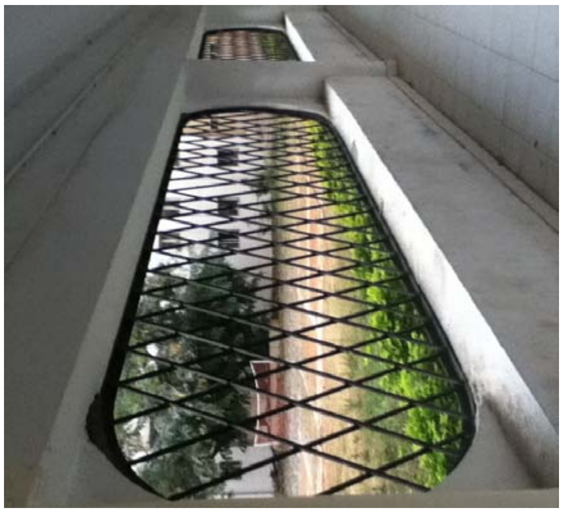
Computer Vision

Ramakrishna Vedantam S
200930008

Assignment 1 Project Report

Data Collection:
The data was collected from the internet and mainly from photos within our campus. Plenty of interesting photographs of the OBH and its surroundings are there in the data set. Especially for rectification the campus provided a large number of interesting images :



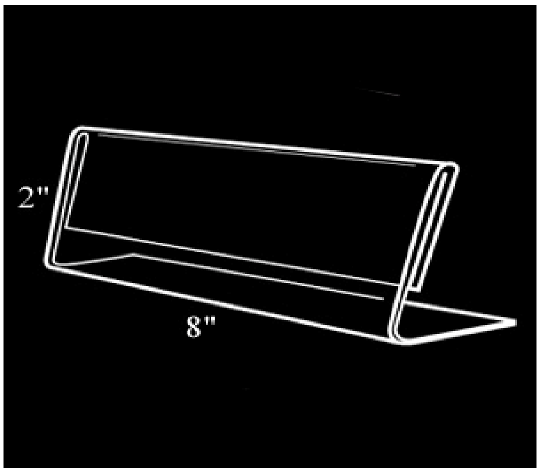
Care was taken that the images were taken at different times of the day to be able to observe the differences in contrast and brightness.The above images were taken in the evening, and the following were taken in the morning:

" NAVJEEVAN "
TRANSPLANTED FROM THE MAIN ROAD (Opp. IIIT)
TO NILGIRI BHAVAN NORTH ON 2nd Sept. 2002
RETRANSPLANTED FROM NILGIRI BHAVAN NORTH
TO PALASH NIVAS CIRCLE on 6th AUGUST 2011

This image was taken at night :

Images were also taken Inodoors, to see the difference they may have with outdoor images:

Images above also have a high amount of distortion whereas some of the others are very clear. Different cameras were used for this purpose.
Apart from the images taken on campus, some other interesting images were taken from the internet:



Metric Rectification

Aim: Given an image with a nameplate, numberplate and so on from a side view, generate the front-on view for it.

Code Description:

Interface

The code shows a side by side display of two images, one that is the image we want to rectify and a blank image. The user has to select four points, starting from the top left corner and click four points in a clockwise manner. One he is done, the user has to select a rectangle on the target plane by clicking two diagonally opposite points. Once done, user has to press “Esc” key and the rectified image is generated.

On clicking a square is drawn at the point of click.

Functions Used
The code uses the cvFindHomography function to compute the homography.

Method
Once the 4 points in the image and the corresponding four points in the rectified image are specified by the user, the points are given to the cvFindHomography function to compute the inverse homography between the source and the destination planes. In this code, the homography is found between the blank image as the source and the input image as the destination. This ensures that for every point in the blank image we are able to get a point with some color in the source image. This process is carried out iteratively. For each pixel we get some color in the source image which is

copied to the destination image to build the rectified image.

Two codes have been made, one that uses a simple integer typecasting to find the pixel color in the reference image and the other is using a billinear intrapolation. There is some bug with the billinear intrapolation, it doesnt work for all the images, hence the executable metric_rectify is only to be used when rectifying.

Experiments and Results:
1.With a licence plate image:

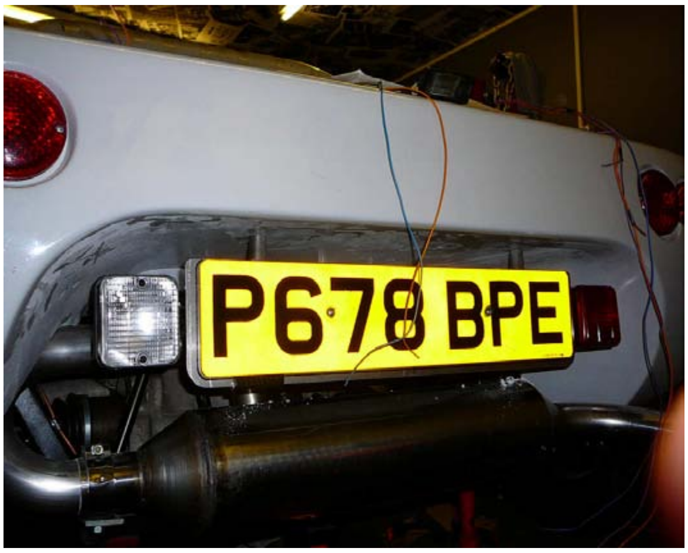


Illustration 1a: original image

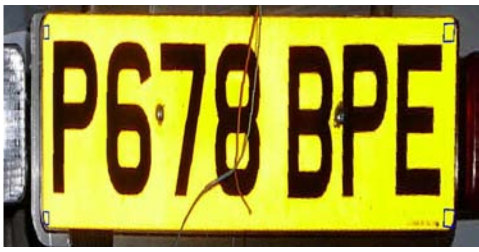


Illustration 1b: without billinear

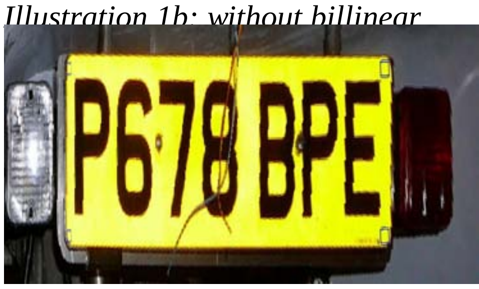


Illustration 1c: Billinear Interpolation

2.With a grill in the OBH:

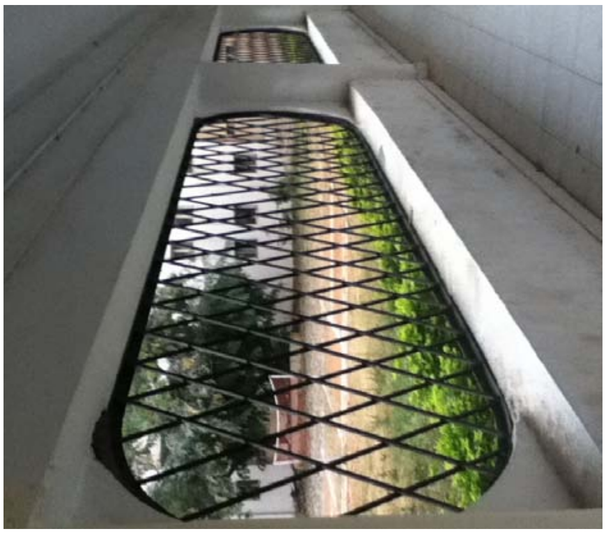


Illustration 2a: grill original

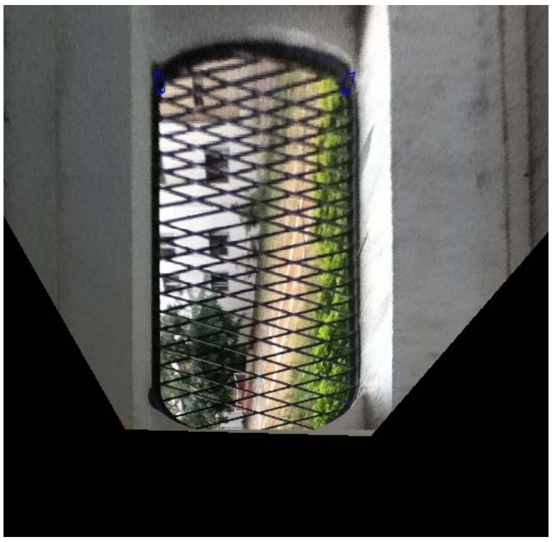


Illustration 2b: grill rectified

3.Badminton Court:

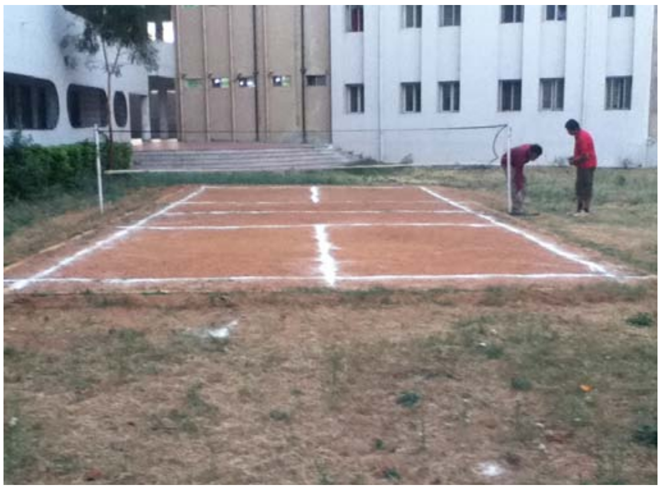


Illustration 3a: Badminton court

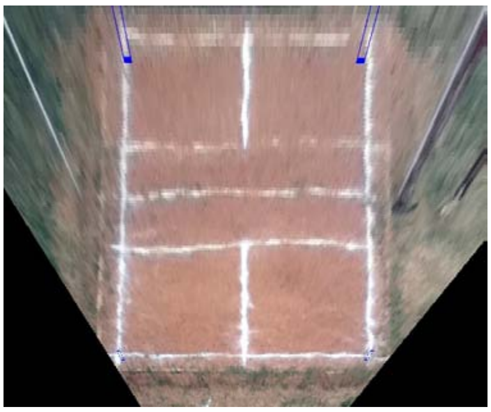


Illustration 3b: Rectified Badminton court

4. A 3-D line object:

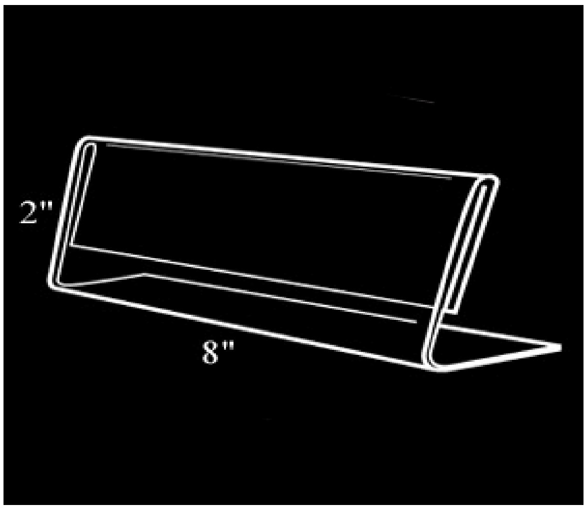


Illustration 4a: Original Image

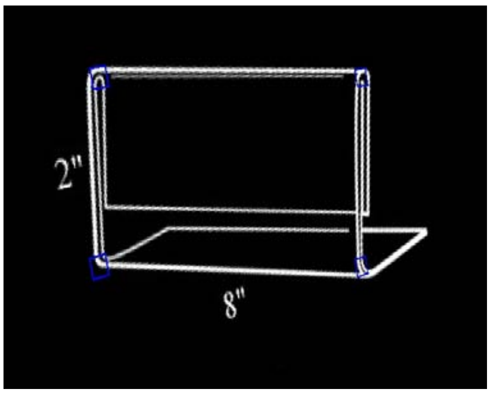


Illustration 4b: Rectified Image

5.Navjeevan Tree Board (Day and Night):

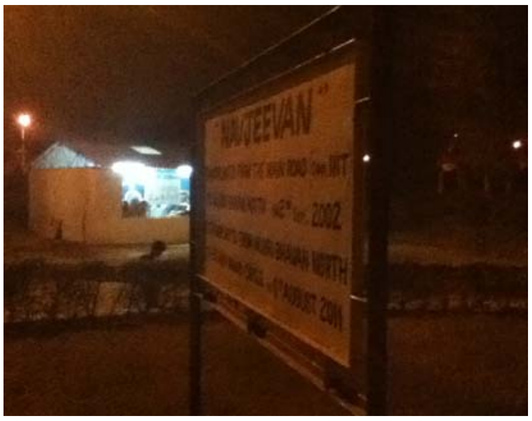


Illustration 5a: Night Image (more Distorted)

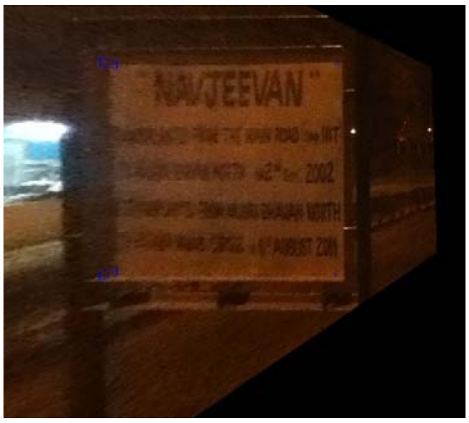


Illustration 5b: Rectified Image

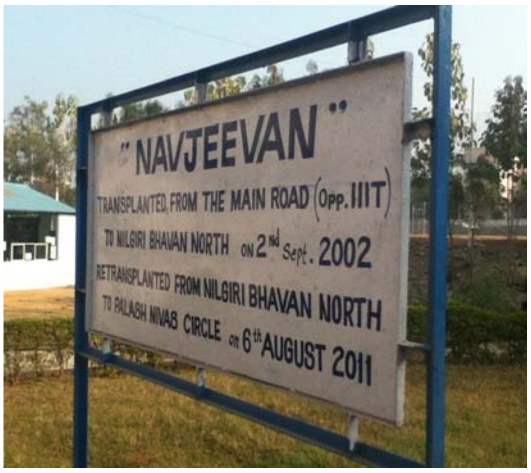

" NAVJEEVAN "
TRANSPLANTED FROM THE MAIN ROAD (Opp. IIIT)
TO NILGIRI BHAVAN NORTH ON 2nd Sept. 2002
RETRANSPLANTED FROM NILGIRI BHAVAN NORTH
TO PALASH NIVAS CIRCLE ON 6th AUGUST 2011

Illustration 6a: Navjeevan (day)

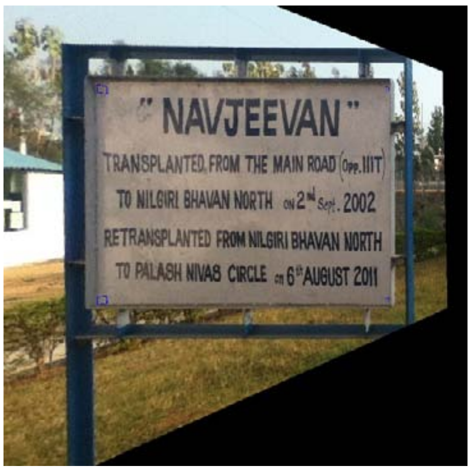

" NAVJEEVAN "
TRANSPLANTED FROM THE MAIN ROAD (Opp. IIIT)
TO NILGIRI BHAVAN NORTH ON 2nd Sept. 2002
RETRANSPLANTED FROM NILGIRI BHAVAN NORTH
TO PALASH NIVAS CIRCLE ON 6th AUGUST 2011

Illustration 6b: Rectified Image

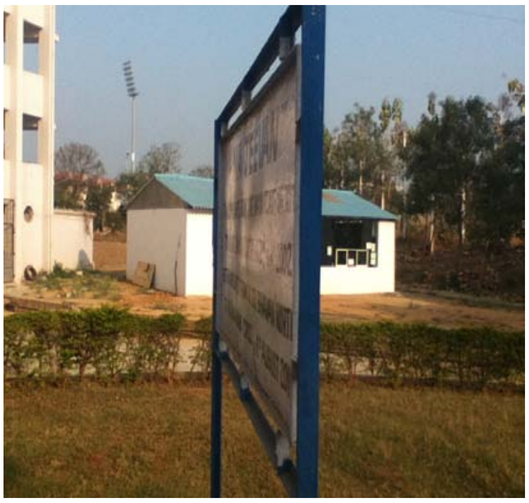


Illustration 7a: Navjeevan (skewed angle)

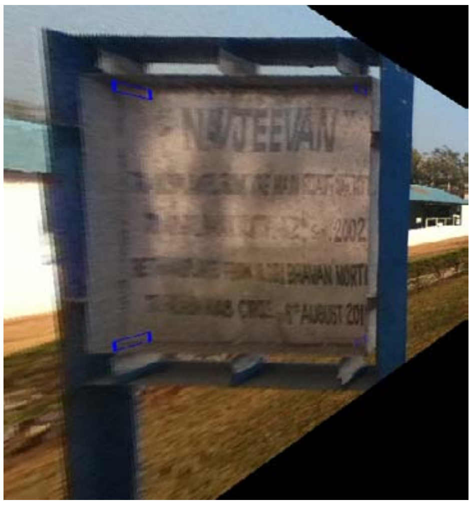


Illustration 7b: Rectified Image

6.Hexagonal Tiles

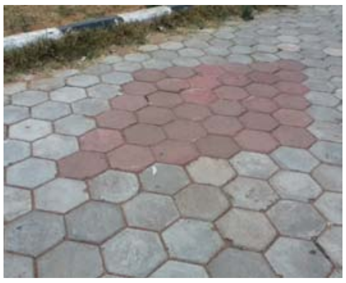


Illustration 8a: hexagon shaped tiles

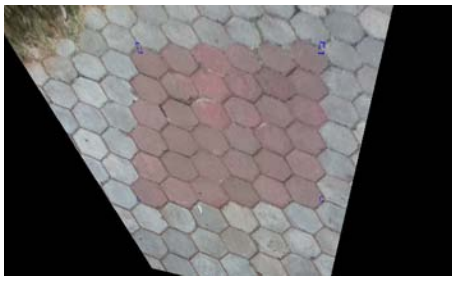


Illustration 8b: Rectified Image

Discussion of the results:

1. The number plate, when rectified with billinear interpolationa and without showed a better result when billinear interpolation was not used. It can clearly be seen that the digits are more legible in the case *1a* as compared to the case *1b*.

2. The grill has points which are in a plane but the plane itself contains points at a dept behind it visible through the grill. All these points map to a new perspective on application of homography and maintain the sense of depth in the new plane as far as what is visible behind the grill is concerned.

3. The image *3b* is an interesting study in observing what happens to the points on the plane perpedicular to the court such as the people standing on the court and the poles on which the nets are attached. Further we can see the effect of a narrow angle from which the plane has been imaged since we see that as we move upward along 3*b* , there is definitely a greater amount of blur visible.

4. It is intersting to note how the digits alongside the sides of the plane are transformed as the plane goes to the frontal view. This is the main reason why this example was chosen.

5. The image *5a* is shown in the night time with a lot of distortion because the camera did not have flash and the lightting was not perfect. We see that the rectified image *5b* is also almost illegible. The image *7a* is taken from a narrow angle where a large part of the plane was not visible, as a result the rectified image *7b* is also poor in quality. The best tradeoff is when the angle is not too acute and the image has less distortion, the case in which we have the best results as illustrated by *6a* and *6b*.

Improvements suggested:

The image given should first be filtered so as to eliminate any camera noise and then given to this program to rectify and bring into frontal view. Incase the only interst is in being able to read the characters in the rectified image more easily then image processing techniques such as error diffusion may be used for this purpose.

PANORAMA

Aim: Given n images, bring all of them into the view of one of the middle images. That is, generate a panoramic view from the given set of images.

Code Description:

How to run
The details are given in the readme file.

Interface
On executing the code, a two side by side windows open that display the two leftmost images. The user has to select the points that match between the two images (atleast 4) and press escape. On pressing escape the next pair of images is shown to the user and so on. All the points clicked by the user in the left image are marked with a yellow dot and those on the right are marked with a red dot. Once the user is done with selecting all the pairwise point matches, the two windows close and a new window with the panorama is shown. Also the generated panorama is stored as “panorama.jpg” in the folder where the code is present.

Functions Used

The openCV functions like cvFindHomography cvMatMult cvSetIdentity and others are used.

Method

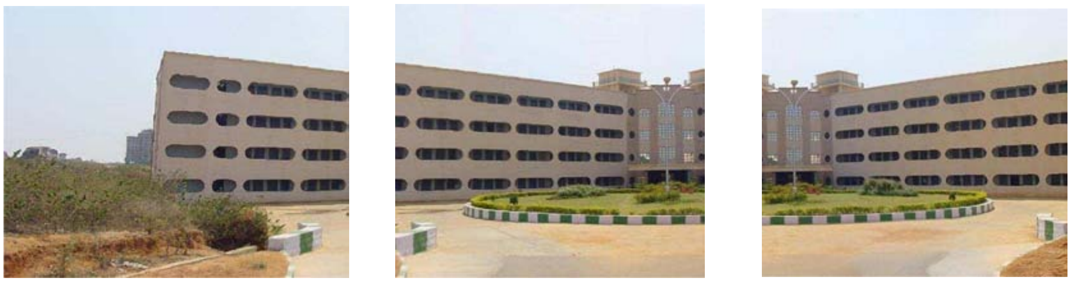
Once the user supplies all the point matches for let us say, n images. We calculate the pairwise homographies for all the images with respect to the adjacent plane. For even 'n' the reference is chosen as (n-1)/2+1 th image where as for all odd 'n' the reference image is chosen to be the (n-1)/2 th image.

Once we have all the homographies with resepect to the adjacent planes, we multiply them to get the relation between a point in a plane on the periphery and the corresponding point in the reference plane. This is done for all the images, after which homography is applied on the extreme points of the leftmost and the right most images to find where they map in the reference plane. Accordingly a new canvas is declared with the size accomodating the points generated in the previous step. Also the offset for transforming the origin in this canvas to the origin of the reference image are stored.

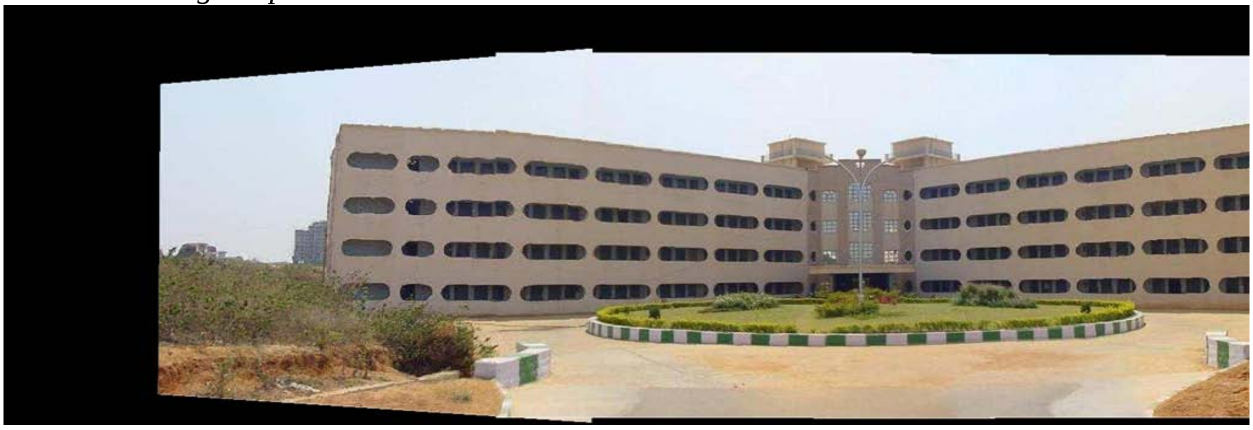
Now we proceed to the actual step of computing the panorama. The coordinates in the canvas are transformed to the frame of reference of the reference image and all the inverse homographies, which take the points in the reference frame to the frames of any of the images are applied on a point in the canvas. According to where the transformed point lies, we pick the color from the corresponding image and map it to the point in the canvas. This is done iteratively for all the points in the canvas. At the end of this step the canvas is filled with the panorama of the 'n' images with respect to the corresponding reference frame..

Experiments and Results:

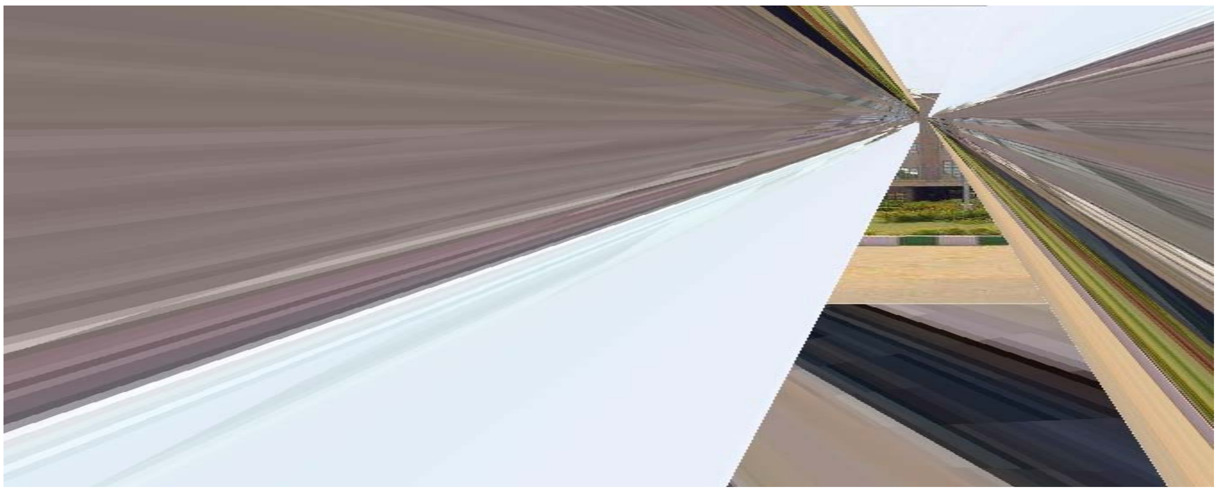
A.Three images from OBH (from the lecture slides):



Panorama with good point matches:



Panorama if the point matches are not good and only 4 points (bare minimum) are chosen:

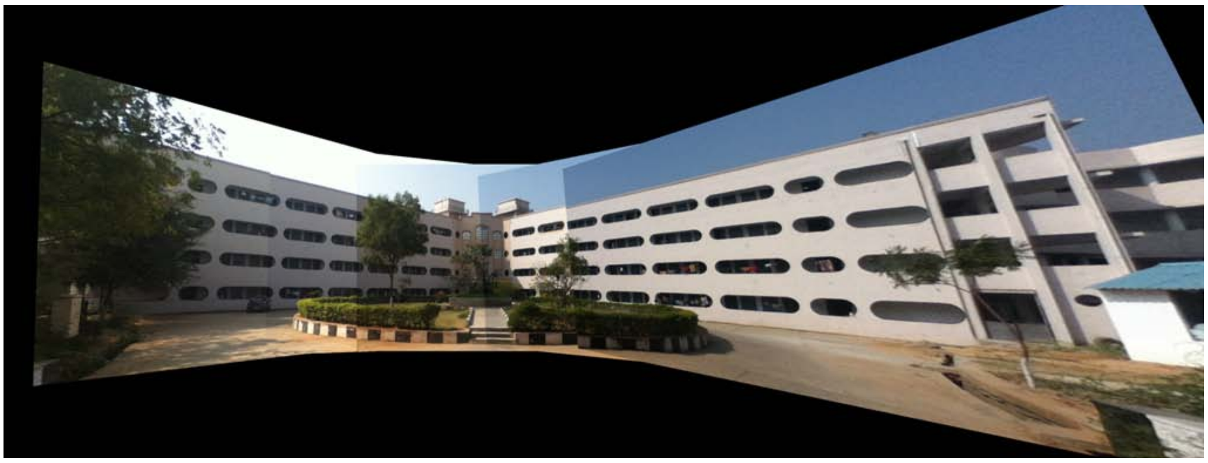


Discussion of the result:

We can clearly see the usefulness of the RANSAC method here, with the ransac threshold set as five and sufficiently large number of point matches, the result is much better than the case where the point matches are the bare minimum required algebraically (namely four) and the point matches given are not that accurate.

B.Five Revamped OBH images:

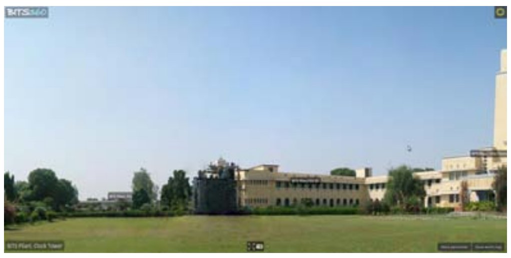
Generated Panaroma:

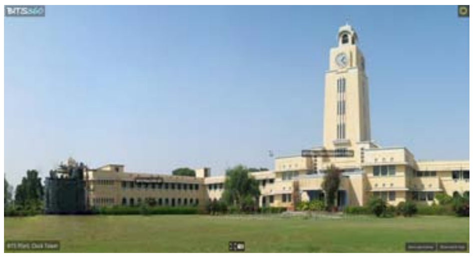


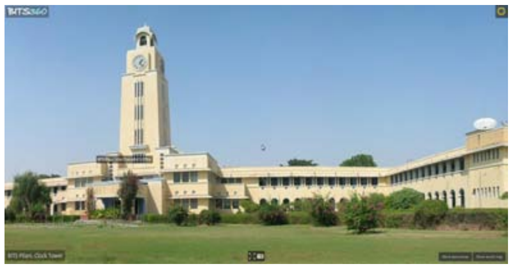
Discussion of the Result:

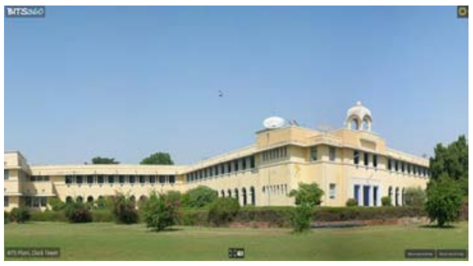
The images were taken in the morning with the sun on the left side of the building. As the camera went from the left to the right, we can see that the contrast of the image changes most noticeable is the difference in the sky. As a result of this, when the camera is facing against the sun the exposure is different from what we get when it is in its direction, in some sense. This is why we get the creases in the middle of the image. These creases can be avoided by burring or smoothing the image.

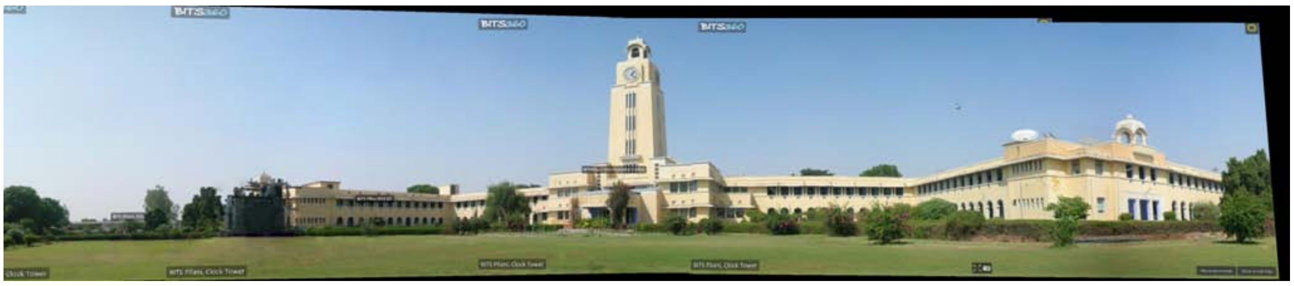
C.Four Images from BITS pilani (courtesy http://bits360.com)











Generated Panorama:

Discussion of the Result:

For this set of images, a large number of matching points were given for each image pair along with a few bad matches. The outliers have been desicively rejected and the appropriate homographies have been found using the RANSAC method giving rise to a pleasant continuous panoramic view. The images were also at similar exposure levels which helped avoid the occurence of creases like in the previous case.

Improvements Suggested:
The improvements that can be made to the present code is to incorporate blending of the images so

that the cerases may be avoided. Further automatic algorithms like the SIFT algorithm may be used to automate the process of selection of matching points. Giving a large number of matching points especially for large images can be tiring !