University of Michigan
EECS 442: Foundations of Computer Vision
Fall 2023. Instructor: Andrew Owens.

Problem Set 8: Panoramic Stitching

Posted: Wednesday, Nov. 8, 2023 **Due: Monday**, Nov. 20, 2023

Please convert your Colab notebook to a PDF file and submit the PDF file to Gradescope, *making sure to label your answers*. We have included the PDF conversion script at the end of the notebook. Nothing needs to be submitted to Canvas. **Credit:** This problem was originally developed by the course staff for MIT 6.819/6.869, and 8.1 uses materials from David Fouhey's EECS 442 class.

The starter code can be found at:
https://colab.research.google.com/drive/1KO0SuICM1j-klFMQsJ9kpQw4pqkafmf2#scrollTo=UGDtnHGBDORy

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

Problem 8.1 *Homography estimation*
First, we will implement a function to compute a homography matrix H from a set of point correspondences. You can implement this function using nonlinear least squares (or, alternatively, the direct linear transform), as described in lecture.

Hint: For nonlinear least squares, we recommend using `scipy` library's built-in nonlinear least squares. To use that function, you need to:

- Define a cost function, $f(\texttt{h}; \texttt{pts1}, \texttt{pts2})$, that calculates the projection error (a vector of length $2N$) between `pts1` and projected `pts2` using homography H. The vector $\texttt{r} \in \mathbb{R}^2$ of residuals for point i is:

$$\texttt{r} = \texttt{pts1}[\texttt{i}] - \texttt{cart}(H * \texttt{homog}(\texttt{pts2}[\texttt{i}])). \tag{1}$$

  Here, `homog(x)` converts `x` into homogeneous coordinates, `cart(x)` converts `x` to Cartesian coordinates, and h is a flattened version of the homography H, which is what we will estimate.[1]

- Provide an initial guess for the homography `h`. A length 9 vector filled with '1's should be good enough.

(a) **(2 points)** Implement a function `fit_homography(pts1, pts2)` that computes the homography matrix from a set of point correspondences.

[1] Alternatively, you can try minimizing this loss using PyTorch and gradient descent. We didn't do this in our implementation, but it should have a very similar result — nonlinear least squares can sometimes provide a big improvement, but probably not for a simple problem like this one!

(b) **(2 points)** Apply the computed homography H to the original points. Plot these original points, the desired destination points, and the transformed points using matplotlib, as a scatter plot. Make sure the destination and transformed points are very close together. Include the visualization in the Colab notebook.

Problem 8.2 *Panorama 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[2] 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.

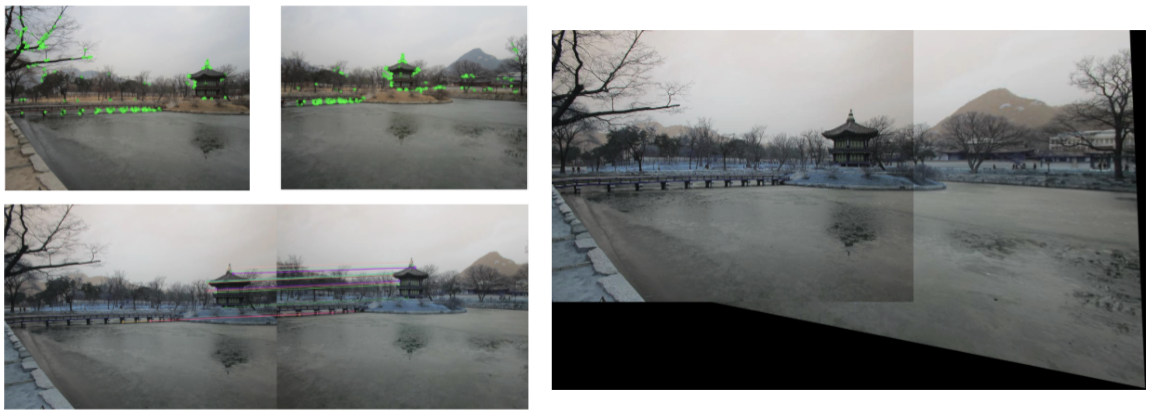

Figure 1: Panorama produced using our implementation. The image pair shown on the left represents the keypoints in the two source images and below them are the predicted feature correspondences. On the right is the stitched panorama.

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.

(a) You will start by computing features and image correspondences.

- **(2 points)** Implement `get_orb_features(img)` to compute orb features for both of the given image.
- **(2 points)** Implement `match_keypoints(desc1, desc2)` 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.

[2]ORB is a hand-crafted feature similar to SIFT. Until recently, SIFT was unfortunately patented, and was therefore difficult to use in OpenCV.

(b) **(2 points)** Your homography-fitting function from problem 8.1 will only work well if there are no mismatched features. To make it more robust, implement a function `transform_ransac(pts1, pts2)` that fits a homography using RANSAC. You can call `fit_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.

(c) **(2 points)** Write a function `panoramic_stitching(img1, img2)` 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.