Chennai Mathematical Institute
Computer Vision
February 29, 2020

Due Date: March 13, 2020 at 6:00PM

Assignment 3: Feature Detection and Description

The goal of this assignment is to create a local feature detection and description algorithm using techniques we have covered in class. The approach we suggest is a simplified version of the same operations in SIFT. The features we arrive at are intended to work for *instance-level* matching, i.e., multiple views of the same physical scene. However, we will not be including the matching as part of this assignment.

Details of the Assignment
For this assignment, you will implement the two distinct steps of feature detection and description that are part of SIFT:

1. **Interest point detection** in **`harris.py`** (see Szeliski 4.1.1 for background and approaches)
2. **Local feature description** in **`sift.py`** (see Szeliski 4.1.2 for background and approaches)

For this project, we suggest specific, relatively simple algorithms for each stage. You are encouraged to experiment with more sophisticated algorithms!

Interest point detection (harris.py)
You will implement the Harris corner detector as described in the lecture materials and Szeliski 4.1.1. See Algorithm 4.1 in the textbook for pseudocode. The starter code gives some additional suggestions. You do not need to worry about scale invariance or keypoint orientation estimation for your baseline Harris corner detector. The original paper by Chris Harris and Mike Stephens describing their corner detector is included in the reference material.

You will also implement **adaptive non-maximal suppression**. While most feature detectors simply look for local maxima in the interest function, this can lead to an uneven distribution of feature points across the image, e.g., points will be denser in regions of higher contrast. To mitigate this problem, Brown, Szeliski, and Winder (2005) only detect features that are both local maxima and whose response value is significantly (10%) greater than that of all of its neighbors within a radius r. The goal is to retain only those points that are a maximum in a neighborhood of radius r pixels. One way to do so is to sort all points by the response strength, from large to small response. The first entry in the list is the global maximum, which is not suppressed at any radius. Then, we can iterate through the list and compute the distance to each interest point ahead of it in the list (these are pixels with even greater response strength). The minimum of distances to a keypoint's stronger neighbors is the radius within which the current point is a local maximum. We call this the suppression radius of this interest point, and we save these suppression radii. Finally, we sort the suppression radii from large to small, and return the n keypoints associated with the top n suppression radii, in this sorted order. Feel free to experiment with n, we used **`n=1500`**.

You can read more about ANMS in the textbook and in the reference papers included.

Local feature description (sift.py)
You will implement a SIFT-like local feature as described in the lecture materials and Szeliski 4.1.2. See the placeholder **`get_features()`** for more details.

Using the starter code (assignment3.ipynb)
The top-level **`assignment3.ipynb`** IPython notebook provided in the starter code includes file handling and visualization functions as well as calls to placeholder versions of the two functions listed above. Running the starter code without modification will visualize random interest points.

Potentially useful NumPy (Python library), OpenCV, and SciPy functions:
`np.arctan2()`, `np.sort()`, `np.reshape()`, `np.newaxis`, `np.argsort()`, `np.gradient()`, `np.histogram()`, `np.hypot()`, `np.fliplr()`, `np.flipud()`, `cv2.Sobel()`, `cv2.filter2D()`, `cv2.getGaussianKernel()`, `scipy.signal.convolve()`.

Forbidden functions (you can use for testing, but not in your final code): `cv2.SIFT()`, `cv2.SURF()`,`cv2.HOGDescriptor()`, `cv2.cornerHarris()`, `cv2.FastFeatureDetector()`, `cv2.ORB()`, `skimage.feature`, `skimage.feature.hog()`, `skimage.feature.daisy`, `skimage.feature.corner_harris()`, `skimage.feature.corner_shi_tomasi()`,`skimage.feature.ORB()`.

We haven't enumerated all possible forbidden functions here but using anyone else's code that performs interest point detection or feature computation for you is forbidden.

Tips, Tricks, and Common Problems

- Make sure you're not swapping x and y coordinates at some point. If your interest points aren't showing up where you expect or if you're getting out of bound errors you might be swapping x and y coordinates. Remember, images expressed as NumPy arrays are accessed image[y,x].
- Make sure your features aren't somehow degenerate. You can visualize your features with plt.imshow(image_features), although you may need to normalize them first. If the features are mostly zero or mostly identical you may have made a mistake.

Writeup
For this assignment, you need to write a report describing the algorithms you have used for interest point detection and feature description. Include discussion of any key design decisions and experiments that led to the decisions. Include your results, discussion of the results.

Extra Credit

Implementation of bells and whistles can increase your grade by up to 10 points (potentially over 100). The max score for all students is 110.

For all extra credit, be sure to include quantitative analysis showing the impact of the particular method you've implemented. Each item is "up to" some amount of points because trivial implementations may not be worthy of full extra credit.

Interest point detection bells and whistles:

- up to 5 pts: Try detecting keypoints at multiple scales or using a scale selection method to pick the best scale.
- up to 5 pts: Try estimating the orientation of keypoints to make your local features rotation invariant.

Local feature description bells and whistles:

- up to 3 pts: The simplest thing to do is to experiment with the numerous SIFT parameters: how big should each feature be? How many local cells should it have? How many orientations should each histogram have? Different normalization schemes can have a significant effect, as well. Don't get lost in parameter tuning, though.
- up to 5 pts: If your keypoint detector can estimate orientation, your local feature descriptor should be built accordingly so that your pipeline is rotation invariant.
- up to 5 pts: Likewise, if you are detecting keypoints at multiple scales, you should build the features at the corresponding scales.

Rubric

- +35 pts: Implementation of Harris corner detector in harris.py
- +15 pts: Implementation of adaptive non-maximal suppression in harris.py
- +35 pts: Implementation of SIFT-like local feature in sift.py
- +15 pts: Writeup on design choices, implementation and results
- +10 pts: Extra credit (implementation of any bells & whistles up to ten points, max score of 110).

What to hand in:
The folder you hand in must contain the following:

- code/ - directory containing all your code for this assignment;
    - harris.py and sift.py should include your implementation of the `get_interest_points()` and `get_features()` functions.
    - Any additional functions implemented should be well-documented and included.
    - Your assignment3.ipynb notebook showing your results.
- A report in PDF format detailing your implementation and discussion of results.
- README - text file containing how to run your bells & whistles and extra credit implementations
  Hand in your project as a zip file on Moodle.

Late Policy:

The assignment will not be accepted past 72 hours beyond the specified due date. You will lose 1 point for each hour you are late beyond the deadline for the first 6 hours. Beyond that, the delay will be counted in day units with a 10point penalty for each day you are late.

Credits:

This assignment draws from a larger feature matching programming project developed by James Hays, Samarth Brahmbhatt, and John Lambert used at Georgia Institute of Technology and Brown University.

References:

Harris, C. and Stephens, M. J. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–152.

Brown, M., Szeliski, R., and Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)* , pp. 510–517, San Diego, CA.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.