README.md

VFX Project 2: Automatic Image Stitching

Author

- 網媒所 王哲瑋 Che Wei Wang r10944037

Code Structure

- `run.sh` : run `main.py` with predefined arguments.
- `main.py` : will handle command line arguments and then `import msop`, `import stitching` and `import utilities` to do image stitching.
- `msop.py` : implementation of MSOP feature detecion, description method.
- `stitching.py` : implementation of KNN-feature-matching, image-alignment, image blending
- `utilities.py` : handle some trivial functions.

How To Execute

- Data format:
    - Images should be named from 0 to N, where N is the number of your images.
    - Focal-length file should list focal-lengths of images in the structure of one line one focal-length, see the provided data as examples
- Execution methods:
    i. Execute `run.sh`.
    ii. Run `python main.py` with arguments:

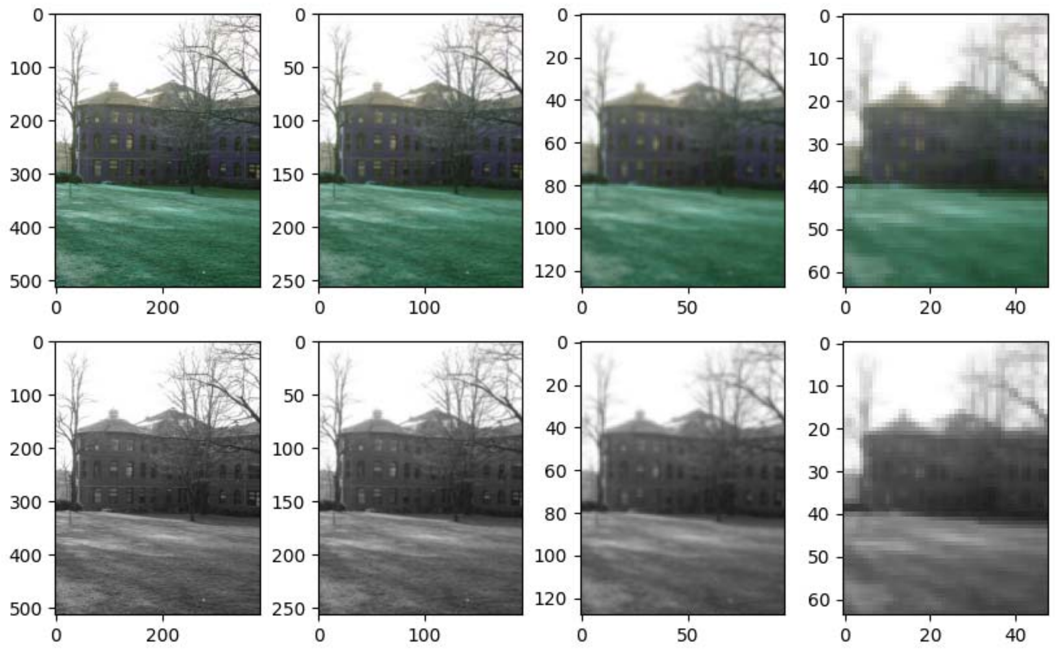
Argument	Explanation	Required
input_dir	The directory where you put all your input images	Yes
image_extension	The extension of your image files	Yes
image_num	The number of your input images	Yes
focal_length_path	The path to your focal-length file	Yes

Feature Detection

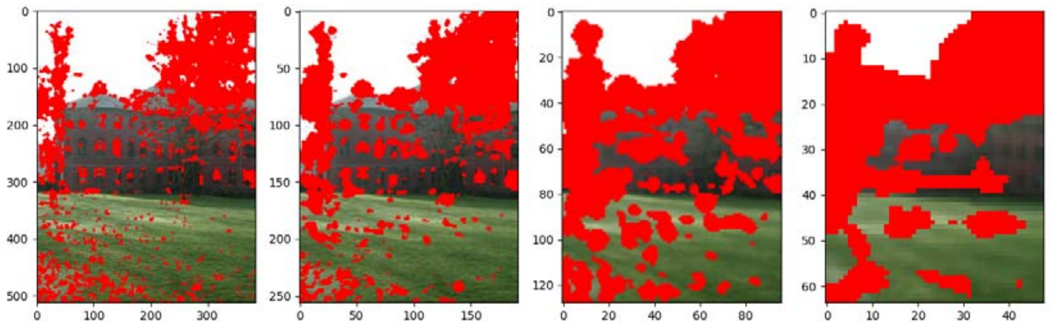
The feature detection algorithm being implemented in this project is MSOP's feature detection, which is multi-scale Harris Corner Detection.

The following steps under "Feature detection" section have to be done for every input image. Here I use one image as an example.

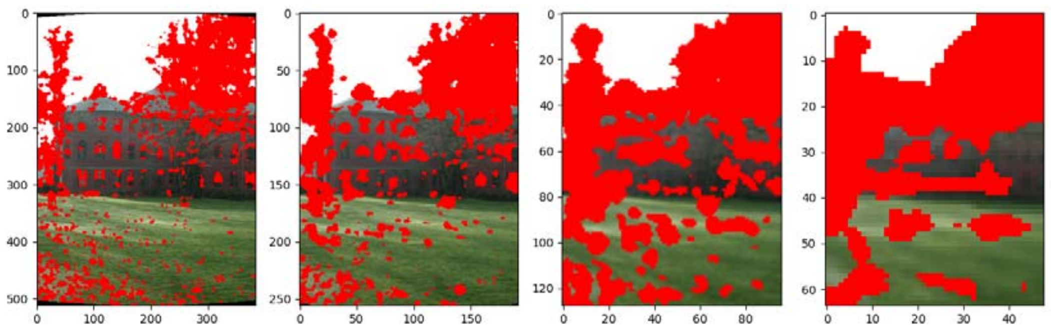
The first thing I do is scale input images into 4 scales: 1, 1/2, 1/4, 1/8, and also the grayscale image of them.



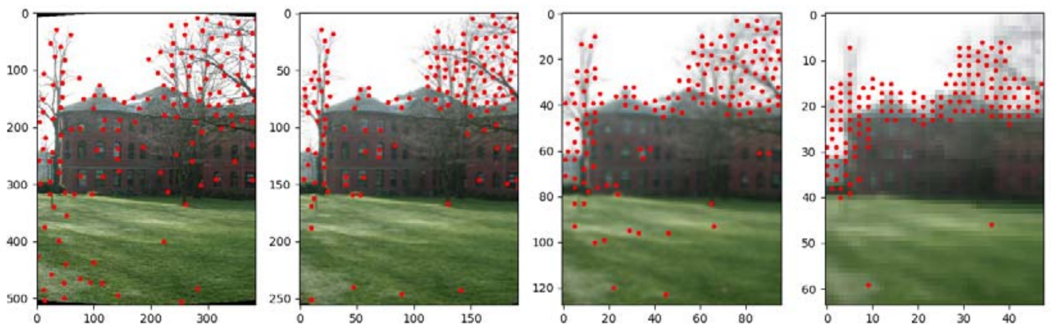
Then, I compute the Harris Corner Response for all scales of the image. Below is a image that shows the pixels of higher Harris Corner Response as red.



Next, I do cylindrical projection on the image. The below image shows the projected images with Harris Response



Then, I pick the feature points using non-maximal suppression methid to try to pick well-distributed feature points.



Feature Description

After I pcik the feature points for every scale of every image, now I can make descriptions for them.

The following steps under "Feature Description" section have to be done for every input image. Here I use one image as an example.

To describe a feature point, I first decide the orientation of it.

The orientation of a feature point is defined by first doing a Gaussian Blur on the image to decrease noise, then compute the gradient at the feature point and use the gradient as the orientation of it. Like the below formula.

$$\mathbf{u}_l(x, y) = \nabla_{\sigma_o} P_l(x, y)$$

$$\sigma_o = 4.5$$

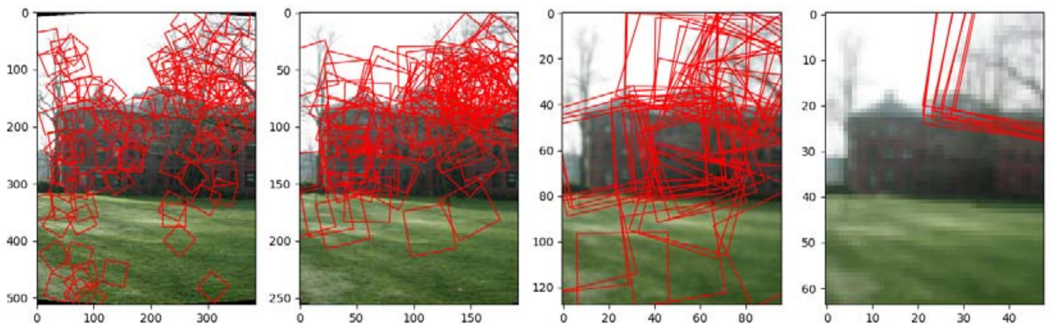
$$[\cos\theta, \sin\theta] = \mathbf{u}/|\mathbf{u}|$$

Then I rotate the image to align the orientation of the feature point toward right and then sample a 40*40 patch around the feature point.

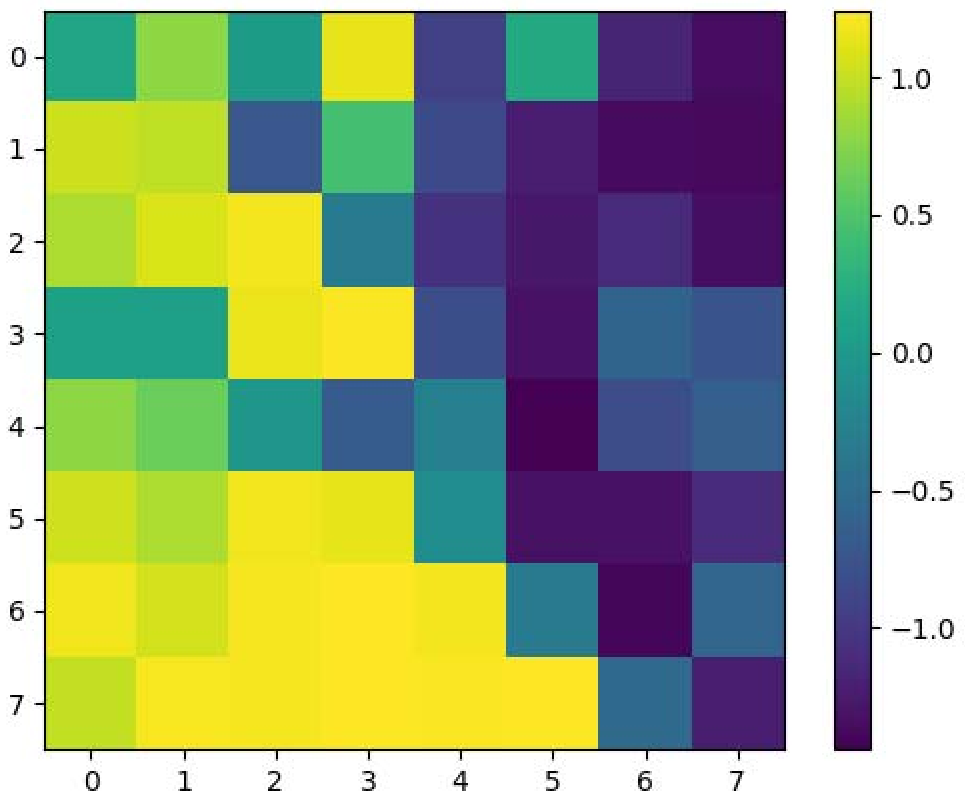
If a feature point does not have a complete 40*40 patch around it, like sitting on the eade of the image, then the feature point will be ignored.

The 40*40 patch is then standardized and resized to an 8*8 patch. This 8*8 patch will be the description of this feature point.

Below image shows how feature point gets its description patch.



Below is an example of how a 8*8 description patch looks like.

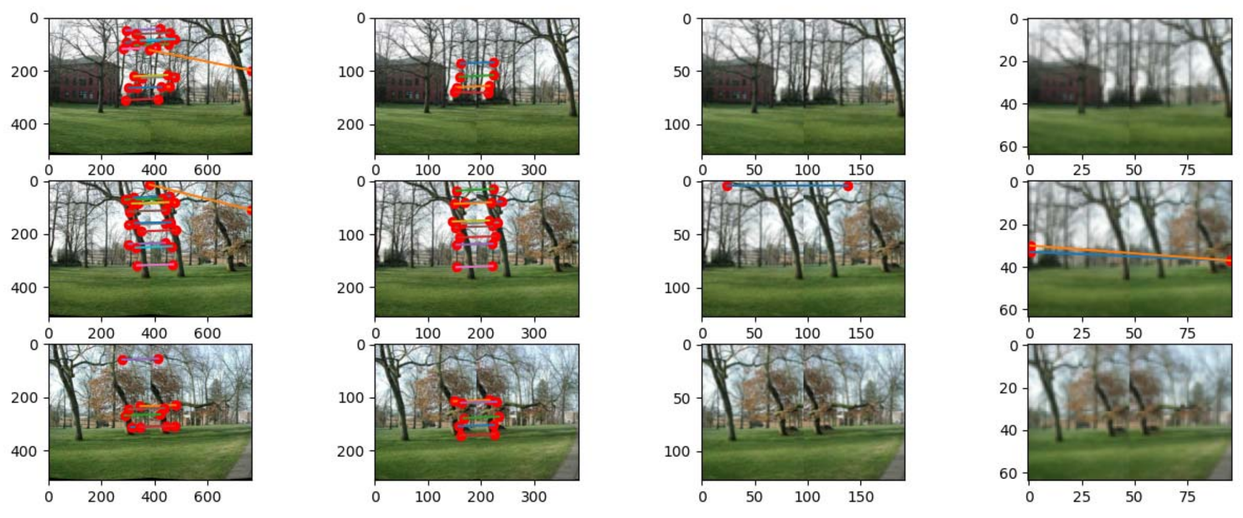


Feature Matching

After I got the descriptions of every feature point of every image, I want to match the feature points between every consecutive images.

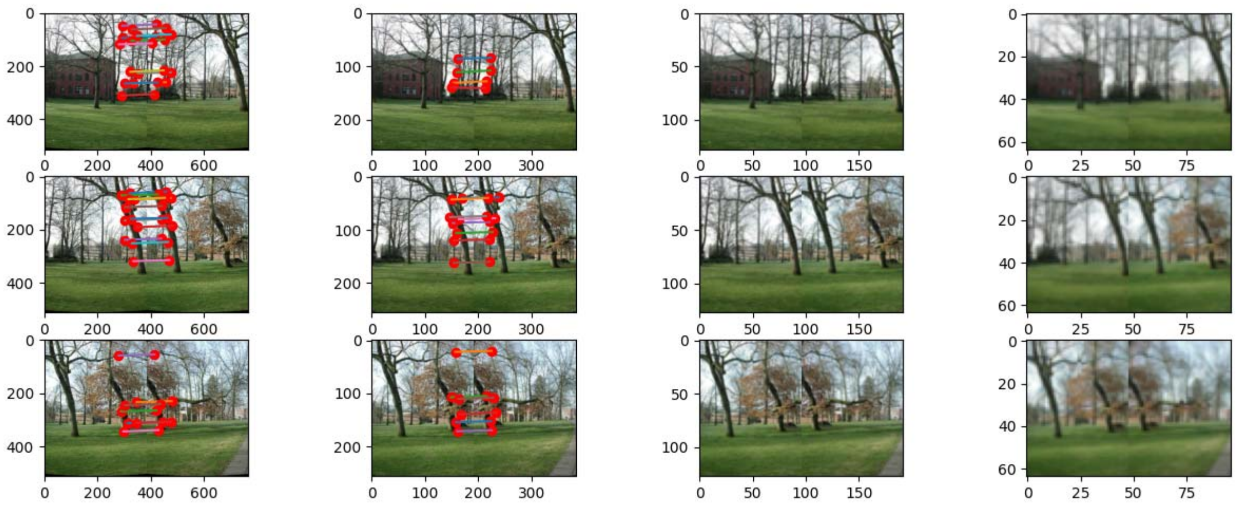
The following steps under "Feature Matching" section have to be done for every consecutive images. Here I use one image pair as an example.

To match feature points, I use KNN algorithm, and below is the result:



You can see there are some obvious matching errors, and those errors happen because those feature points sit on the edge of the image, and dont have a complete 40*40 description patch.

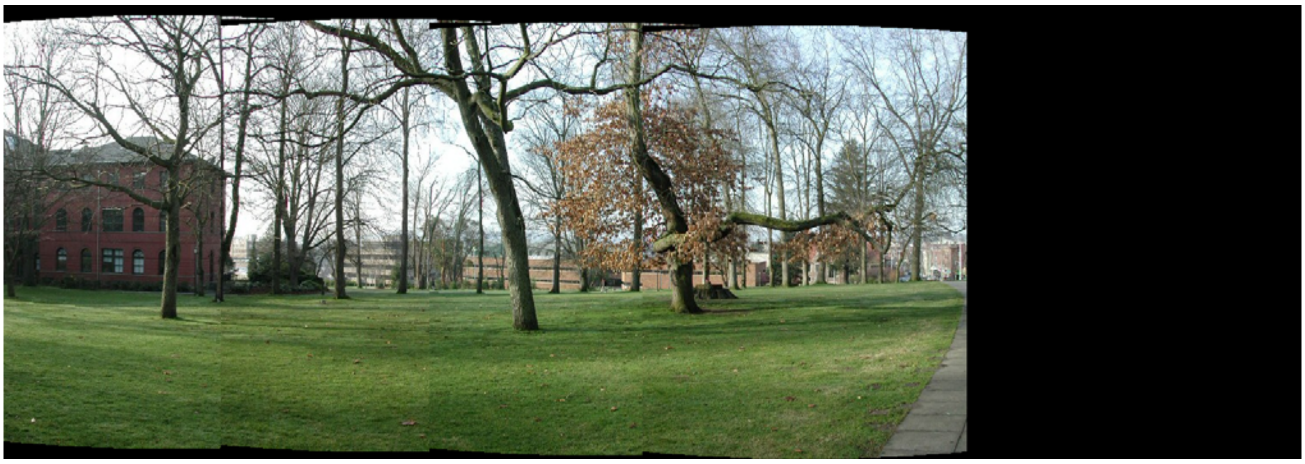
So like I said in previous section, I ignore those kind of feautre points, and this is the result:



Much better, right?

Image Alignment

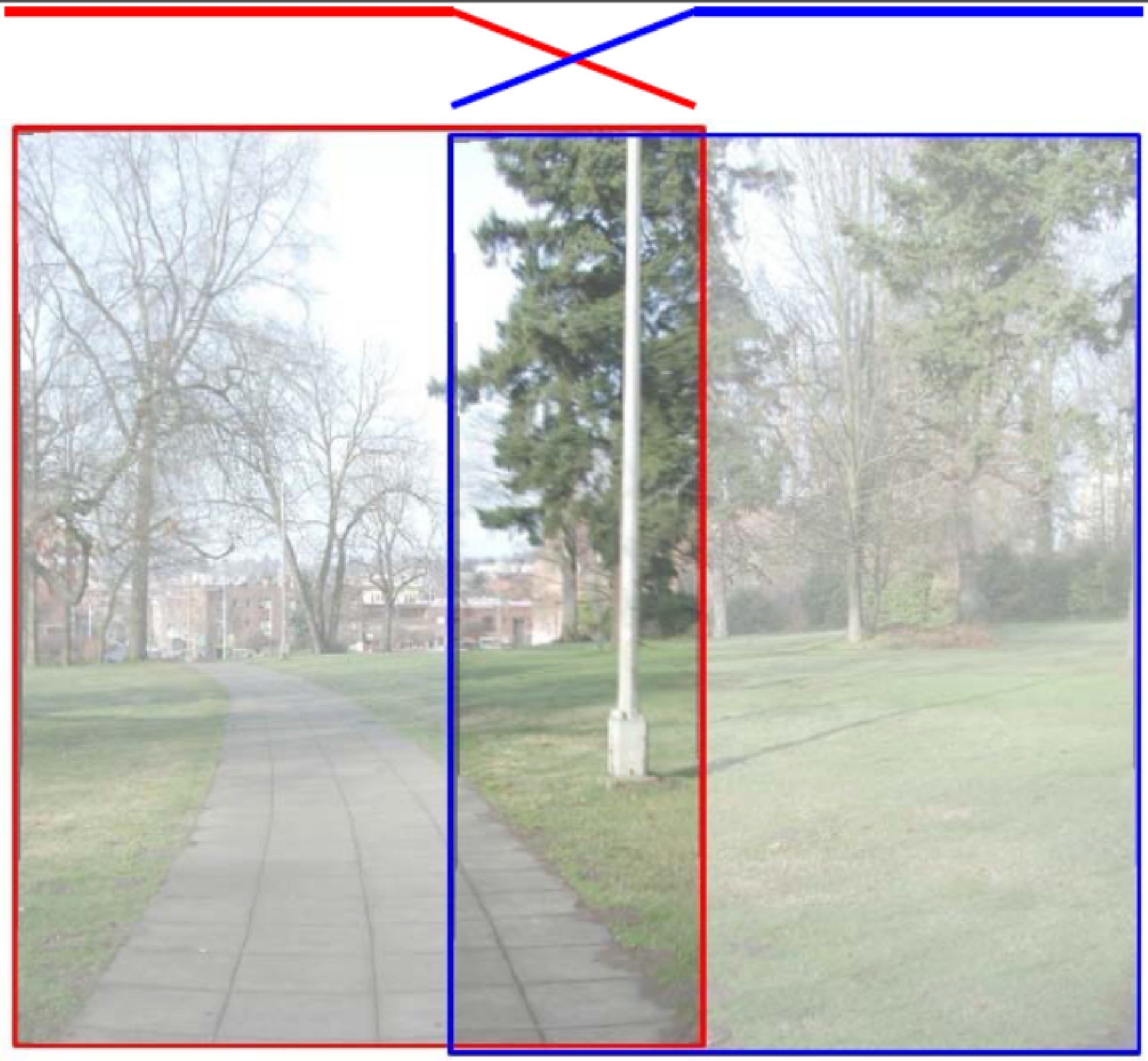
After we match the feature points between each consecutive images, we can compute the transform between each image and then align them together.



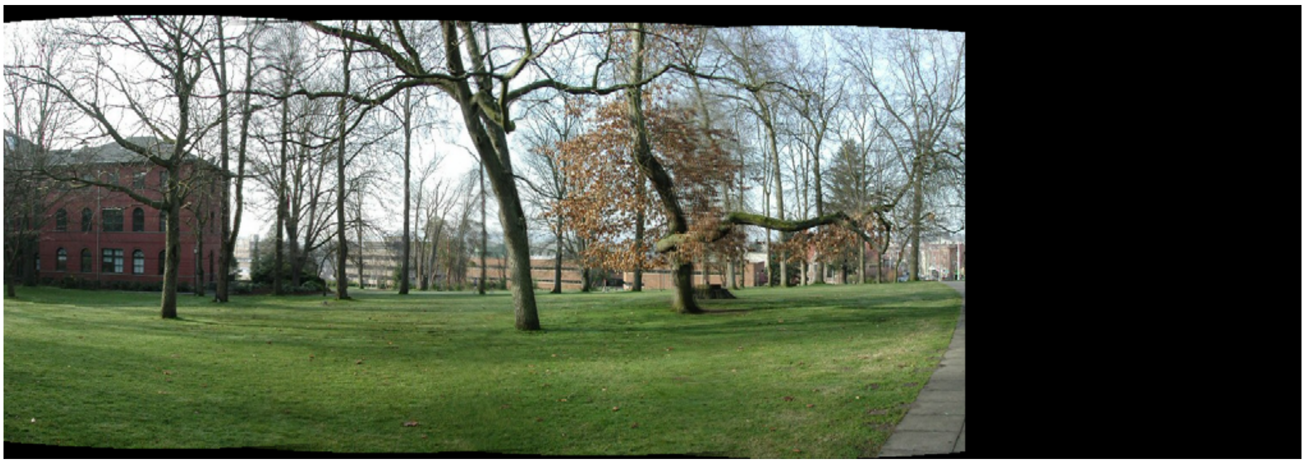
Hmmm, the image are aligned correctly, but the color between them are not merged well. We can easily see the connection boundary between them.

Image Blending

To solve the "color between images are not merged well" issue. I use linear interpolation to blend the colors between images, illustrated as below image.



Then I get a much better result:



Result

Here are the final complete results of the stitched images: