Traditional Assignment1 : Image Stitching using OpenCV Python

**Background:** This project was implemented in two stages. In stage 1, I have worked on two image stitching problems with a custom dataset I have borrowed from [1]. In this stage, I have worked on the simple image stitching algorithm that accepts two images as input and generates a single stitched image as output. In stage 2, I have worked towards improvising the algorithm implemented in stage 1 by improving certain drawbacks seen in stage 1 (like formation of hard seams after stitching) and opening room for stitching multiple images together.

**Stage 1 - Two image stitching problem:** The two image stitching problem is a simple straightforward stitching approach that does not take into consideration any constraints and hence doesn’t require much complex transformations. The algorithm [2], [3] I have made use of in this stage comprises 6 steps including: 1. computing keypoints and descriptors using SIFT, 2. calculating the distances between the descriptors in each image, 3. picking m best matches out of the existing matches, 4. computing the homography, 5. warping the images to prepare for stitching and 6. finally stitch them together. This is a very basic version of implementing an image stitching algorithm involving only two images and not many constraints. The following are inputs and the outputs obtained using this algorithm.

  

A B C

**Stage 2 - Multiple image stitching problem** In stage 2, I started working on improving the algorithm a bit so as to fix 2 issues: 1. Include multiple images in the stitching process, 2. Fix the hard seaming problem by using a blending algorithm. Inorder to get started on multiple image stitching, I borrowed intuition from [4] and started working on putting together various things that fine tune the stitching process. The stitching process is pretty much the same as the one in stage 1. The only difference lies in how to select the appropriate images (dataset borrowed from [5]) and warp them accordingly to prepare for stitching. In this particular algorithm I follow, the following conditions hold:

1. The image dataset is partitioned into two sets namely left and right.
2. We start with the left and warp two images at a time until all the images in the left are covered.
3. We then work with right in a similar fashion but in the reverse order.
4. Upon obtaining the left\_stitched image and right\_stitched image, we warp them together and stitch them.

While warping the images I have computed homography in a way similar to stage 1. Now while we try to warp the images, it is important to include image corners in the homography and properly arrange images together. That is during warping it is possible for a few pixels to be misplaced [6] that could result in holes. To fix this we can use backward warping. This works the following way: we observe the four corners of the image and apply forward warping to them and for the rest of the pixels we apply inverse warp. We then pass this warped image to be stitched and blended for a smooth finishing.

   

I II III IV



V

**Blending:** Now if after observing figure C and figure V, we can have some intuition on why blending is necessary. Blending gives a smooth finishing to the stitched image without lines depicting stitching differences like in fig C. We call these lines seams. I have used blending in my program by applying certain smoothing masks on warped images such that they blend together smoothly with finer texture.

**Efforts in the project:** I have worked towards fixing the stitching problem that only involved translation among an axis but not rotation. The datasets in gauchospace deal with the case where there is rotation of the camera involved while capturing the image. In the approach discussed above we are dealing with left to right image stitching which is not suitable for rotation problems. Also the warping for rotation is different from that of translation. Upon the professor's suggestion, I have verified the stitching algorithm that starts from the center image and works the way upto the sides. But I didn't have much success with that algorithm due the dimensionality problem and not being able to extract the minimum number of points of correspondence between images. Hence I limit my work towards the translation problem for this assignment but wish to experiment more later.

**Implementation details:**

This project was implemented in Spyder IDE (standard python editor in anaconda environment) and makes use of OpenCV library version 3.4.2.

**References:**

1. [Image stitching datasets](https://flyingspringrol.github.io/cs194/proj5/)
2. [Image Stitching Using OpenCV](https://towardsdatascience.com/image-stitching-using-opencv-817779c86a83)
3. [OpenCV image stitching second part](https://python.plainenglish.io/opencv-image-stitching-second-part-388784ccd1a)
4. [Image Stitching <br> A Simplistic Tutorial](https://kushalvyas.github.io/stitching.html)
5. <https://www.kaggle.com/datasets>
6. <https://vision.cs.utexas.edu/378h-fall2015/slides/lecture14.pdf>