University of Michigan
EECS 504: Foundations of Computer Vision
Winter 2020. Instructor: Andrew Owens.

Problem Set 9: Panoramic Stitching

Posted: Tuesday, March 24, 2020 **Due:** Tuesday, March 31, 2020

For Problem 9.1, please submit your solution to Canvas as a notebook file (`.ipynb`), containing the visualizations that we requested. Your `.ipynb` notebook should be named as `<unique_name>_<umid>.ipynb`. Example: `adam_01100001.ipynb`. Also, please remember to put your name and unique name in the first text block of the notebook. **Credit:** This problem was originally developed by the course staff for MIT 6.819/6.869.

The starter code can be found at:
`https://colab.research.google.com/drive/1OJchQ9xgO3M5jb14Fkhy_bWtZgGFwBPq`

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

Problem 9.1 *Panoramic Stitching*

In this problem we will develop an algorithm for stitching a panorama from overlapping photos (Figure 1), which amounts to estimating a transformation that aligns one image to another. To do this, we will compute ORB features[1] in both images and match them to obtain correspondences. We will then estimate a homography from these correspondences, and we'll use it to stitch the two images together in a common coordinate system.

In order to get an accurate transformation, we will need many accurate feature matches. Unfortunately, feature matching is a noisy process: even if two image patches (and their ORB descriptors) look alike, they may not be an actual match.

To make our algorithm robust to matching errors, we will use RANSAC, a method for estimating a parametric model from noisy observations. We will detect keypoints and represent descriptors using ORB. We will then match features, using heuristics to remove bad matches. We have provided you with two images (Figure 1) that you'll use to create a panorama.

Below is an outline of the functions you will be implementing:

(a) **[Optional]:** As an exercise, you can use RANSAC to fit a 2D line for a simple dataset. Sample n points from the line $y = 2x + 1$, then add 10 gross outliers (which you can sample from a uniform distribution). Use RANSAC and least squares to recover the line from this dataset.

[1]ORB is very similar to SIFT. Until recently, SIFT was patented, and was difficult to use in OpenCV.

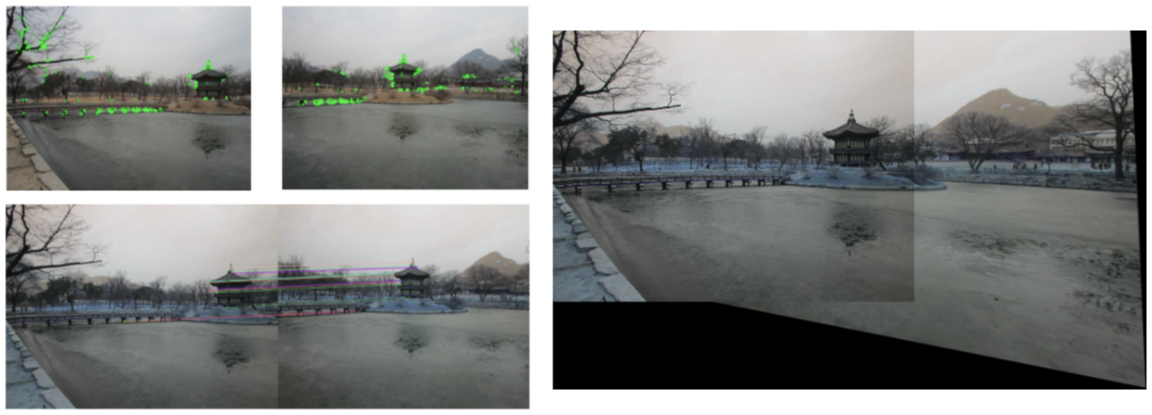

Figure 1: Panorama produced using our implementation. The image pair shown on the left represents the keypoints in the two source images and below them are the predicted feature correspondences. On the right is the stitched panorama.

(b) Implement `get_orb_features(img)` [1 point] to compute orb features for both of the given image. Implement `match_keypoints(desc1, desc2)` [1 point] to compute keypoint correspondences between the two source images using the ratio test. Run the plotting code to visualize the detected features and resulting correspondences.

(c) Write a function `find_homography(pts1, pts2)` [1 point] that takes in two $N \times 2$ matrices with the x and y coordinates of matching 2D points in the two images and computes the 3×3 homography H that maps `pts1` to `pts2`. You can implement this function using either nonlinear least squares or direct linear transform (DLT). As discussed in class, you may use `scipy` library's built-in nonlinear least squares function.

(d) Your homography-fitting function from (c) will only work well if there are no mismatched features. To make it more robust, implement a function `transform_ransac(pts1, pts2)` [1 point] that fits a homography using RANSAC. You can call `find_homography(pts1, pts2)` inside the inner loop of RANSAC. You will also be responsible for figuring out the set of parameters to use to produce the best results.

(e) Write a function `panoramic_stitching(img1, img2)` [1 point] that produces a panorama from a pair of overlapping images using your functions from the previous parts. Run the algorithm on the two images provided. You result should be similar to that of Figure 1.

(f) **[Optional]:** Use your Laplacian blending code from PS2 to remove the seam between the two images.

(g) **[Optional]:** Extend the algorithm to handle n (> 2) images, and run it on your own photos, or photos you found online.