COS 429 Computer Vision Fall 2020, Homework 1

Due: September 18th, 11:59 PM (Princeton Time)

1 Part 1: Blob Detection

In this section, you will implement a blob detector using the Laplacian-of-Gaussian filter. This will allow you to automatically identify salient features in your image, which will help you find correspondences between image pairs in Part 2 of this assignment. You will do this by completing the function `DetectBlobs(...)` whose template is provided in `detectBlobs.py`. We have provided `evalBlobs.py` and the associated `drawBlobs.py` for viewing the results of your code in order to help you debug.

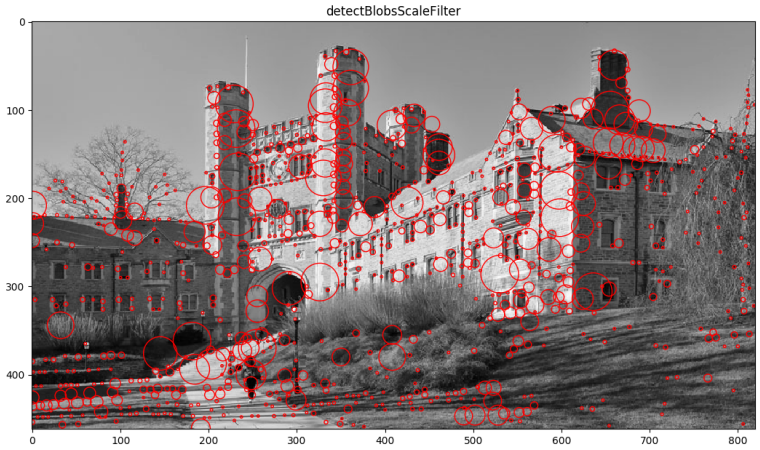

Figure 1: An example of potential blob detection output

2 Part 2: Image Stitching

In this section, you will implement a system for stitching images together into a panorama based on your blob detector from Part 1. This will consist of several steps:

1. Detecting feature keypoints in the input images using your blob detector from Part 1. If you were not able to complete Part 1 successfully or you want to work on Part 2 first, you can use the compiled solution to Part 1 `detectBlobsSolution.pyc`.

2. Compute feature descriptors for each of the keypoints. You can simply use the local neighborhood of pixels around the detected keypoints or use the SIFT feature descriptor (already provided in the starter code).

3. Find correspondences of detected keypoints in the left and right images by computing the euclidean distances between their respective feature descriptors. Based on these distances, you will have to decide which to keep and which to discard.

4. With your chosen point correspondences, you will have to fit a homography between the image pairs using the RANSAC algorithm.

5. With your homography matrix H, you need to figure out how to properly warp one image onto the other to construct a panorama.

You will complete these steps by filling in the function templates provided in `utilsImageStitching.py`. Lastly, you will complete `evalStitchingMulti.py`, which will be a more complete version of a system for "Recognizing panoramas" – i.e., a system that can take as input a "pile" of input images (including possible outliers), figure out the subsets that should be stitched together, and then stitch them together. In order to help you debug, we have provided `evalStitchingPair.py` which demonstrates the expected usage of your functions from `utilsImageStitching.py` and draws your detected blobs and keypoint matches.

Figure 2: An example of keypoint matches. Input to RANSAC

Figure 3: An example of a stitched panorama

2.1 Report

After completing your system for recognizing panoramas, assemble a report of results. In your report, include:

- Constructed panoramas from each of the image sets located in `image_sets/`
- Constructed panoramas from two image sets of your choosing (place them in `image_sets/success/` and `image_sets/failure/`, respectively). You should include one example in which your system succeeds in stitching everything together correctly. The other should be an example of a failure of your system on a "difficult" image sequence. For instance, you can try to find an image set with a lot of repetition, or images separated by an extreme transformation (large rotation, scaling, etc.). For the failure case, describe in a few sentences what you think is the cause of the failure.

3 Allowed Software Libraries

In any of the code you write for this assignment, you are not allowed to use any pre-existing functions except the following:

- Any function already defined or used in the starter code
- `convolve` (Documentation) and `maximum_filter` (Documentation) from `scipy.ndimage`
- `resize` (Documentation) and `warpPerspective` (Documentation) from `cv2`. OpenCV version `3.4.2` should be used, you can install using
  - `conda install python=3.7` followed by
  - `conda install -c anaconda opencv==3.4.2`
- Anything from `numpy`
- Anything from the Python Standard Library (Documentation)

4 Logistics

4.1 What you need to submit

Compressed into a `<NetID>_HW1.zip`:

1. Completed `detectBlobs.py`
2. Completed `utilsImageStitching.py`
3. Completed `evalStitchingMulti.py`
4. A `report.pdf` consisting of results on each set of test images
5. A directory `image_sets/` with your own two image sets

4.2 Grading

Your final grade for HW1 will consist of:

- 60 points: Your panorama system's results on the three provided test sets under `image_sets/` (20 points each).
- 20 points: Your panorama system's results on your own two test sets under `image_sets/` (10 points each).
- 20 points: Your panorama system's results on four hidden test sets (5 points each).
- **Important!** If the panorama system in your solution uses the provided blob detector `detectBlobsSolution.pyc` instead of your own implementation from Part 1, you will incur a penalty of 50%. Your grade will be 50% of what you would have gotten using your own detector.

How we will grade: We will unzip your submitted `<NetID>_HW1.zip` and will copy your `detectBlobs.py`, `utilsImageStitching.py` and `evalStitchingMulti.py` into an empty folder. We will also copy over your `image_sets/success/` and `image_sets/failure/` directories into an empty `image_sets/` directory and will insert the original image sets plus the four hidden image sets. We will run your submission by running `python evalStitchingMulti.py <path_to_folder_of_images> <file_path_to_output.png>` from the command line. If this fails, we will include `detectBlobsSolution.pyc` in the directory and rerun. Additionally, we will inspect the code for any violation of the rules. Your code must be able to run within 5 minutes on an AWS EC2-T2 instance.