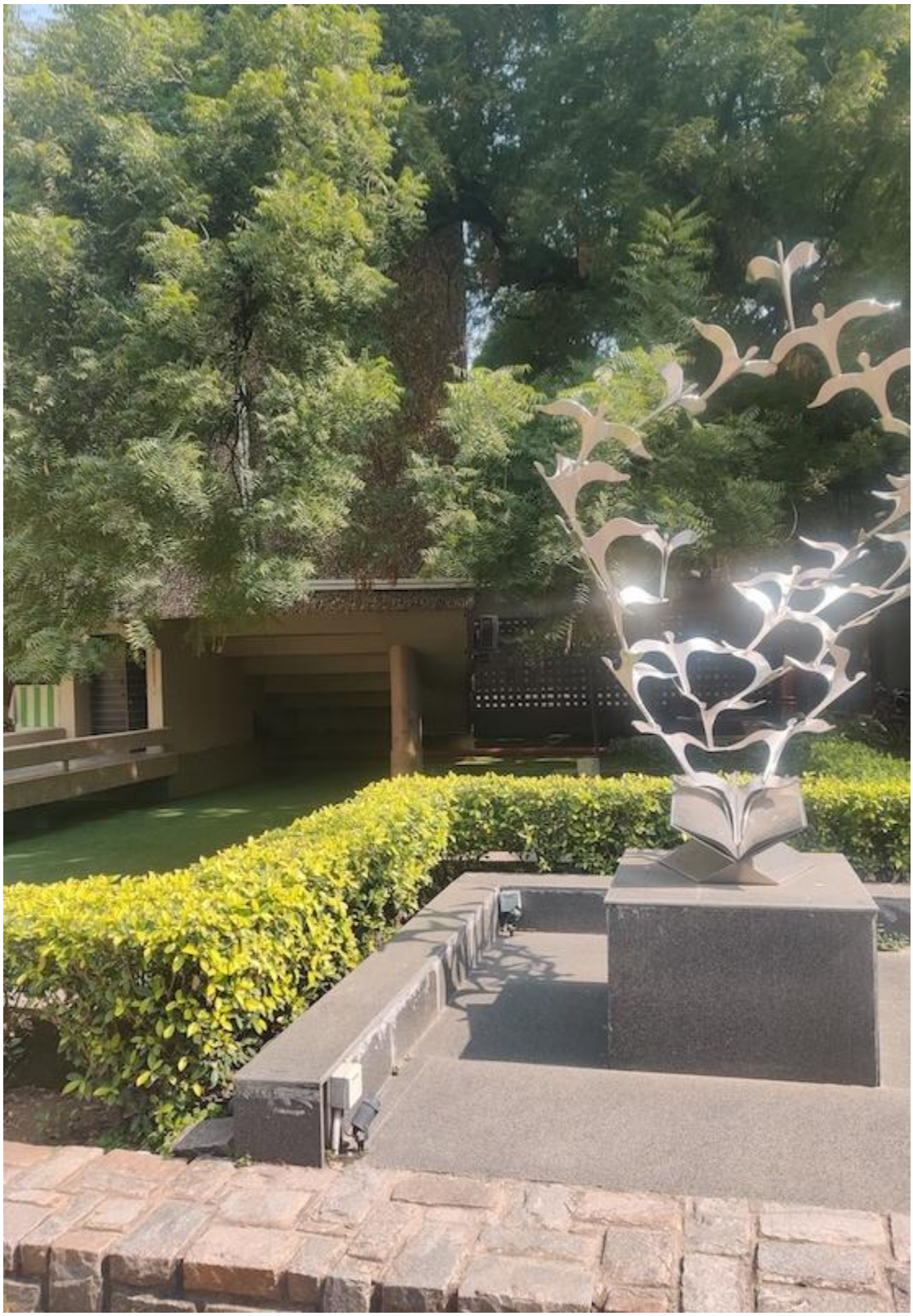
2 **k=0.12 and threshold=0.695**



3 **k=0.12 and threshold=0.755**



4 **k=0.12 and threshold=0.715**



5 **k=0.12 and threshold=0.715**