Lab 7: SIFT Feature Matching

Due Oct 21, 2020 by 11:59pm **Points** 10 **Submitting** a text entry box or a file upload

The task for today is to detect SIFT features in one image, match them to another image, and analyze how many matches are correct.

Data

We will use these images (**graffiti.zip** (https://elearning.mines.edu/courses/25410/files/2294791/download?download_frd=1)), which come from a larger dataset (http://kahlan.eps.surrey.ac.uk/featurespace/web/data.htm). The images are of a planar surface, taken from various viewpoints. Recall that a homography maps points on one plane to points on another plane (see the lecture slides on “Homography” from Week 6). The authors of the dataset provide the “ground truth” homographies between images, which you can use to determine if a point is being matched correctly. We will consider a point to matched correctly if the error distance between its ground truth location and the matched location is less than or equal to s, where s is the size of the SIFT feature.

Code

The Python program you need is given here (**main_match.py** (https://elearning.mines.edu/courses/25410/files/2294863/download?download_frd=1)). At the beginning of the code are algorithm parameters that you will have to adjust. The parameter “ratio_match_dist” defines the ratio of best-match-distance to second-best-match-distance. This is used to reject ambiguous matches. The smaller this ratio, the more potentially false matches are rejected, but you also lose more valid matches.

Metrics

We will count:

- TP (number of true positives), which is the number of correct matches
- FP (number of false positives), which is the number of incorrect matches
- P (number of actual positives), which is the number of features that should have been matched correctly (ie., the predicted location based on the homography transformation is somewhere within the query image)

The metrics we want to find are:

- Precision = TP / (TP + FP). This tells us, of all our matches that we found, what fraction of our matches is correct. In other words, how many of our matches are “inliers”.
- Recall = TP / P. This tells us, of all the features that should have been matched, what fraction were actually matched.

We want both of these to be as high as possible, but there is a tradeoff between them. For example, raising the criteria for a valid match improves precision, but decreases recall.

To do

Download the code and the images. As described in the lecture on Wednesday, you will have to change the versions of OpenCV and Python, in order to use the SIFT code.

Run the code as is, and observe the matching features. The program prints the result of matching to the console. It also displays a green “G” at the predicted location of the ground truth point, and a yellow “D” at the location of the detected

point. The size of a circles corresponds to the size of the SIFT feature.

Try increasing the “ratio_match_dist” parameter. You should get more true positives, but also more false positives. The program prints the number of true and false positives at the end.

Now, add code to the end of the main program to calculate recall and precision. Run the program on query images 2 and 3. Calculate recall and precision for the following values of the parameter “ratio_match_dist”: 0.5, 0.7, 0.8, 0.9, 1.0. Three significant digits are sufficient when reporting the values; it is not necessary to display the values with more digits than that.

Query image 2

- Ratio = 0.5: Recall = ?, precision = ?
- Ratio = 0.7: Recall = ?, precision = ?
- Ratio = 0.8: Recall = ?, precision = ?
- Ratio = 0.9: Recall = ?, precision = ?
- Ratio = 1.0: Recall = ?, precision = ?

Query image 3

- Ratio = 0.5: Recall = ?, precision = ?
- Ratio = 0.7: Recall = ?, precision = ?
- Ratio = 0.8: Recall = ?, precision = ?
- Ratio = 0.9: Recall = ?, precision = ?
- Ratio = 1.0: Recall = ?, precision = ?

Turn in:

- The values of the table.
- Your Python program.