Panorama Stitching
Disparity Map

Mahendra Nandi

May 2021

1 Panorama Stitching

In the first part of this project, we shall perform image stitching using Python and OpenCV. Given a pair of images that share some common region, our goal is to "stitch" them and create a panoramic image scene. The steps to achieve that goal are as follows:

- Keypoint detection
- Local invariant descriptors (SIFT)
- Feature matching
- Homography estimation
- Perspective warping

1. **Keypoint Detection**
In our code, we have used the function $cv2.ORB\_create()$ to create a descriptor object, and then we have extracted keypoints by calling the method $detectAndCompute()$. We have visualised the keypoints in the image by the function $cv2.drawKeypoints()$.

2. **Feature Matching**
After extracting keypoints from both images,we would like to compare the 2 sets of features and stick with the pairs that show more similarity. For this, we have used a BruteForce (BF) Matcher object by calling the function $cv2.BFMatcher$, in which the first parameter is the distance metric and the second one is the crossCheck boolean.

3. **Homography Estimation**
Homography is a 3x3 matrix, essentially, a linear transformation that will relate the two images. Since we have obtained feature points from the two images, we

can pass them into the $cv2.findHomography()$ function to get a perspective transformation.

4. **Perspective Warping**

Once an accurate homography has been calculated, the idea is to transform one of the images so that both images merge as one. This is done using the $cv2.warpPerspective()$ function in OpenCV.

An Example

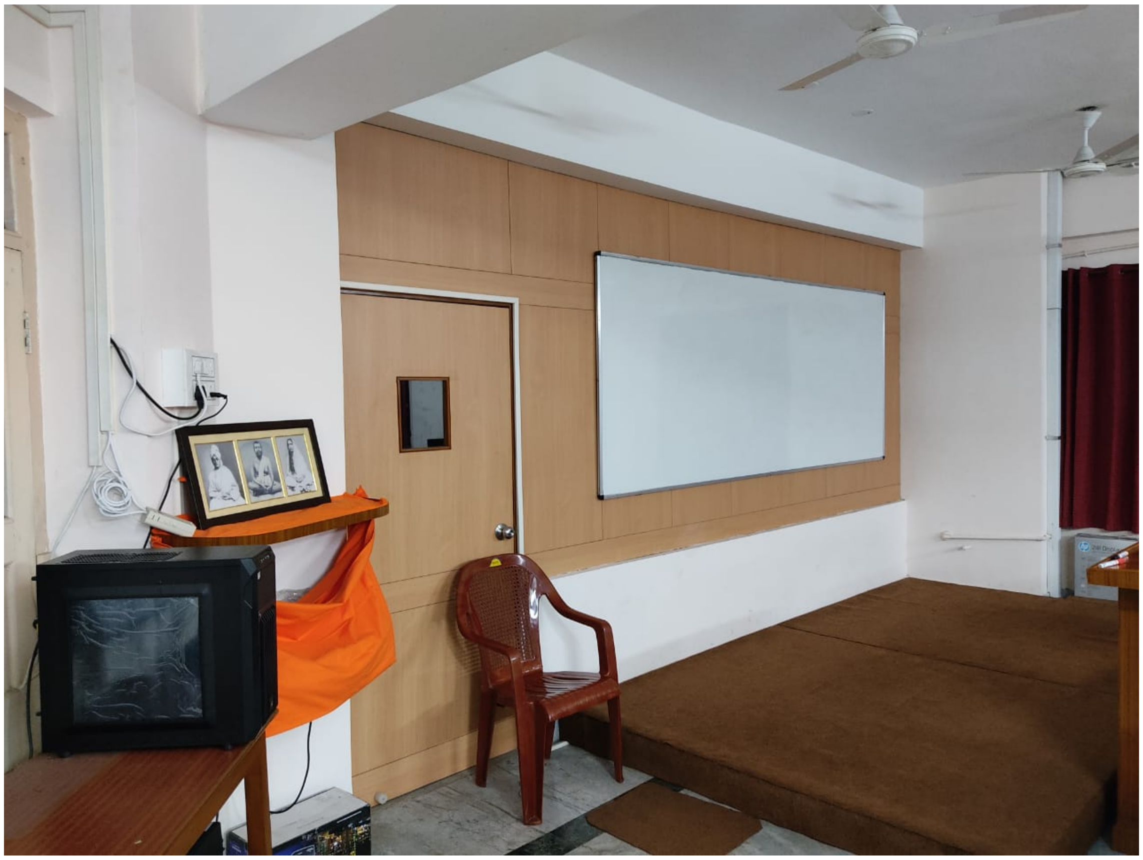

Figure 1: One of the two images to be stitched

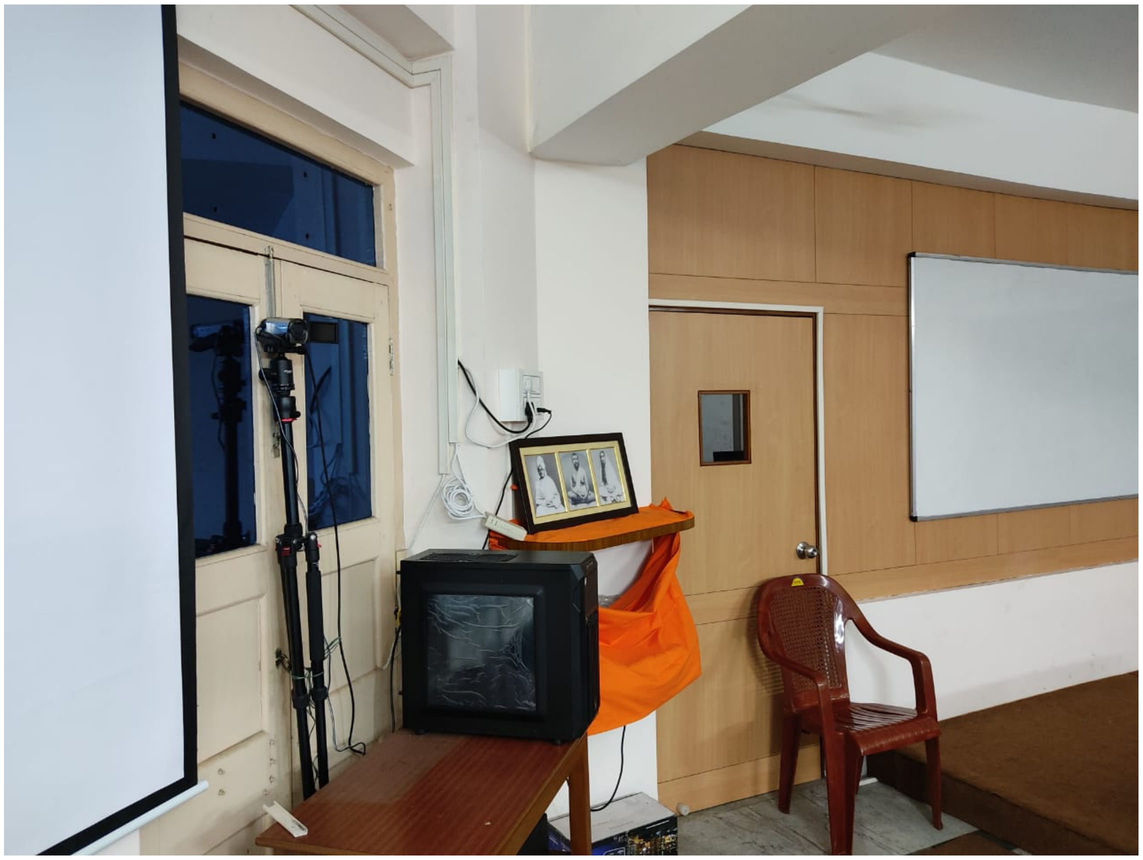

Figure 2: another one of the two images to be stitched

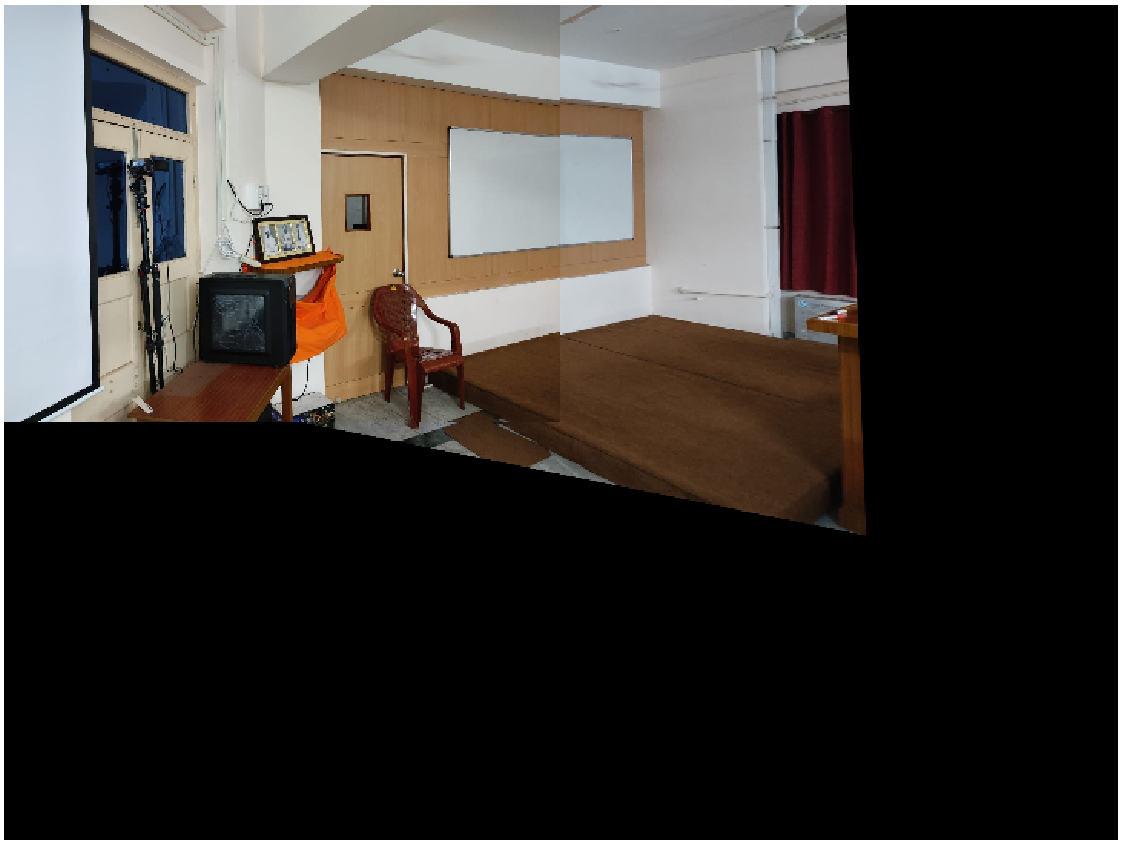

Figure 3: The stitched panorama

2 Disparity Map

If we have two images of same scene, we can get depth information from that in an intuitive way. Below is an image and some simple mathematical formulas which proves that intuition. The above diagram contains equivalent triangles.

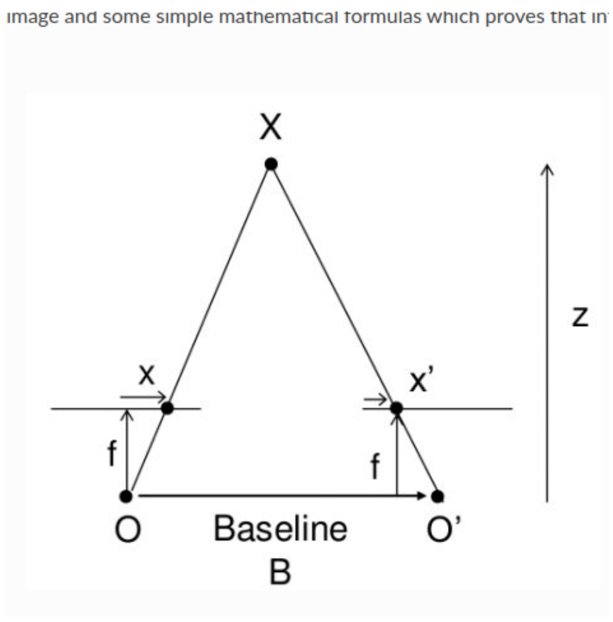
image and some simple mathematical formulas which proves that in

Figure 4:

Writing their equivalent equations will yield us following result:

$$disparity = x - x' = \frac{Bf}{Z}$$

x and x' are the distance between points in image plane corresponding to the scene point 3D and their camera center. B is the distance between two cameras (which we know) and f is the focal length of camera (already known). So in short, above equation says that the depth of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. So with this information, we can derive the depth of all pixels in an image.

So it finds corresponding matches between two images. We have already seen how epiline constraint make this operation faster and accurate. Once it finds matches, it finds the disparity.

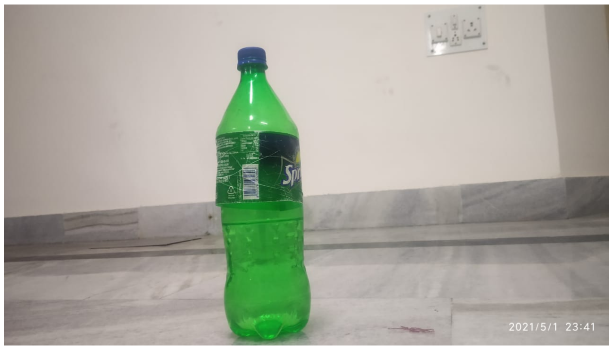

Figure 5: Image-1 used for disparity map

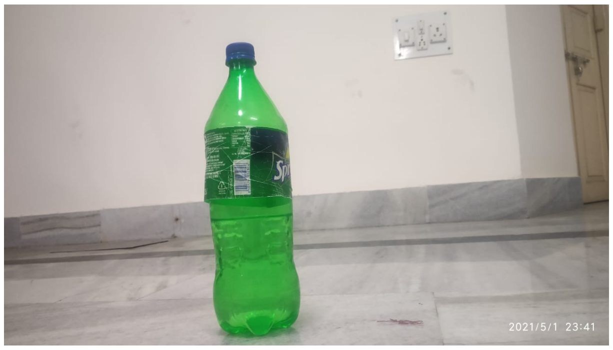

Figure 6: Image-2 used for disparity map

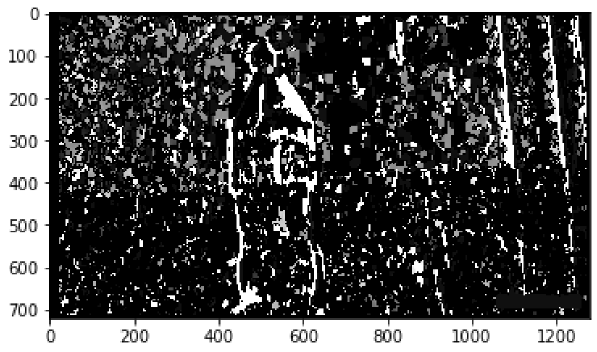

Figure 7: One example of disparity map