CSE47201: Introduction to Computer Vision

Assignment #2

Due on Wednesday, November 10th, 2021, before the class

Objective

The goal of this assignment is to understand feature detection and image stitching. You will also learn how to compute and apply image transformations. If you have any issues, feel free to contact the instructor for advice.

Instructions (READ CAREFULLY)

- Complete individual steps and turn in your outputs (see **Task #**) to Blackboard. There are a total of **5 Tasks** in this assignment.
- A single PDF file with all solutions and a discussion of your solution.
- Along with the PDF file you should provide a link or a zip for your results and code (e.g., GitHub).
- No extension. Please start your assignment as soon as possible. Do NOT wait until the deadline.

1. Feature detection

Our goal is to implement the Harris corner detector for a gray image. We will break down the steps we studied into several tasks below.

Task 1. **Filtering, gradient, structure tensor, and Harris operator:** You first need to apply a Gaussian filter on your input images (see "fspecial('gaussian')" for Gaussian – use a sigma of 0.5 with a radius of 5. You should compute the image gradients and then use the gradient images as input for a function that computes the structure tensor (see H matrix we studied in the lecture) for each pixel within a 11-by-11 patch (i.e., pixel location±(5,5)). Next determine if each point is a corner based on the Harris operator. Your feature selection will then pick the 1,000 strongest points as the features of the image. Execute this detector on im1.

Hint: As you already did in the first assignment, use "filter2" for convolution. Also, you may want to use "ind2sub" to convert 1D indices to 2D.

Task 2. Maxima suppression: As you can observe in the step above, the selected features are typically clustered (show these clusters as a figure in your result). To avoid this, implement a non-maxima suppression within a radius of 10 for the final selection process (i.e., find local maximum for each pixel within that radius). Execute the corner detector again on im1.

Hint: See "fspecial('disk')" to use a predefined binary circle window.

Task 3. Similarity measure: Make a rotation-invariant descriptor within a 21-by-21 patch (assume that scale is constant across the input images). Use the structure tensor (H) to compute eigenvectors to determine the orientation of your patch and use intensities for your descriptor. Then, compute the distance (both SSD and NCC) of every feature in im1 to every feature in im2 and plot the corresponding distance matrix as image. The distance matrix contains at position (i,j) the distance between i-th feature in im1 and j-th feature in im2. Explain your observation. Once your similarity matrix is ready, you can compute putative matches based on their scores. Illustrate putative matches.

Hint: For this, you will need to learn about “maketform” and “imwarp” functions that can do transformation and image interpolation (the official document says “maketform” is not recommended but you can still use it or check out its compatible form there). Also, use “eigs” for eigendecomposition.

2. Image stitching

Our goal is to align images into a panorama. We will implement a RANSAC-based method to estimate an affine transformation as well as homography between two images. Use the feature detector and descriptor you implemented above – you need to choose at least 12,000 points here. Please adjust the parameter for better results and describe your configurations in your document.

Task 4. Implement a RANSAC to estimate transformations; give a 2-pixel threshold for the inlier test. Estimate the affine transformation and the homography between image im1 and im2. Warp one image onto the other using the estimated transformation. Based on putative matches you created, roughly compute a ratio of how many points are well matched after your visual inspection (it is not necessarily accurate, once again, “roughly”). Then, we use that ratio to represent the inlier group. Suppose that you want to find a correct model in a chance of 0.99 given the inlier ratio. Calculate how many iterations you will need for both affine transformation and homography. Also, use this for your implementation, too.

Hint: You may need “randperm” to shuffle putative matches and “\” operator to solve a linear system. Once again, use “maketfor” and “imwarp” to transform your image.

Task 5. Execute stitching using both affine transformation and the homography for each of the two pairs in the panorama directory (you can also use any two image pairs you want for this assignment, e.g., campus scene). Create a new image big enough to hold the panorama and composite the two images into it. You can composite by simply averaging the pixel values where the two images overlap.