VFX HW2

Group 9:
R11942163 電信碩一 陳詠源
R11942161 電信碩一 顏子鈞

Description of this project

The objective of this project is to implement feature-based image stitching algorithms to synthesize a panoramic image from a set of images taken consecutively in time and space(single direction). To achieve this, we will implement the following components to process the input images: feature detection, feature matching, image matching, and image blending. This project covers techniques such as cylindrical projection, SIFT, and RANSAC which are necessary to complete our objective of constructing a panoramic image.

Description of the algorithm implemented

There are 6 pipelines in our implementations:

Pipeline	**Algorithm / Method**
Cylindrical projection	Warp image onto a cylindrical plane (focal lengths are estimated by AutoStitch)
Feature detection	Harris Corner Detection
Feature descriptor	SIFT descriptor
Feature matching	L2-distance brute force
Image matching	Best match with minimum cost (compute with keypoint neighborhood)
Image blending	Linear blending at the intersection between 2 images

Implementation Details

Cylindrical Projection

For the convenience of subsequent operations, we project the input images onto cylindrical coordinates, so that our matching process only involves translation. The focal length f of the image is estimated using the Autostitch software, and the following formula is then used for projection for each (x,y) coordinate:

$$x' = f * arctan(x/f)$$

$$y' = f * y/(\sqrt{x^2} + y^2)$$

But before this, we have to set the center of the image (H/2, W/2) to (0,0). Then after applying the formula above, we add the offset back.

Now that we have every coordinate transformation result, we can apply forward wrapping.

Feature detection

We select corners as our features/keypoint using Harris corner detection.

0. Convert RGB image to grayscale
1. Apply gaussian blur to the image
2. Compute horizontal and vertical derivative of the image, Ix, Iy
3. Compute the product of derivative, Ixx, Iyy, Ixy
4. Compute the sums of the products of derivatives
5. Define the matrix M at each pixel, and compute the response R of the detector at each pixel, $R = det(M) - k * (trace(M))^2$, where we set k = 0.04
6. Threshold on value of R, compute non-maxima suppression. Only points with intensity > 0.01*max(R) will be selected as the keypoint.

Feature descriptor

For each keypoint detected from Harris corner detection, we calculate its corresponding SIFT descriptor.

1. Obtain the image patch of size (16x16) centering at the keypoint
2. Calculate its main orientation:
   Compute the horizontal and vertical gradient of the image patch, calculate its magnitude and angle:
   $$mag = \sqrt{Ix^2 + Iy^2} \qquad angle = arctan(Iy/Ix)$$
   For each pixel of the image patch, we add its mag to the corresponding $angle$ bin of the orientation histogram. Finally, we return main orientation as the bin with highest mag.
3. Rotate the image patch according to its main orientation.
4. Divide the image patch into 16 sub-patches of size (4x4). For each sub-patch, calculate its 8-bin orientation histogram (every bin represents 45 degree). Hence, we will have 16 subpatches and 8 orientation each, which can be described by a 128-dimension array.
5. Normalize the 128-dimension array. If any entry still has a value larger than 0.2, clip that entry into 0.2, and normalize the 128-dimension array again.

Feature matching

For every SIFT descriptor (corresponding with its keypoint), we use brute force to calculate if there exists any matched descriptor: 2 descriptors are counted as matched, only if their L2-distance are smaller than $threshold$*(the second smallest L2-distance). $threshold$ is an adjustable parameter, and it is different for every dataset.

Image matching

We have multiple matches, for each match coordinate pair (x1,y1,x2,y2), we can compute the translation offset (for right-hand-side image to translate and warp on left-hand-side image):

$x\,offset = -x2 - (W1 - x1)$, where W1 is the width of left image

$y\,offset = y2 - y1$

We select the optimal match with the minimum cost. The cost is calculated as below:

1. Obtain the neighborhood window of (x1,y1) and (x2,y2), denoted as $window1$ and $window2$
2. Random sampling N points inside the window.
3. cost = sum (abs ($window1$[random_points] - $window2$[random_points]))

To accelerate the process, we will discard any matches with:

- $y\ offset$ > offset_threshold1, (set to 75 pixel at most)
- $x\ offset$ > offset_threshold2, (set to around W/2)

as we observed the translation offset is not very large.

Image blending

For now, we already computed the translation offset between every adjacent image pair, the next thing to do is to stitch all of them together.

For the regions where every adjacent image pair has no intersection, we simply just warp the image according to the translation offset.

For the regions where every adjacent image pair intersects with, this is where we perform blending:

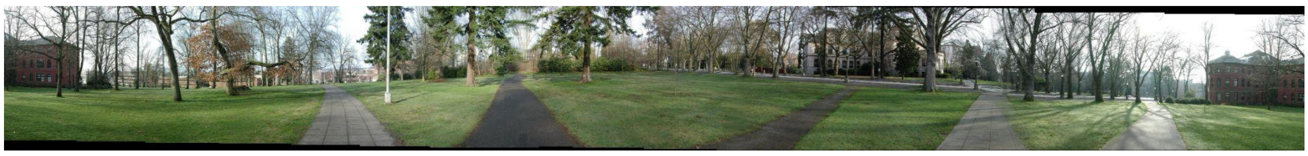
1. We obtain the intersection region of the left-hand-side image, denoted as $intersection_{left}$, then element-wise multiply it with a mask. The mask's value is a linspace of 1.0 to 0.5 (from left to right), which specifies the blending weight applied on $intersection_{left}$.
2. We obtain the intersection region of the right-hand-side image, denoted as $intersection_{right}$, then element-wise multiply it with a mask. The mask's value is a linspace of 0.5 to 1.0 (from left to right), which specifies the blending weight applied on $intersection_{right}$.
3. The final intersection region will be replaced by $intersection_{left} + intersection_{right}$.

Finally, we will trim the panorama size (as we initial it very large) to fit the actual images. No further bundle adjustment is done.

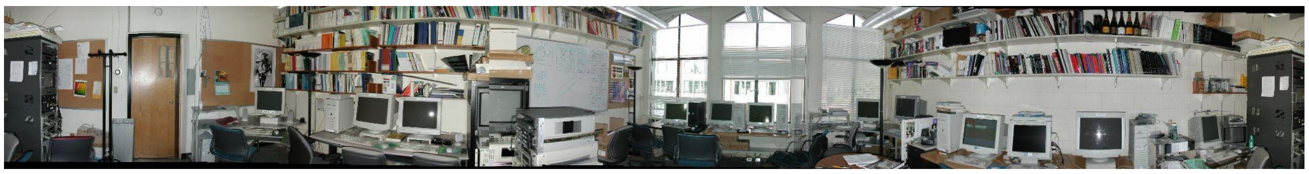
Experiment and Comparison

1. Dataset taken with tripod:

- parrington (18 images)

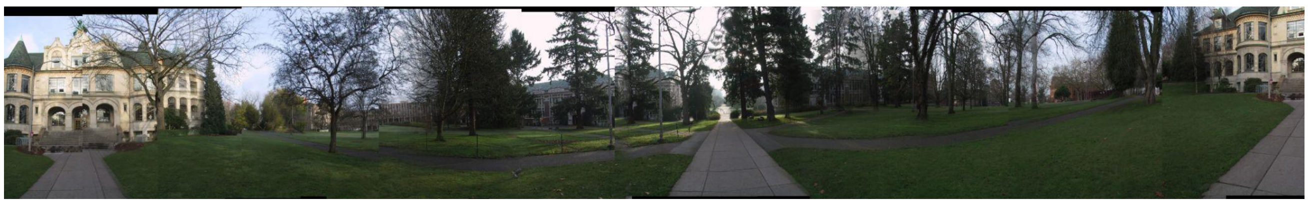


- grail (18 images)

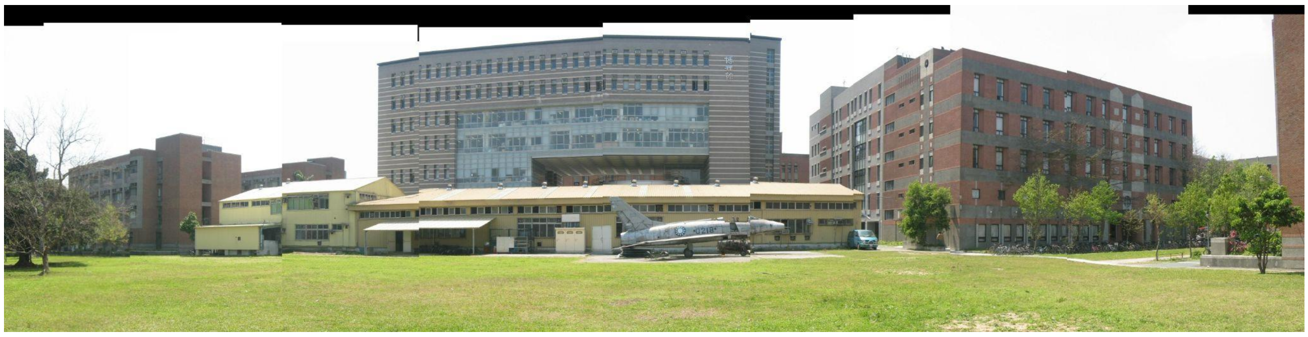


2. Dataset taken without tripod:

- Denny (15 images)



- csie (11 images) (no blending)



In general, panorama of dataset taken with tripod has better quality, as it is easier to perform stitching. We can observe that the panorama of Denny has some distortion on the left side:

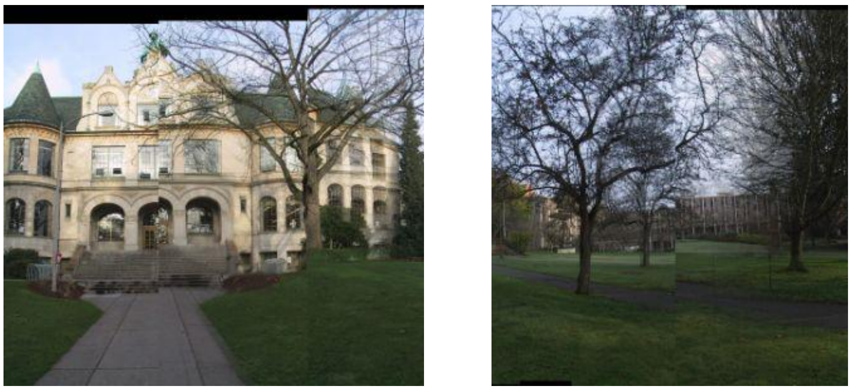


Fig: Zoom in of panorama (denny)

Besides that, we also observed that image blending can improve our panorama's quality, but sometimes it can make things worse, for example the csie dataset with blending looks very weird:

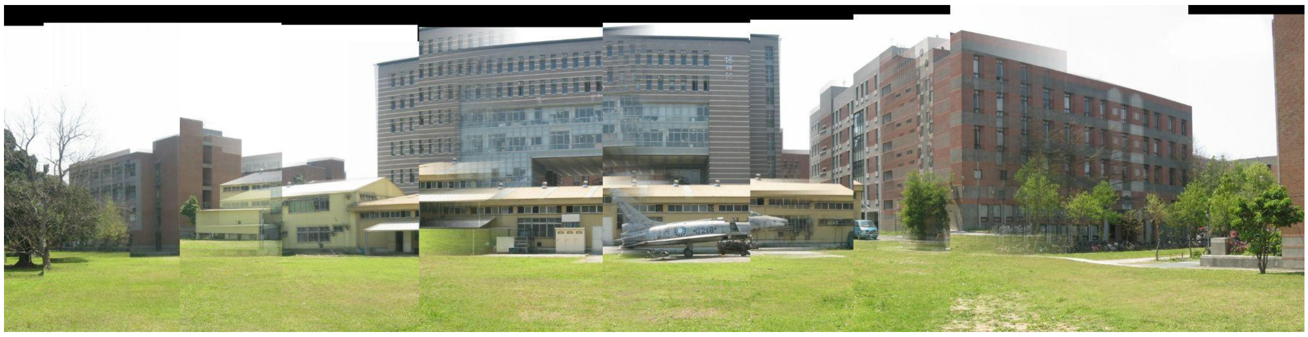
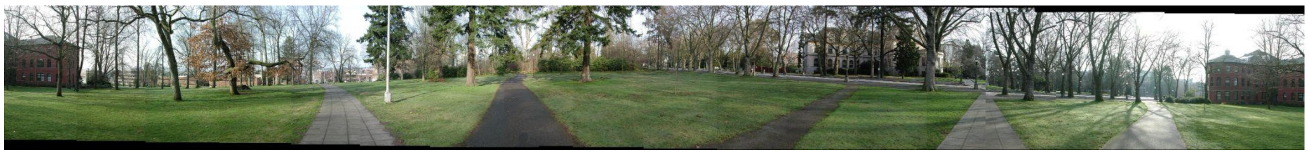
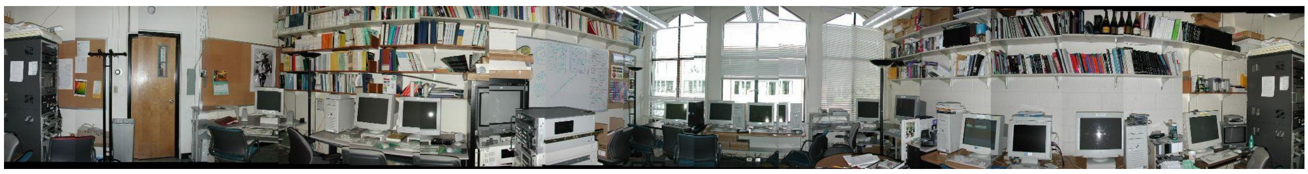


Fig: Csie panorama with image blending

But nonetheless, it has more goods than its side effects. We showed that the other 3 datasets without image blending has some obvious discontinuity effect:





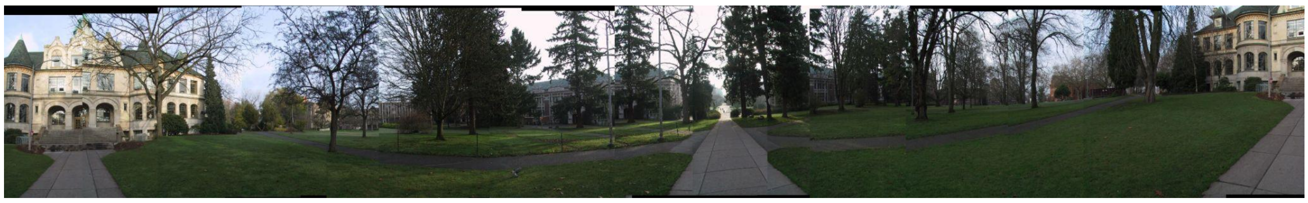


Fig: panoramas without image blending

What we have learned from this project

The task of synthesizing panoramic images is an application of image stitching that requires many steps to complete. We have learnt some useful experience from this project, the following are some notable examples:

1. Importance of using a tripod when collecting images. It can be seen from our results that datasets of images taken using a tripod produce panorama images of a better quality. In our method of only applying translation for the matching process, if the images were not taken using a tripod, the resulting panorama image will appear to have vertical offsets at the intersection/boundaries of two neighboring images. This problem is less obvious when our method takes images captured using a tripod as input. It is possible that the translation stage is not as robust when it comes to matching points with vertical differences, hence there is room for improvements. There could be other methods to overcome this problem programmatically, using not only translation as the matching process, or require inputs to be taken using a tripod.

2. Choosing best matching keypoint pairs as an optimization problem allows the use of different methods and not be limited to RANSAC. In our method, we computed the minimum cost with keypoint neighborhood.