CV HW5 P1 Write Up

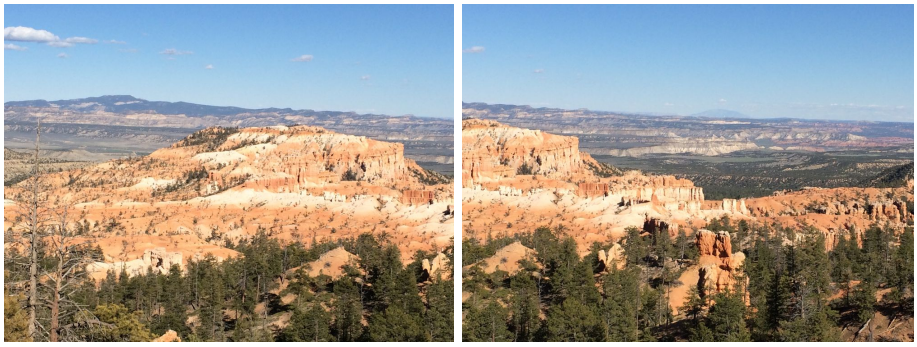
My algorithm is composed of:

1. SIFT detect key points
2. Use BFmatcher and knnmatch to match the keypoints
3. Filter matched keypoints with Lowe ratio test, check if one distance is within 75% of another
4. Filter matched keypoints with fundamental matrix calculation, check if after this filter the number of keypoint is within 75% of previous stage, if so -> matched, if not -> no match
5. Filter matched keypoints with homography matrix calculation, check if after this filter the number of keypoint is within 75% of previous stage, if so -> matched, if not -> no match
6. In a directory, all images are checked with previous steps to get if a pair is matched or not, if matched, they will be recorded in a dictionary in the form of {a: [b, c]}, which means b and c are matched with a.
7. After this the problem is modeled as a graph problem, in which each image is a node and matched image are connected by edges between them, note this is an undirected graph.
8. Then all paths between any two of the nodes in this graph will be generated. If doing multi-image mosaic, the path with all images will be selected (basically travelling salesman problem, a path passing through each node, meaning a path that all images are matched/ overlapped). Or doing dual-image mosaic, then all path with length 2 will be selected (because dual-mosaic requires two images).
9. If doing multi-mosaic, for example I have four images a, b, c, d and a path a-b-c-d, a and b will be stitched together, then ab-c-d -> abc-d -> abcd. Or doing dual-mosaic, for example I have images a, b, c and paths a-b, a-c, b-c then there will be mosaic ab, ac, bc.
10. Stitching is done with warpPerspective and offset calculated from that. For the overlapping region of the two images, cv2.addWeighted(a, 0.5, b, 0.5, 0) is used to create an ‘average’ of the images. I also draw a white boundary around the warped image for better display of the boundary. None overlapping region is just pasted onto another directly.
11. During all process above, a log of what happened will be generated and saved to each individual image set folder as ‘log.txt’.

My decision criteria (75%): comes from trial and error.

Blending algorithm: it is explained above (the stitching part, number 10).

Advantage: for images with clear and distinct features, the SIFT will work great as long as the distortion (change in aspect) is not too big. E.g.:



However, for the ones with repetitive features or too few features, the algorithm will not work well, the matcher will likely to match to wrong key points and thus give the wrong fundamental or homography matrix e.g.

PS: my results folder structure is:

```
results
└── image_set_name
    ├── dual_mosaic
    │   ├── matches
    │   └── mosaics
    └── multi-mosaic
        ├── matches
        └── mosaics
```

In matches folder you will find side_by_side SIFT match images
In mosaics folder you will find stitched mosaics
In the dual mosaic and multi mosaic folders you will also find the log.txt which is output of the program at each key stage.

For multi image mosaic you could check
results->tree-mrc->multi-mosaic->mosaics->multi-mosiac.jpg

I shrunk all output images sizes to half to reduce file size.