RANSAC

This homework is intended provide students with the opportunities to implement image matching using local invariant features

The zip file containing sample images and code is **hw4.zip**. This assignment consists of several Python script files.

main_match.py is the main script for the matching part of the assignment;

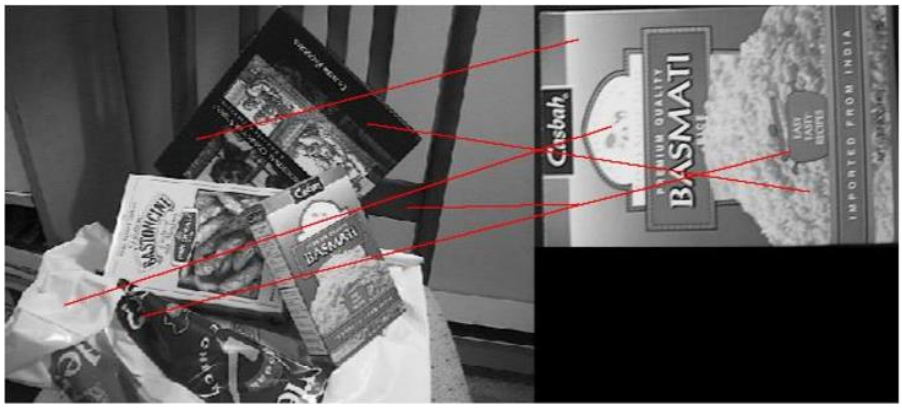
To make the assignment more tangible, we've implemented some boilerplate code in hw_utils.py, which you SHOULD NOT modify. Your function implementations for the entire assignment should be stored in solution.py. You are free to use main_match.py to get started. If you are using Juypter notebook, please modify the provided main files, which should be straightforward.

> Hand in all parts of this assignment in PLATO (both the code and report PDF file as specified). To get full marks, your functions (i.e., *.py files) must not only work correctly, but also must be clearly documented with sufficient comments for others to easily use and understand the code. You will lose marks for insufficient or unclear comments. In this assignment, you also need to hand in scripts showing tests of your functions on all the cases specified as well as the images and other answers requested. The scripts and results (as screenshots or otherwise) should be pasted into a single PDF file and clearly labeled. Note that lack of submission of either the code or the PDF will also result in loss of points.

The assignment

SIFT Keypoint Matching

> This assignment will make use of the SIFT features that are described in the paper **Distinctive Image Features from Scale-Invariant Keypoints**, by David Lowe. You don't need to read the whole paper, but you may wish to refer to parts of it to understand the approach.
>
> The unzipped directory contains a skeleton Python program main_match.py (which you can run by typing *python main_match.py* at the command prompt).



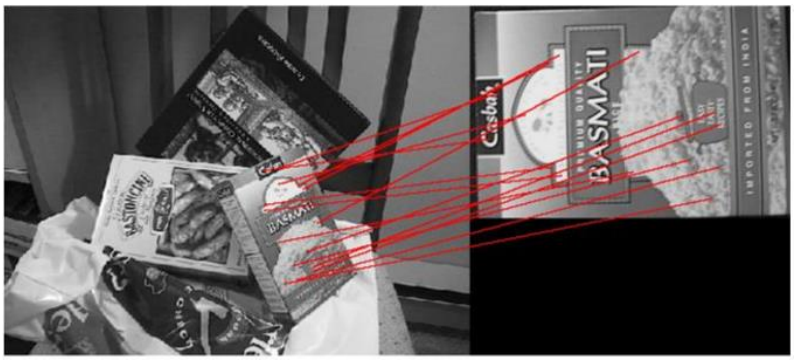
(Skeleton code의 Main_match.py 실행 화면)

1. (44 points) The sample program, main_match.py, loads two images and their invariant keypoints and then draws 5 lines between randomly selected keypoints to show how matches can be displayed. Your task is to improve this program so that it identifies and displays correct matches by comparing the keypoint descriptor vectors. Note: Your program should find **all possible matches**, not just 5 as shown in this sample.

   The function *FindBestMatches* in the file solution.py takes the invariant descriptor vectors stored in the numpy arrays *descriptors1* for the first image and *descriptors2* for the second, and returns a list of index pairs of the matches. Each row corresponds to a descriptor vector. To select the best match for a vector from the first image, you should measure its angle to each vector from the second matrix. As the descriptor vectors are already normalized to have unit length, the angle between them is the inverse cosine (*math.acos(x)* function in Python) of the dot product of the vectors. The vector with the smallest angle is the nearest neighbor (i.e., the best match).

   However, many keypoints from one image will have no good match to the second image. To eliminate false matches, the most effective method is to compare the smallest (best) match angle to the second-best angle. A match should be selected only if this ratio is below a threshold.

   Hints: The Python function *sorted* can be used to find the two smallest values in a list. The list method *index* can then be used to determine the (original) indices of these sorted elements.



(Sample capture of main_match.py : matches for "scene" and "basmati")

Select a threshold that gives mostly good matches for the images ``scene'' and ``book'', "scene" and "box". Try different thresholds so that only a small number of outliers are found (less than about 10). You can judge this by eye, as the outliers will usually produce matching lines at clearly different angles from the others. Print the box image showing the set of matches to the scene image for your suggested threshold value. Write a short description of the particular threshold value used, why you chose it and how important it was to get the value correct.

Include resulting "scene-book" and "scene-box" images showing your best set of matches in a PDF.

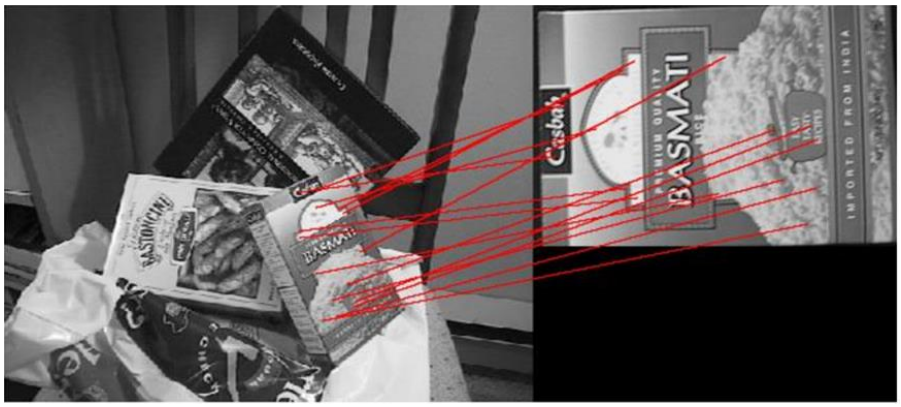
2. (56 points) Now given two images, you could obtain keypoint matches. Unfortunately, there are often lots of erroneous matches in this list. This part of the assignment is designed to take the matches from the previous part and to reduce the number of false matches using RANSAC.

   Implement the RANSAC routine to filter out false matches in the function *RANSACFilter*. In details, for each RANSAC iteration you will select just one match at random, and then check all the other matches for consistency with it. Repeat the random selection 10 times and then select the largest consistent subset that was found. Assume that we are sampling with replacement and that the final consensus set should include the initial match, i.e., the size of consistency set is at minimum 1.

   To check other matches for consistency with the first match, you need to use the *keypoints1* and *keypoints2* arrays that are provided for each image. Each row provides 4 numbers for a keypoint specifying its location, scale, and orientation in the original image (see the function ReadKeys in the file *hw_utils.py* for details).

   To check that one match is consistent with another, you should check that the change of orientation between the two keypoints of each match agrees within, say, 30 degrees. In other words, if the two keypoints for one match have a difference in orientation of 130 degrees, the

second match should have an orientation difference between 100 and 160 degrees. It is best to consider an example here. Suppose we want to compare 2 matches for consistency in orientation. The keypoint orientations from the first match are 45 degrees and -45 degrees. The orientation from the second match are 0 and 90 degrees. You should start by computing the difference in orientation for the two matches: d1 = -45 - 45 = 90 and d2 = 90 - 0 = 90. The difference between the two matches in this case is 180 degrees and would be outside the 30 degree tolerance, hence the second match is not part of the consensus set. Note that orientation is measured in radians, and that orientations are equal modulo 2 pi. Also, check that the change of scale agrees within plus or minus, say, 50%. Assuming two keypoints have scales s1 and s2, the change in scale is defined as s2/s1 (see slides). As an example, if the change of scale for our first match is 3, then to be within a consensus set, the second match must have change in scale within the range of 1.5 to 4.5 (assuming scale agreement of plus or minus 50%). For this assignment, we won't check consistency of location, as that is more difficult.



(Sample capture of main_match.py : matches for “scene” and “basmati” with RANSAC)

Try different values for the orientation and scale agreement (instead of using 30 degrees and 50% as mentioned above), and raise the matching threshold to get as many correct matches as possible while having only a few false matches. Try getting the best possible results on matching the difficult ``library'' and “library2” images.

Include resulting “library-library2” images showing your best set of matches in a PDF.

Deliverables

You will hand in your assignment in PLATO. You should hand in one zip file including two files, a file containing your code (i.e., *.py file) and a PDF report. In code, All your answer must be in solution.py. In addition, hand in a PDF document showing your solutions (python function), scripts (i.e., records of your interactions with the Python shell) showing the specified tests of your functions as well as the images and other answers requested. Also, this file must have sufficient comments for others to easily use and understand the code. The PDF file has to be organized and easily readable / accessible.

There must be 3 captures in your PDF. 2 for 1-1("scene-book" and "scene-box"), 1 for 1-2("library-library2").

Assignments are to be handed in before 11:59pm on their due date.