Threshold = 0.3

Threshold = 0.5

Threshold = 0.6

Threshold = 0.2

```
---------------------------------------------------------------------------
--
AssertionError                            Traceback (most recent call last)
<ipython-input-64-6fa78382a76b> in <module>
      1 # Test run matching with no ransac 1.3
      2 plt.figure(figsize=(20, 20))
----> 3 im = Match('./data/scene', './data/book', ratio_thres=0.2)
      4 plt.title('Match')
      5 plt.imshow(im)

<ipython-input-47-527ad4660e3d> in Match(image1, image2, ratio_thres)
     38     matched_pairs = [
     39         [keypoints1[i], keypoints2[j]] for (i, j) in matched_pairs]
---> 40     assert len(matched_pairs) > 0, "No match received"
     41     im3 = DisplayMatches(im1, im2, matched_pairs)
     42     return im3

AssertionError: No match received


<Figure size 1440x1440 with 0 Axes>
```

1.3:

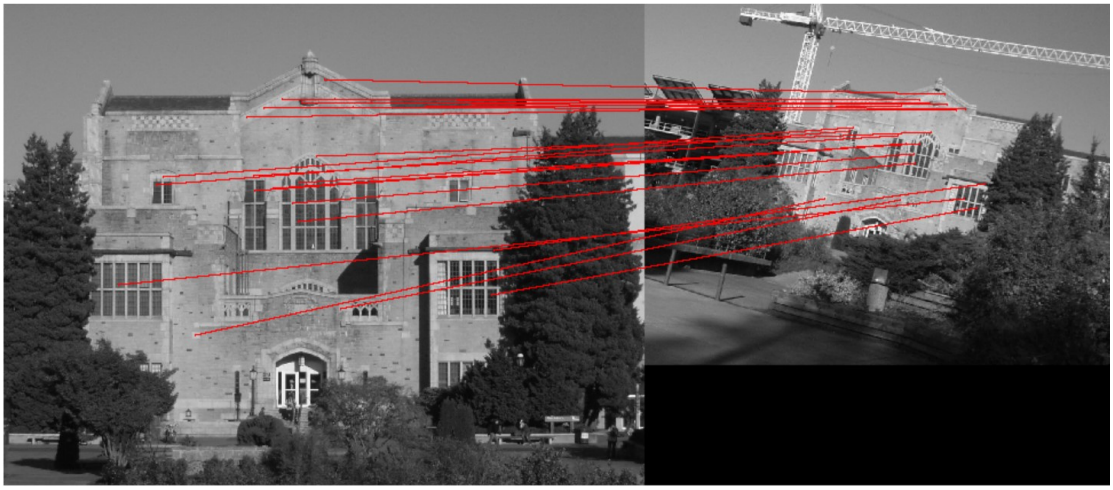
Conclusion:

For the the images “scene” and “book”, the best threshold is 0.5, with many lines in the image but matched well (looks). If the lines are less than 10, I found one, threshold is 0.3.

When threshold is too small, like 0.2, there will be no matched found, hard to find matched locations.

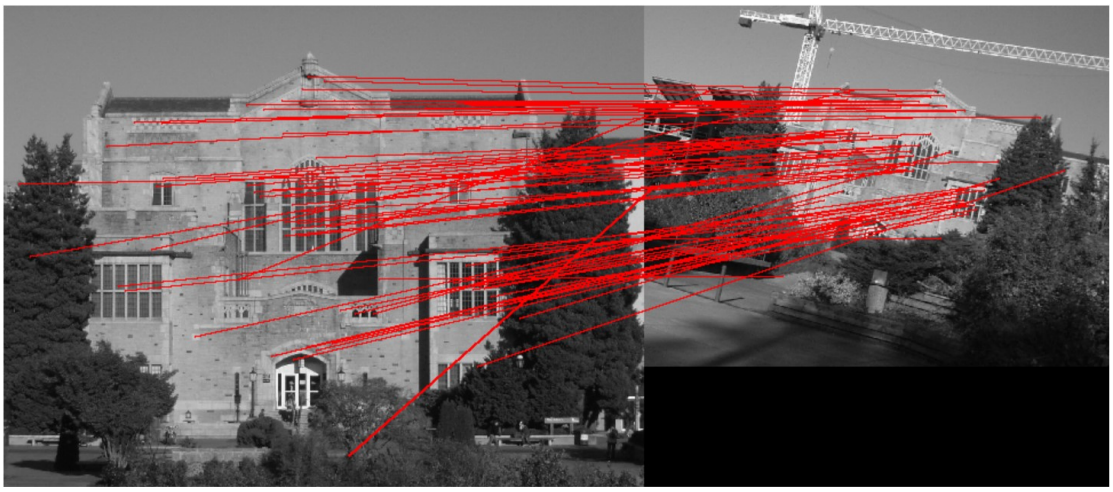
When it is too large, like 0.6, many matched lines but some are not supposed to be, not the matched lines. The mismatched lines appear.

ratio_thres=0.6 orient_agreement=30 scale_agreement=0.5



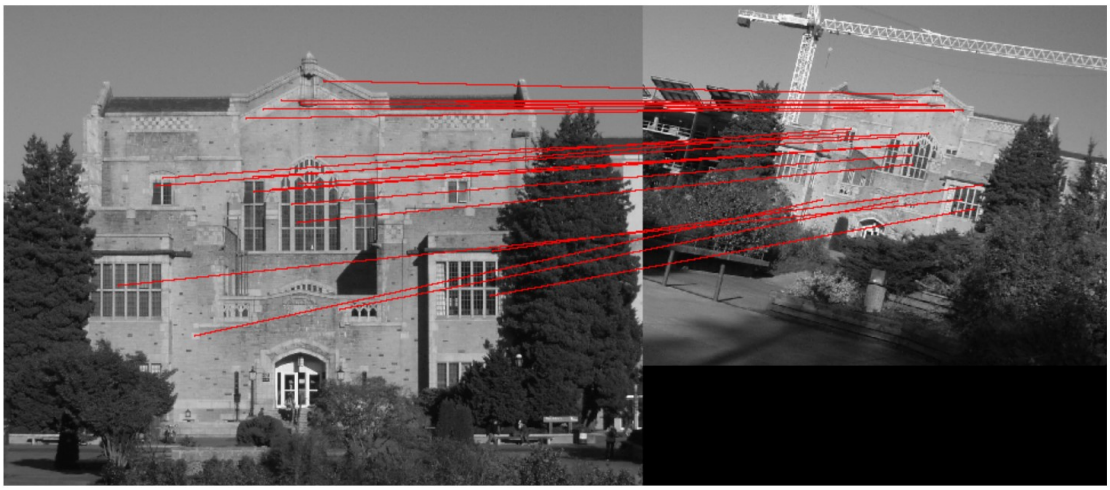
ratio_thres=0.8 orient_agreement=30 scale_agreement=0.5

(at least one line is wrong, for the grass before the labrary)



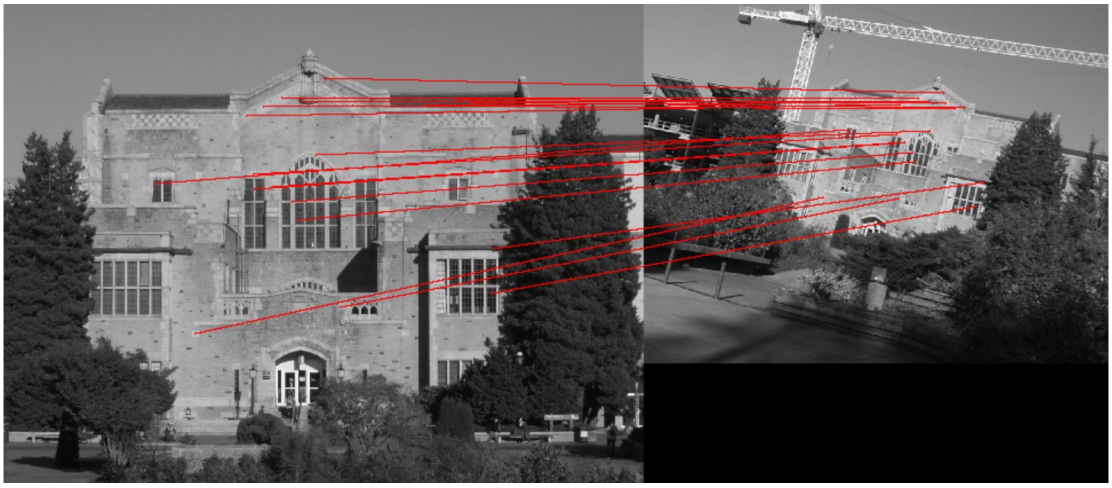
ratio_thres=0.6 orient_agreement=180 scale_agreement=0.5

(one match line is wrong, at the window)



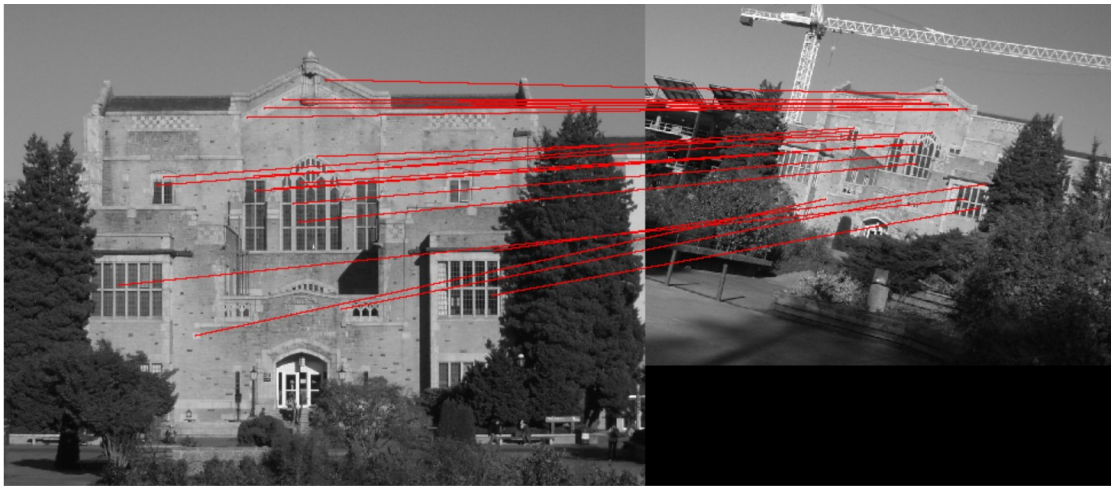
ratio_thres=0.6 orient_agreement=30 scale_agreement=0.1

(looks more precise)



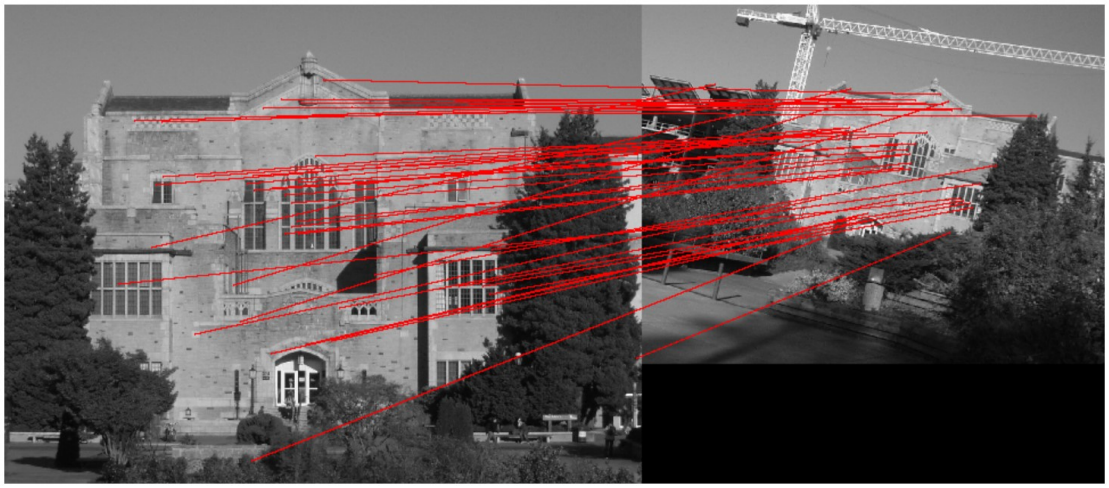
ratio_thres=0.6 orient_agreement=30 scale_agreement=0.4

(looks ok)



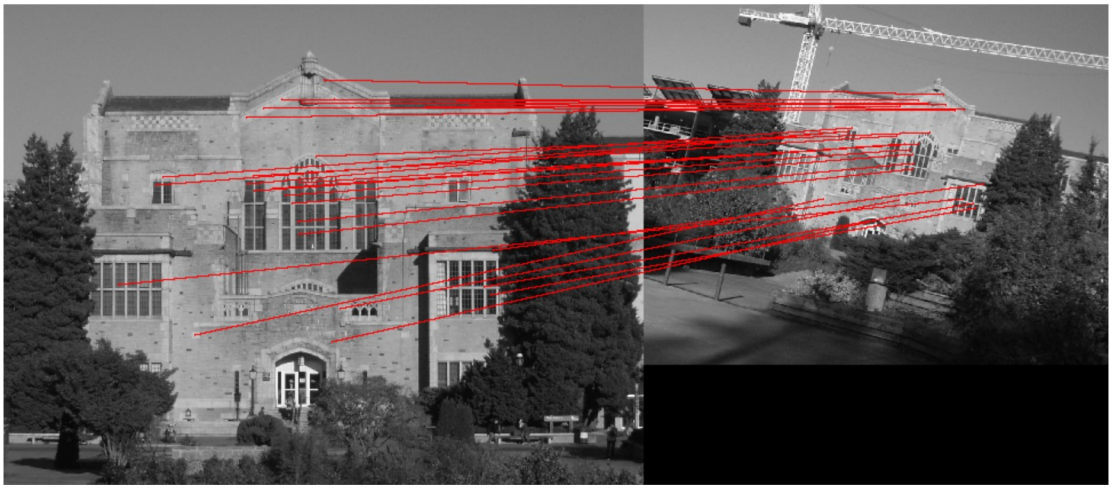
ratio_thres=0.7 orient_agreement=360 scale_agreement=0.9

(oh,no!)



ratio_thres=0.62 orient_agreement=30 scale_agreement=0.6

(best)



1.4:

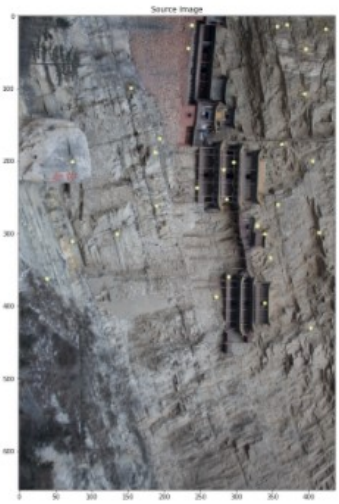
Conclusion: The best one I found is ratio_thres=0.62 orient_agreement=30 scale_agreement=0.6.

Among these parameters, the ration threshold matters much. It changes the number of matched lines, when it large than about 0.64, the mistake matched line appear. So larger threshold ratio means more matched lines as well as the mistakes. While, smaller matched lines can't represent the similarity much, so small threshold ratio is also not welcomed.

For the two agreements, *_agreement represent the tolerance, it turns out if the values of *_agreement are too large, the precise will be less, which means the mismatched lines will appear. They are also can't be so small, if so, the matches lines reduced and won't represent match relations well.

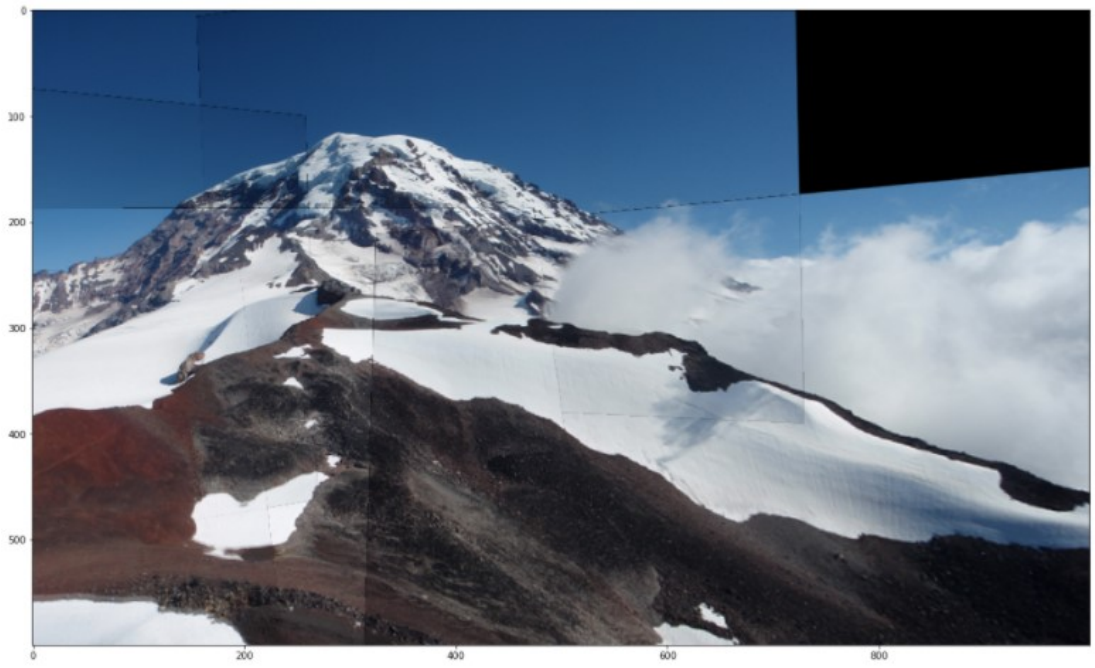
2.3

Press any key to exit the programa

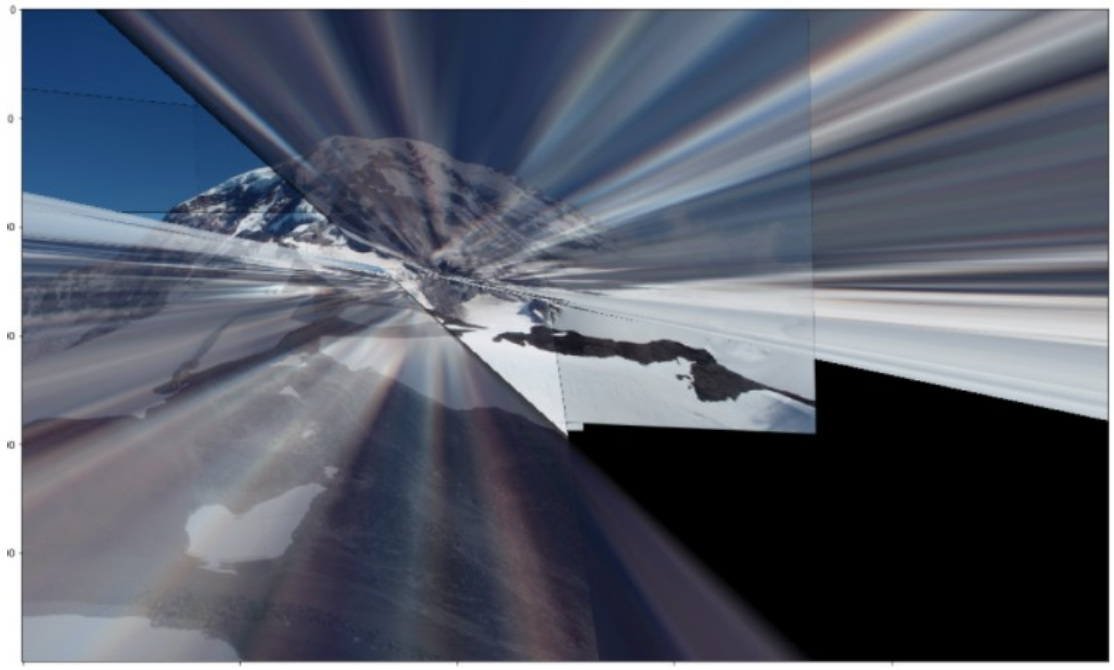


2.4

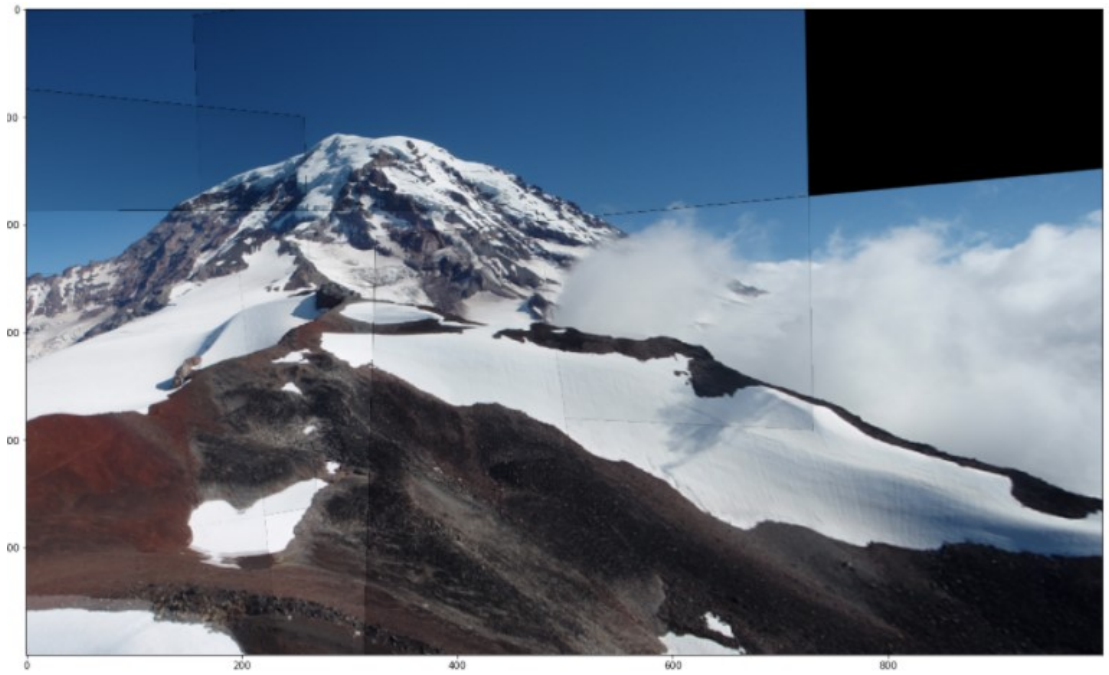
Given data



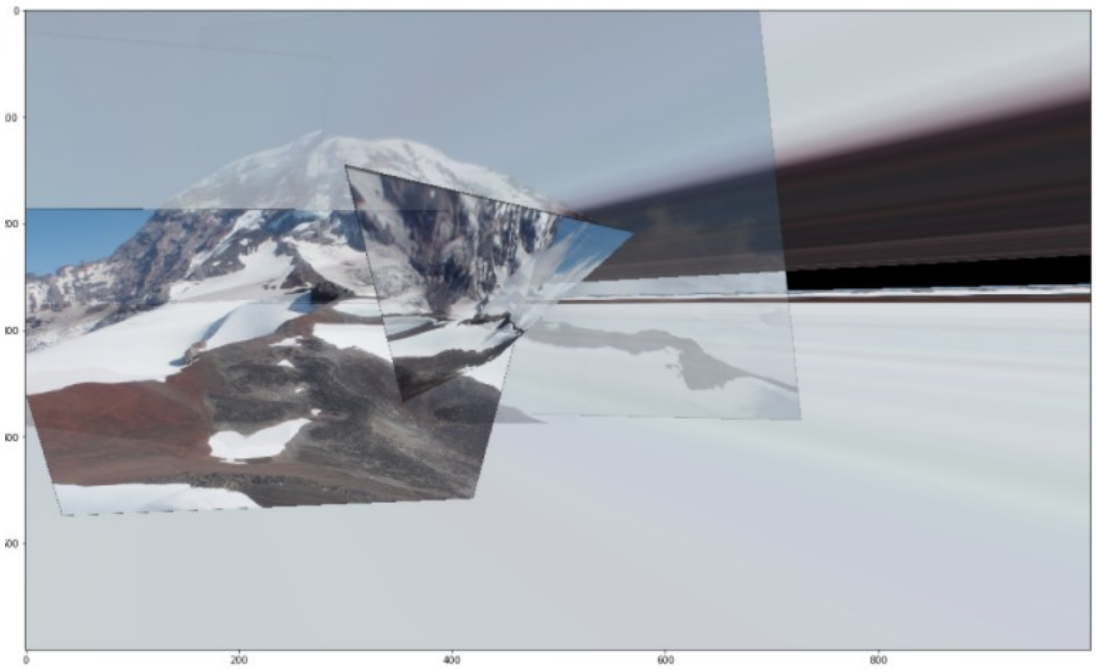
num_iter = 20 tol = 10 ratio_thres = 0.9



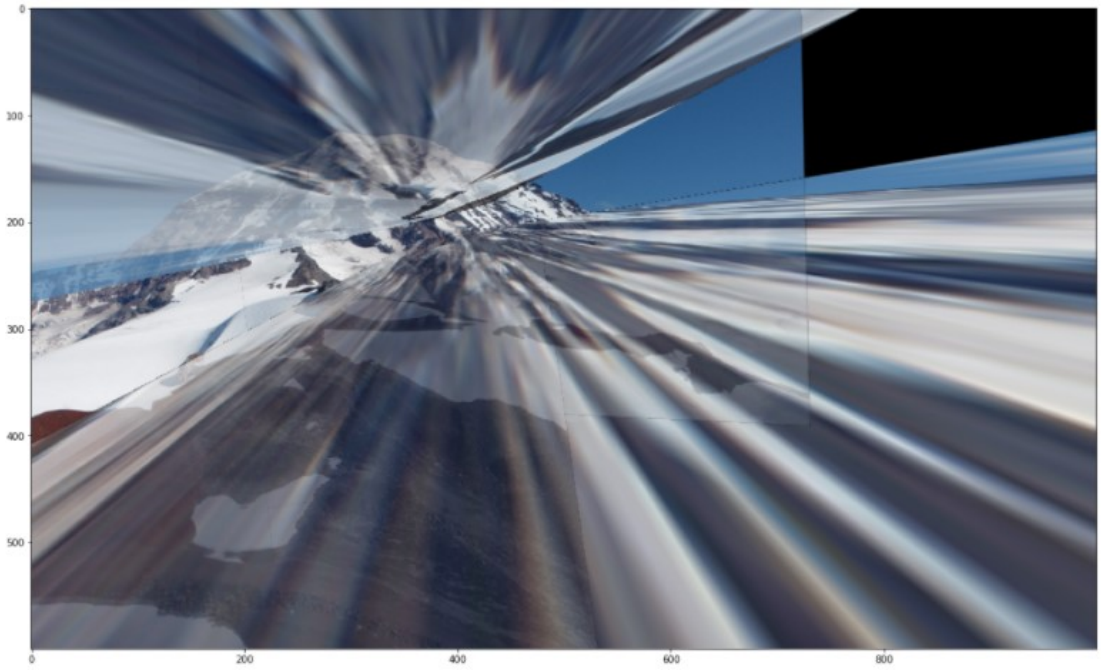
num_iter = 200　tol = 10　　ratio_thres = 0.9



num_iter = 50　tol = 100　　ratio_thres = 0.9



num_iter = 50 tol = 10 ratio_thres = 1.0



Conclusion:

If number of iteration is too small, the mix of projection of different images will happen. If number of iteration is big, it is fine, the picture looks still well, just waste time to run.

If the tol is too large, the color of the pictures will mixed, if it is too small, nothing series happened.

If the ratio of threshold is too large, pictures' edge is obvious and some pictures looks

warping. If it is too small, the number of pictures is not enough to consist the whole one.

2.5

```
path = './data/'

canvas_height = 1125
canvas_width = 1500
image_list = ['fountain4', 'fountain0']

num_iter = 100
tol = 10
ratio_thres = 0.8
image_list = [op.join(path, im) for im in image_list]
create_pano(image_list, ratio_thres, canvas_height, canvas_width,
            num_iter, tol, figsize=(20, 20))
```


```
path = './data/'

canvas_height = 1125
canvas_width = 1500
image_list = ['garden0', 'garden3','garden4']

num_iter = 200
tol = 15
ratio_thres = 0.8
image_list = [op.join(path, im) for im in image_list]
create_pano(image_list, ratio_thres, canvas_height, canvas_width,
            num_iter, tol, figsize=(20, 20))
```


```
path = './data/'

canvas_height = 1125
canvas_width = 1500
image_list = ['irving_out3', 'irving_out6','irving_out5']

num_iter = 150
tol = 10
ratio_thres = 0.9
image_list = [op.join(path, im) for im in image_list]
create_pano(image_list, ratio_thres, canvas_height, canvas_width,
            num_iter, tol, figsize=(20, 20))
```


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400