TTIC 31040: Introduction to Computer Vision
Spring 2021

Problem Set #4

Out: May 3, 2021

Due: May 12, 11:59pm
Instructions: Turn in a write-up of your solution (as markdown cells) along with the code and the results as an iPython (Jupyter) notebook, including any figures or tables you generate. Please name your notebook the following:
`firstname-lastname-ps4.ipynb` and upload it on Canvas. Make sure the notebook you are submitting has been fully run, so that all the results are displayed in it.
Collaboration policy: it is OK (and encouraged) to discuss the problem set and any ideas for solving it with other students. However, in the end you must write your solution on your own. This includes writing any code and running any experiments.

1 Panorama Stitching

For this problem, you will use what we have learned previously about homography estimation as well as more recent material (RANSAC and image descriptors) to stitch images into a panorama. You can use either the OpenCV library or the `scikit-image` and any function or descriptor you like for this problem. We have included some starter code and some tips on `scikit-image` and OpenCV functions, you are welcome to use any of the recommendations in your implementation but you don't have to follow the structure of the starter.

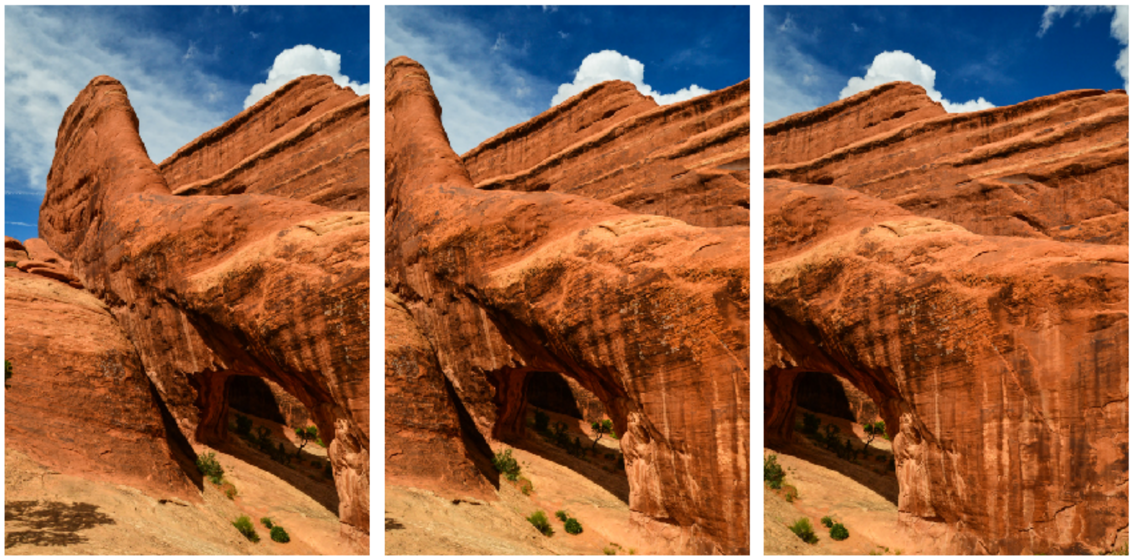

Figure 1: A set of images with sufficient overlap and low scene motion and parallax will allow for seamless panorama stitching.

Problem 1 [5 points]
Load the images, display them, and convert to grayscale. You can use either one of the two image sets provided.

End of problem 1

Problem 2 [20 points]
Use either OpenCV or `scikit-image` to detect keypoints (and extract features) from all three images. We recommend either SIFT (Scale-Invariant Feature Transform) or ORB(Oriented FAST and Rotated BRIEF). Visualize the keypoints and the correspondences estimated. What kinds of false positives do you see? What kinds of image features produce "good" correspondences?

End of problem 2

Problem 3 [20 points]
Homography and RANSAC. We have three images we wish to stitch together – our goal is to keep the central image as an anchor and warp the "left" and the "right" image so that they align with the central image. This is achieved through homography estimation, with the added requirement of a RANSAC routine to filter out the incorrect correspondences. Use either OpenCV (`cv2.findHomography` with `cv2.RANSAC`) or scikit-image (`skimage.measure.ransac` with `ProjectiveTransform`) to robustly estimate a homography warping `image_0` to `image_1` and `image_2` to `image_1`).

The above functions return the set of inliers after homography estimation, plot them and compare to the original correspondences. Have all of the outliers been removed? Play around with the parameters of RANSAC (such as the number of iterations and residual threshold) to see if you can get better results.

End of problem 3

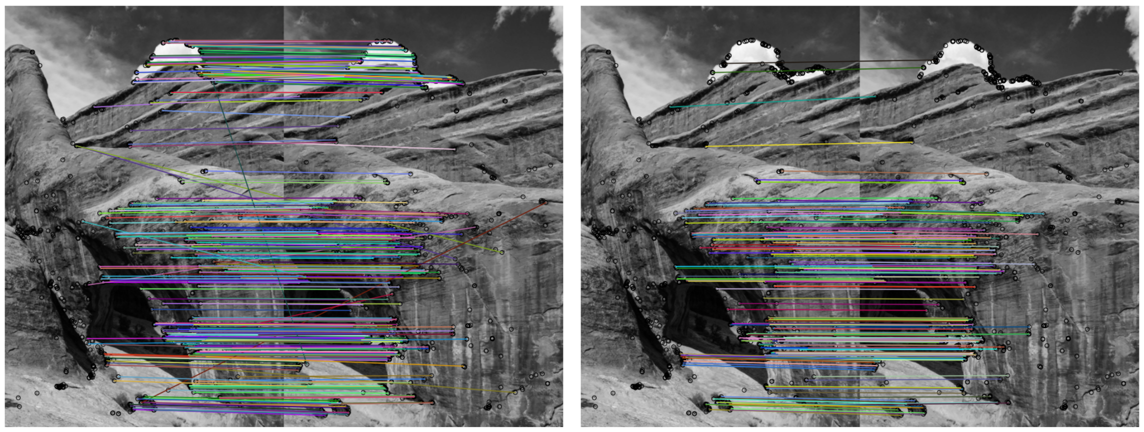

Figure 2: The output of a brute force matcher on ORB features (left); output of the inliers for homography estimation with RANSAC (right).

Problem 4 [25 points]
Image warping and blending. We need to use the estimated transformations to "place" the three images in their right place. First we will warp the corners with the estimated homographies (starter code provided) to find the extent of the full panoramic image, and to give us an idea of where to place the middle image (hint: the offset is part of what we used to estimate the panoramic image shape). After applying the proper transformations to all three images, they should look like Figure 3. Keep track of a binary mask for each image indicating whether there is a valid pixel at a given location, and use the shifted graylevel images and the masks to do a simple summation followed by normalization to produce something like the panorama in Figure 4.

You will need to produce both the grayscale panorama and a color panorama (*Hint: you should be able to do most of the above operations on grayscale images, then simply use the output on the color images to produce a color panorama.*)

End of problem 4

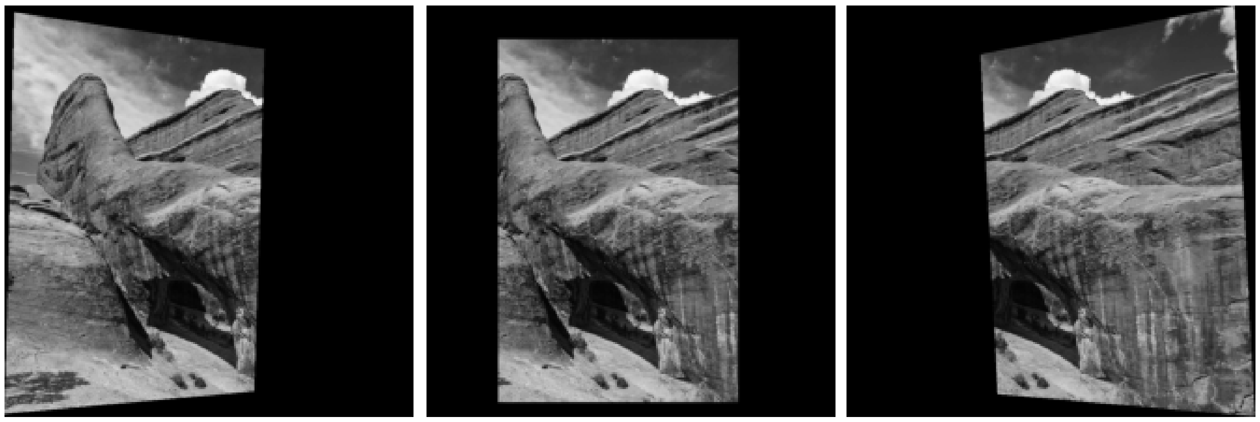

Figure 3: The three images, offset, warped and ready for stitching.

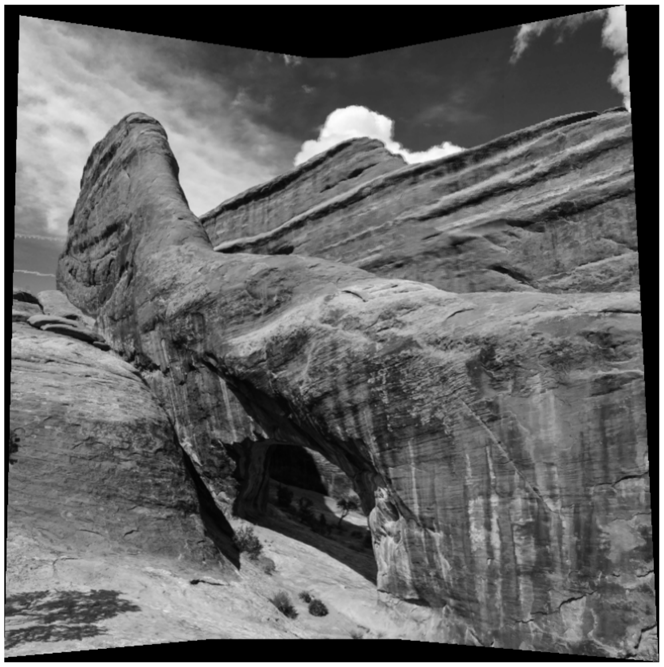

Figure 4: The result of graylevel panorama stitching.

2 Hough transform

Here we will consider the parametric Hough Transform (HT). We discussed HT in the lecture in some details for lines; now let us develop the HT formulation for detecting *ellipses*.
Consider the task of finding ellipses in an image of size $N \times N$ pixels.

Problem 5 [10 points]
Suppose that we want to detect ellipses in the image without any restriction on their position, orientation and shape. What is the HT voting space? For every dimension, describe its interpretation and the range of the values (do not worry about the quantization).

End of problem 5

Problem 6 [10 points]
Ignoring quantization (i.e., treating the voting space as a continuous multi-dimensional real space) write the formula for a Hough Transform vote generated by an edge pixel (x, y) (which may be on an ellipse).

End of problem 6

Problem 7 [10 points]
Now suppose we know the *aspect ratio* of the ellipses we are trying to detect, i.e., the ratio of the major to minor axes of the ellipse. The location, orientation and scale are still unknown. How would the voting space and the voting calculations change?

End of problem 7