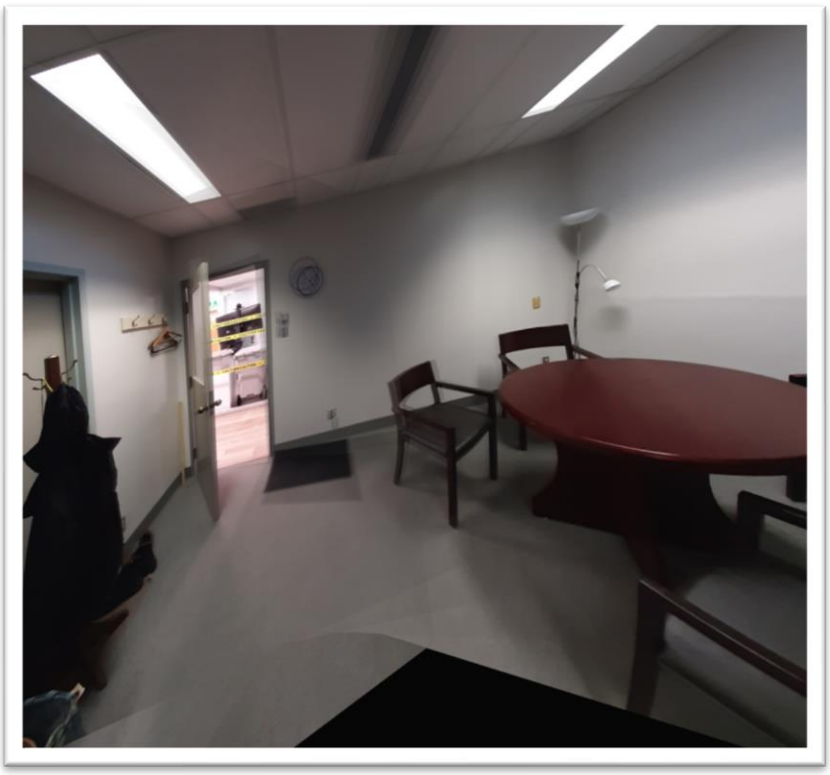


ELEC474 Project Report

STUDENT ID: 20119641

MIKE FENG

Inventory List

Dir Name	Description
D1	Stored source code for **final version**(i.e. Main_MikeFeng_20119641_V2.5_[FINAL].ipynb) and other version
D2	Has the report of the project
D3	Contains the test result of final versions
D4	Self-Assessment excel
D5	Contains the test result of other versions

Declaration of originality

This is to certify that to the best of my knowledge: the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Recoverable Signature

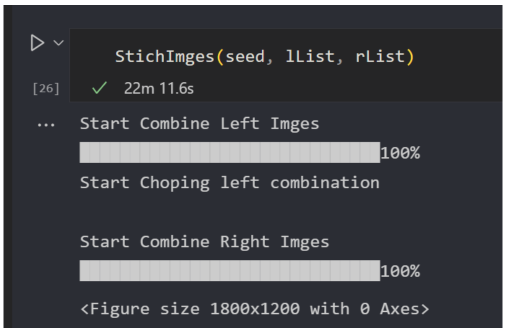
X WenTao Feng

WenTao Feng

Signed by: cd97b093-33bd-4518-9adb-be2f4f4b9391

Result description

Overall, for this project, I got 3 different versions of code that represent 3 different methods.
For method 1, I take the composition method from Kushalvyas. This method takes a much longer time than the other 2 methods. There is no more space for a list of matching metric values, please check the code result in the Jupiter notebook for the final version of the code



```
[26]  StichImges(seed, lList, rList)
      ✓ 22m 11.6s

...   Start Combine Left Imges
      ██████████████████████████████100%
      Start Choping left combination

      Start Combine Right Imges
      ██████████████████████████████100%
      <Figure size 1800x1200 with 0 Axes>
```

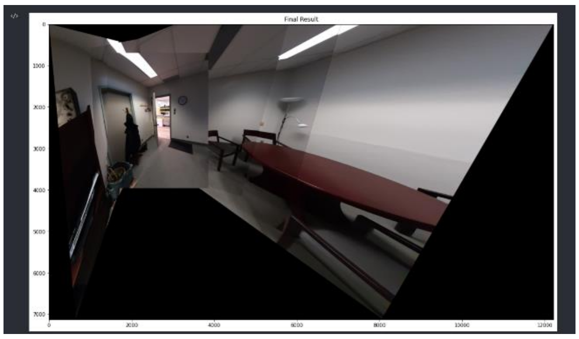
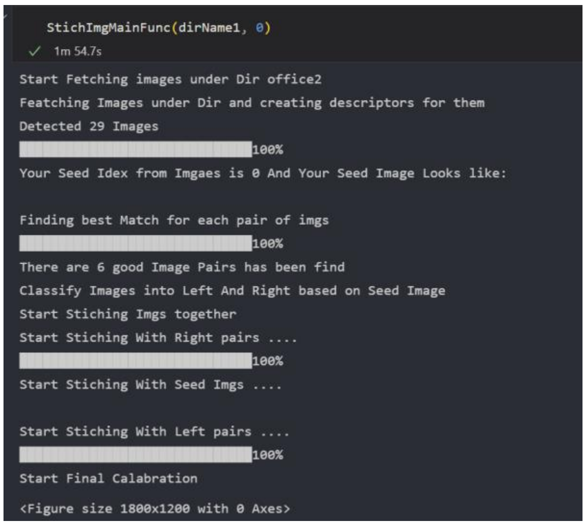


Figure 1: First method's result and runtime

The second method is based on the first method with different merging methods and the image split method. The Second method has 2 different versions which I named V2 and V2.5. The difference between them is the different composition methods. V2.5's composition method can deal with radiometric transformations since it glued the image based on pixels by pixels. This also results in a much longer run time than V2. But still, perform better than the first method.

And I take the V2.5 as my final method

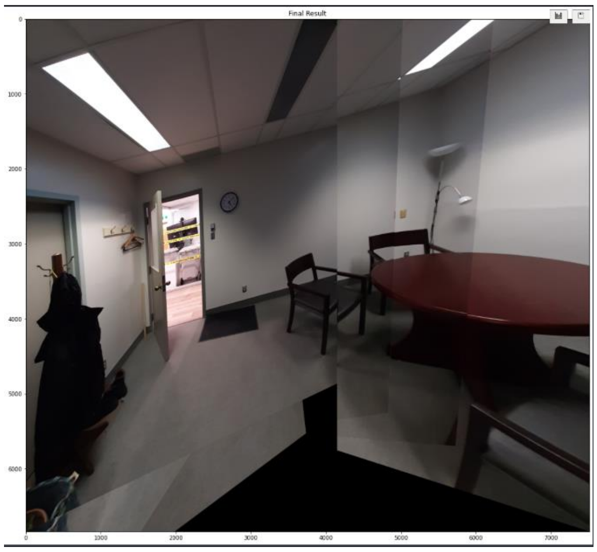


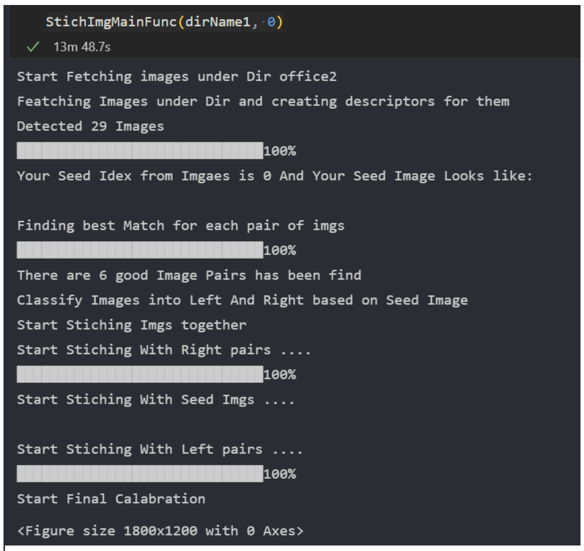
```
StichImgMainFunc(dirName1, 0)
✓ 1m 54.7s

Start Fetching images under Dir office2
Featching Images under Dir and creating descriptors for them
Detected 29 Images
████████████████████████████100%
Your Seed Idex from Imgaes is 0 And Your Seed Image Looks like:

Finding best Match for each pair of imgs
████████████████████████████100%
There are 6 good Image Pairs has been find
Classify Images into Left And Right based on Seed Image
Start Stiching Imgs together
Start Stiching With Right pairs ....
████████████████████████████100%
Start Stiching With Seed Imgs ....

Start Stiching With Left pairs ....
████████████████████████████100%
Start Final Calabration
<Figure size 1800x1200 with 0 Axes>
```





```
StichImgMainFunc(dirName1, 0)
✓ 13m 48.7s

Start Fetching images under Dir office2
Featching Images under Dir and creating descriptors for them
Detected 29 Images
████████████████████████████100%
Your Seed Idex from Imgaes is 0 And Your Seed Image Looks like:

Finding best Match for each pair of imgs
████████████████████████████100%
There are 6 good Image Pairs has been find
Classify Images into Left And Right based on Seed Image
Start Stiching Imgs together
Start Stiching With Right pairs ....
████████████████████████████100%
Start Stiching With Seed Imgs ....

Start Stiching With Left pairs ....
████████████████████████████100%
Start Final Calabration
<Figure size 1800x1200 with 0 Axes>
```

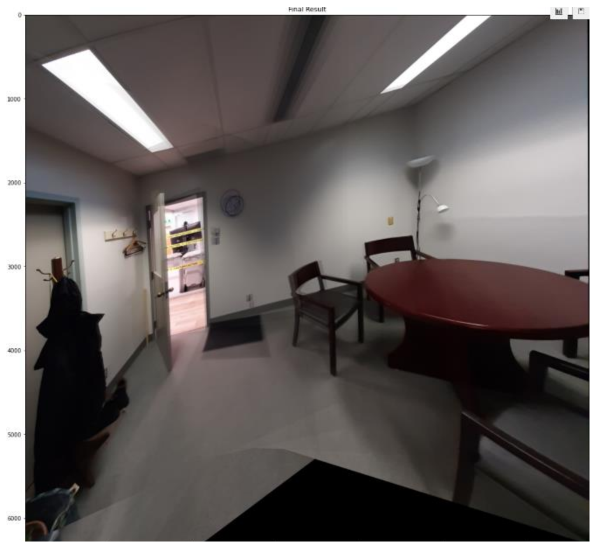
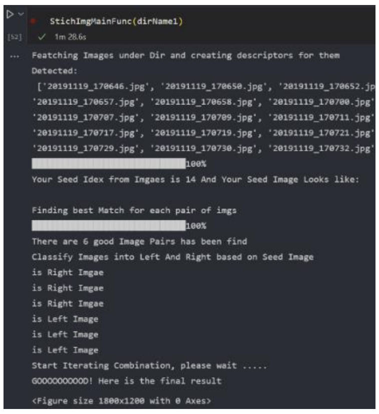


Figure 2: Second method's results and runtimes

The third method is a very simple iteration method with simple composition method. It can achieve the best time performance among 3 methods. However, the result is not as good as other two method since the composition method intentionally chops out some part of the merged image.



```
StichImgMainFunc(dirName1)
[52]  ✓ 1m 28.6s
...   Featching Images under Dir and creating descriptors for them
      Detected:
       ['20191119_170646.jpg', '20191119_170650.jpg', '20191119_170652.jp
      '20191119_170657.jpg', '20191119_170658.jpg', '20191119_170700.jpg'
      '20191119_170707.jpg', '20191119_170709.jpg', '20191119_170711.jpg'
      '20191119_170717.jpg', '20191119_170719.jpg', '20191119_170721.jpg'
      '20191119_170729.jpg', '20191119_170730.jpg', '20191119_170732.jpg'
      ████████████████████████████100%
      Your Seed Idex from Imgaes is 14 And Your Seed Image Looks like:

      Finding best Match for each pair of imgs
      ████████████████████████████100%
      There are 6 good Image Pairs has been find
      Classify Images into Left And Right based on Seed Image
      is Right Imgae
      is Right Imgae
      is Right Imgae
      is Left Image
      is Left Image
      is Left Image
      Start Iterating Combination, please wait .....
      GOOOOOOOOOD! Here is the final result
      <Figure size 1800x1200 with 0 Axes>
```

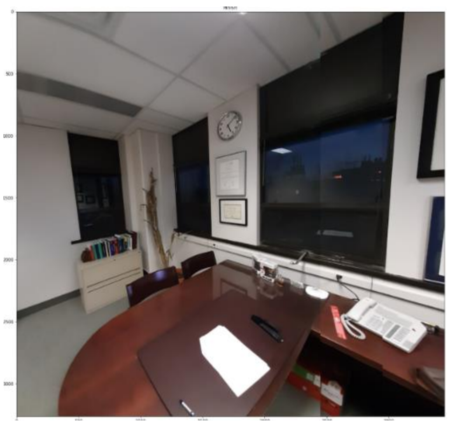


Figure 3: Third method's result and runtime

Detail of implementations

Step1: Match Features

For matching features of the images, I provide 3 options as the feature extraction methods which are sifted, surf and brisk. **For this lab I only did a test on SIFT**, so please be aware of bugs in the other 2 methods. The matching metric that I used is the number of matches in each pair. I initialized a constant called BEST_MATC_METRIC. By manually changing the value we can obtain different numbers of a good matching image compared to our seed image, which is defined by a constant called SEED_IDX. For the data structure, I used a class to store all the key points, descriptors, matches and images. Using class as my data structure can avoid passing wrong parameters (such as using a list to store all the key points and descriptors for an image) and easy to do data transfer between different functions.

After finding all the good matches images, I perform a right list and left list classification which based on seed image, split images into "the left side of seed image" and "the right side of the seed image", and then ranking them based on the number of their matches with seed image.
The method I used to classify right, and left is to apply the homographic matrix to the point [0,0], remember to pad it to [0,0,1] since the homographic matrix is 3 X 3. After the point has been warped, just simply check the x sign of the point to distinguish whether it is left or right.

After this step, I obtain a seed image, a right good match image list and a left good match image list

Step2: Estimate Transformation

For this step, every version looks similar. By detecting the key points and descriptor, we can obtain the matches between them, then apply a function from OpenCV called **cv.findHomography** to obtain the homographic matrix, then use **cv.warpPerspective** to get the final transformation result.

Step3: Merge Images

This would be the hardest part of this project, which result in 3 different versions of my code, which has a different merging algorithm. I will only go through the final method that I used in this project:

For version 2.0, What I do for a pair of images is to make one image as **imgMain**, and one image as **imgWarp**. I will apply the homographic transformation to the **imgWarp** and then copy the **imgMain** into the left side of the warped image to create the final image. Version 2.5 is different from 2.0 in this step: It applies and copies the image pixel by pixel to the left side of the warped image by comparing the difference of colour. This takes much longer time than version 2.0, but it has a very strong radiometric resistance. Version 2.5 also brings another issue: remnants of the merged parts, which will be noticeable in the large barrel deformation image pair which can be seen in **the final result V2.5 images 1 and 2**
The stitching order that I apply to the final method is to stitch all the images into the left list iteratively, which means I stitch image 1 and image 2 in the left list into a combination image, then stitch the combination image with image 3 in the left list, then do it iteratively until I finished all the

image in the left list. By doing the same thing with the right list, I can obtain three images which are the seed image, right combination image and left combination image. First, I do stitches between seed and right combination, seed and left combination. After that, for the final two images, I would perform the final stitches and complete the result. The reason why I combine two images with the seed image in the last few steps is that if you try to combine the right and left directly, you will receive an error from OpenCV that indicates there are less than 4 descriptors between those images. Therefore, combining them with the seed image in the final step can provide more descriptors for the final combination.

I also implemented some optimization methods: The first is chopping out the black side. This function can greatly reduce the pixel since after warping, the image will be way below the buffer that we provided, therefore chopping out the black side can save a lot of time when doing iterating stitching. The second method is the final refine or the final warping. Since the basis of my logic is the warping left and right together first and then stitching them together with the "help" from the seed image, the final shape will usually look like an upside-down "V" with a much longer arm on the right side. Therefore, by doing the final warping we can obtain a much better result.

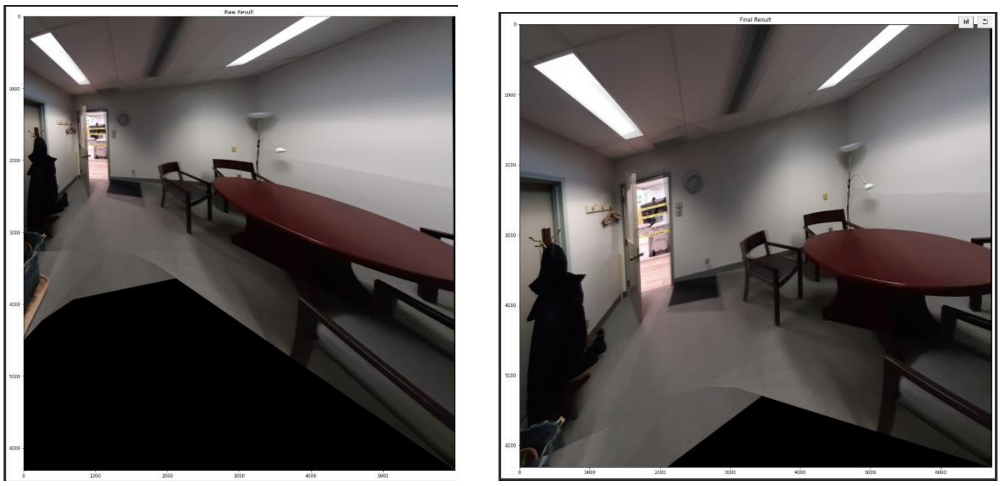


Figure 1: These images show the function of final warping

Discussion and improvement

Overall, I think I had a decent result for this specific scenario *office2*. In the D3 folder, I put all surroundings into 3 results and they all look good. However, there is still a lot of improvement space for my code. If I have more time, I will try to implement the method that can find the seed image by itself, by iterating through all the pairs (i.e for 29 images will be 29 X 29 pairs) to find the best center image or seed image. And for the right and left classification, I can only do left and right instead of all directions. Finally, the most important thing is that I should find a method to obtain the camera parameter to do pre-calibration to all the images since from the result I can find that the distortion is so intense.