

COMP 6341 Computer Vision

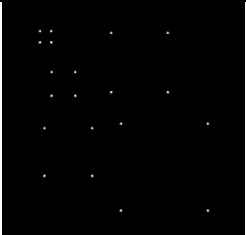
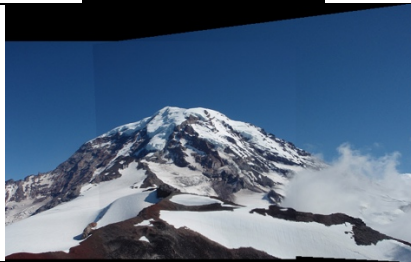
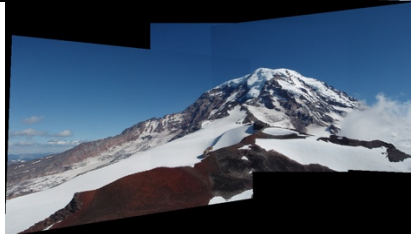
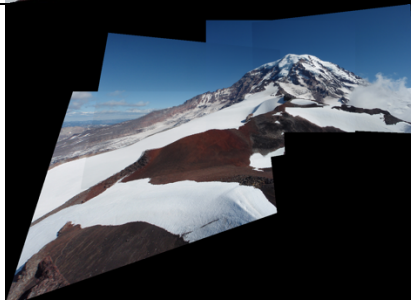
Project README

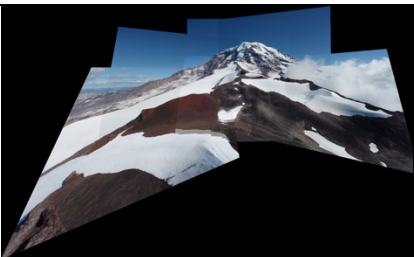
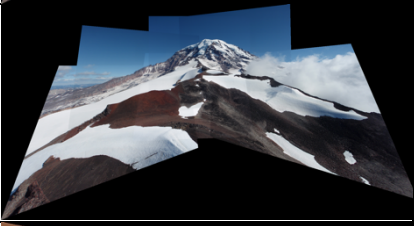

Student: Yixuan Li 40079830

1. Implemented Features

- 1) Normalized Harris corner detector, and save the result image.
- 2) Feature Matching between two images, and save the result image.
- 3) Compute the homography and save the result image after RANSAC.
- 4) Image stitching, and save the result image.
- 5) Stitching “Rainier1.png, ..., Rainier6.png” images. (Mandatory extra 1)
- 6) Stitching images taken by myself. (Mandatory extra 2)

2. Tuned Inputs

Testing Purpose	image paths input	Harris Corner Threshold	SSD threshold	Ratio test threshold	RANSAC inlier threshold	RANSAC iterations	Result panoramas
Harris Corner Detection	Boxes.png #	90	0	0	0	0	
ALL	Rainier1.png Rainier2.png #	80	1000	0.9	10	60	
ALL	Rainier1.png Rainier2.png Rainier3.png #	80	1000	0.8	4	150	
ALL	Rainier1.png Rainier2.png Rainier3.png Rainier4.png #	75	1200	0.9	3	300	

ALL (Mandatory Extra 1)	Rainier1.png Rainier2.png Rainier3.png Rainier4.png Rainier5.png Rainier6.png #	75	1200	0.9	1	350	
ALL (Mandatory Extra 1)	Rainier1.png Rainier6.png Rainier2.png Rainier5.png Rainier3.png Rainier4.png #	60	1200	0.9	1	1200	
ALL (Mandatory Extra 2)	my_images/ev1.png my_images/ev2.png my_images/ev3.png my_images/ev4.png my_images/ev5.png #	60	1500	0.9	4	300	

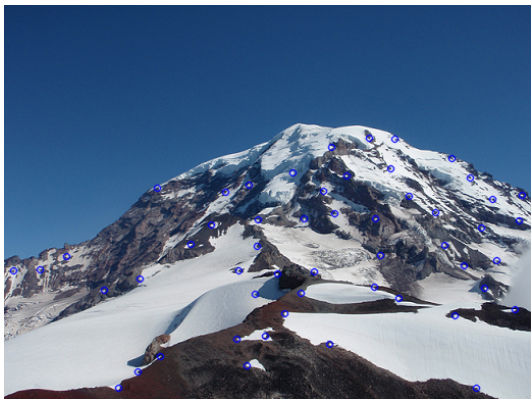
("ALL" indicates Harris Corner + Matching + RANSAC + Stitching + Saving images)

3. Saving Files Naming Convention

1. All files' names starting with "1" is the results of Harris Corner Detection.

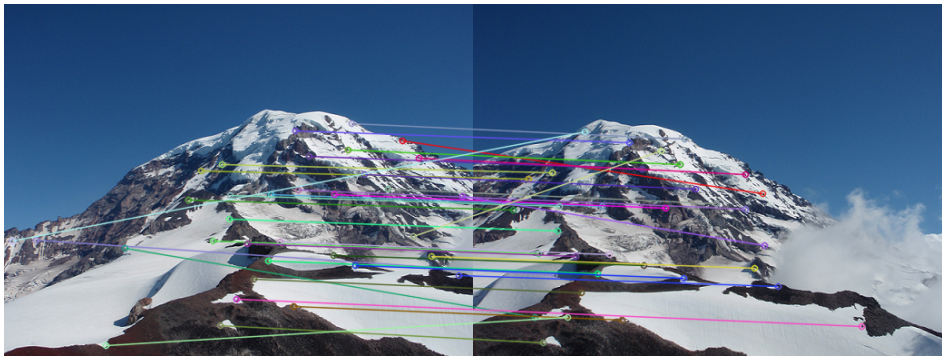
For example: 1b.png,

1c.png



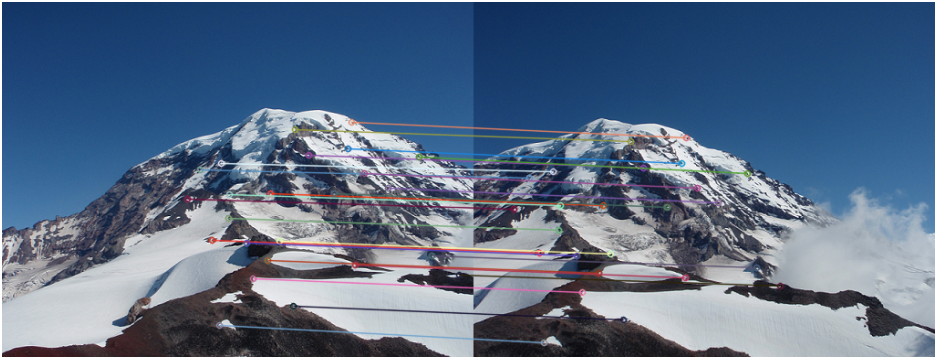
2. All files' names starting with "2" is the results of Feature Matching before RANSAC.

For example: 2b.png



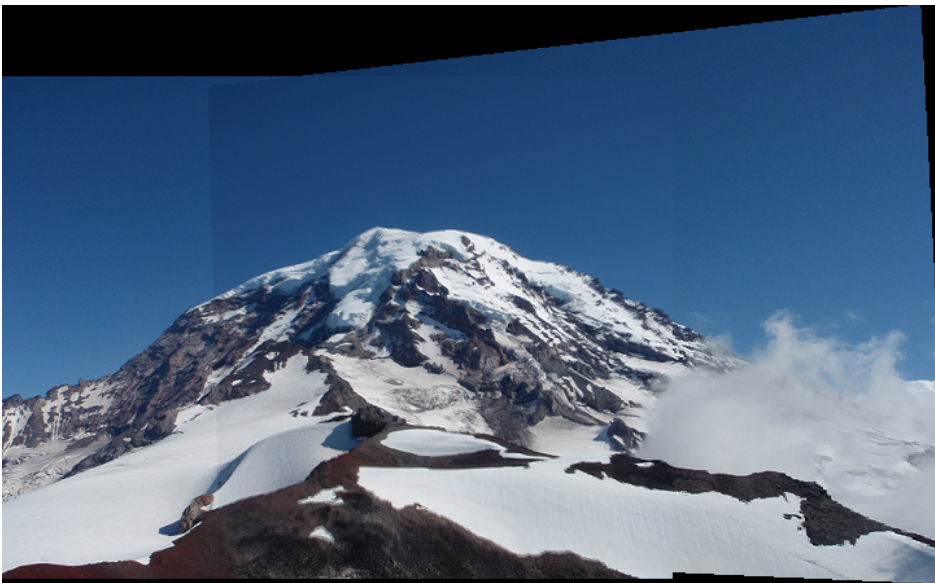
3. All files' names starting with "3" is the results of Feature Matching after RANSAC.

For example: 3b.png



4. All files' names starting with "4" is the results of stitched image.

For example: 4b.png



5. The program will show every single result image along the process, you can enter "0" on keyboard to continue the program after each result image is shown.

4. Function Explanation

1. File: [main.py](#): main file

Function:

- 1) ***main()***: the entry function
2. File: [my_images.py](#): define a class which stores all reading images and image paths
3. File: [helper_functions.py](#): defines helper functions

Functions:

- 1) ***read_threshold_harris()***: read an integer as harris corner threshold
- 2) ***read_ssd_threshold()***: read an integer as SSD distance threshold
- 3) ***read_ratio_test_threshold()***: read an integer as ratio test threshold
- 4) ***read_inlier_threshold()***: read an integer as inlier threshold for RANSAC

- 5) ***read_no_of_iterations()***: read an integer as RANSAC running iterations
 - 6) ***print_params_table()***: print all parameters at the very beginning after all inputs
 - 7) ***save_image()***: save image under the specified folder (by default, "result_images/")
 - 8) ***delete_all_previous_result_images()***: clean the folder ("result_images/") at the beginning
4. [File: harris_corner_detector.py](#): define a class that does harris corner detection
- Functions:
- 1) ***is_local_maxima()***: determine whether it is local maxima within 3x3 window
 - 2) ***suppress_neighborhood()***: suppress the neighborhood in 3x3 window
 - 3) ***non_maximum_suppression()***: apply non-maximum suppression as in assignment 2
 - 4) ***dist()***: compute the distance between 2 pixels (used in the function (5))
 - 5) ***adaptive_suppression_within_r()***: apply adaptive non-maximum suppression as in assignment 2
 - 6) ***harris_corner_detection()***: apply harris corner detection
5. [File: image_descriptors.py](#): define a class that stores all sift_descriptors of an image
- Functions:
- 1) ***is_out_of_bound()***: determine whether the index is outside of the image
 - 2) ***calc_sift_descriptor()***: calculate the sift descriptor for an interest point
 - 3) ***calc_sift_descriptor_for_all_interest_points()***: calculate sift_descriptors for all interest points and store them in a dictionary
6. [File: sift_descriptor.py](#): define a template class for a sift descriptor
- Functions:
- 1) ***set_magnitude_and_theta()***: set magnitude and theta for each pixel in 16x16 window
 - 2) ***set_orientation_histogram_for_grid_cells()***: calculate the orientation histogram for 16 4x4 grid cells in 16x16 window
 - 3) ***normalize_descriptor()***: normalize the histogram (contrast invariant)
 - 4) ***get_descriptor()***: return the descriptor as a 1x128 vector
7. [File: homography_calculator.py](#): define a class that calculates the homography
- Functions:
- 1) ***project()***: project a point (x1, y1) from image 1 to image 2 as (x2, y2) and return it
(required in project guideline)
 - 2) ***computeInlierCount()***: compute the number of inlying points given a homography H
(required in project guideline)

- 3) ***find_all_inliers()***: find all the inliers with the best homography with the highest number of inliers
- 4) ***RANSAC()***: take a list of potentially matching points between two images and returns the homography transformation that relates them (required in project guideline)
8. **File: [descriptors_matcher.py](#)**: define a class that matches the descriptors between two images

Functions:

- 1) ***feature_distance()***: compute the distance between 2 sift_descriptors
 - 2) ***matching_descriptors()***: find the matching and store the cv.DMatch in a list
9. **File: [image_blender.py](#)**: define a class that blends 2 images
- Functions:
- 1) ***stitch()***: stitch two images (required in project guideline)
 - 2) ***findStitchedImageSize()***: determine the image size of stitched image
 - 3) ***copyImageOne()***: copy all pixels in image 1 to the stitched image
 - 4) ***copyImageTwo()***: copy all pixels in image 2 to the stitched image, and blend with image 1's pixels if they overlap
 - 5) ***isInImage1()***: check if a pixel is in image 1
 - 6) ***isInImage2()***: check if a pixel is in image 2
 - 7) ***isPurelyBlackInImage1()***: check if a pixel is a black pixel in image 1
10. **File: [filename_manager.py](#)**: manage the filenames of saved images

Functions:

- 1) ***get_2_harris_output_filenames()***: return 2 file names for harris corner detection outputs
- 2) ***get_matching_output_filename()***: return a file name for the output of feature matching (before RANSAC)
- 3) ***get_RANSAC_matching_output_filename()***: return a file name for the output of feature matching (after RANSAC)
- 4) ***get_image_stitch_output_filename()***: return a file name for the stitched image

5. References

1. Assignment 2 instruction and my submit code
2. cv.findHomography (doc):
https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#cv.FindHomography
3. numpy.linalg.pinv (doc):
<https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.pinv.html>