CSE 4310: Fundamentals of Computer Vision
University of Texas at Arlington
Spring 2022
Alex Dillhoff

Assignment 2

This assignment covers keypoint matching and image stitching with SIFT and RANSAC.

1 Keypoint Detection

For keypoint detection, you may use whichever feature extraction method you would like. One recommendation is to use SIFT from `scikit-image` (available in v0.19).

1.1 BONUS (25 points): Implement SIFT from scratch

As a bonus, implement SIFT following the details of the original paper and compare it on a set of images with the `scikit-image` implementation.

2 Keypoint Matching

Create a matching function that, given two sets of keypoint features, returns a list of indices of matching pairs. That is, pair (i, j) in the list indicates a match between the feature at index i in the source image with the feature at index j in the second image.

2.1 Plot Keypoint Matches

Create a plotting function that combines two input images of the same size side-by-side and plots the keypoints detected in each image. Additionally, plot lines between the keypoints showing which ones have been matched together.

3 Image Stitching

In this part, you will implement an image stitching solution which computes a transformation matrix used to warp one image so that it is stitched to another. The first part involves estimating a transformation matrix based on keypoints that are matched. These keypoints will probably contain outliers, so these estimations will then be validated as part of RANSAC.

3.1 Estimate Affine Matrix

Create a function `compute_affine_transform` which takes in a set of points from the source image and their matching points in the destination image. Using these samples, compute the affine transformation matrix using the normal equations. This function should return a 3×3 matrix.

3.2 Estimate Projective Matrix

Create a function `compute_projective_transform` which takes in a set of points from the source image and their matching points in the destination image. Using these samples, compute the projective transformation matrix using the normal equations. This function should return a 3×3 matrix.
When using this matrix later on, do not forget to apply a perspective divide!

3.3 RANSAC

Create a function `ransac` which takes in a set of keypoints in the source image and their potential matches in the destination image. Additionally, it should take in parameters to determine the number of iterations that RANSAC can run, the minimum number of samples to fit a model with, and a threshold boundary. Implement the RANSAC function following the pseudocode on the Wikipedia page (or possibly other sources).

3.4 Testing

Test your implementation on the sample images provided on Canvas to verify your result. Compare using RANSAC with an affine model versus projective model, try each approach using the same set of images. RANSAC will not compute the same results each time. Depending on the selection of keypoints, this may not find a best fit. Create a brief report for this section that shows some of the samples you tried and what the outcome was.
Save your code as `stitch_images.py`.