CSCI 677: Advanced Computer Vision - Fall 2021

Instructor: Prof. Nevatia

Assignment 3

Due on October 7, 2021 before 17:00 PDT

Instructions

This is a programming assignment. The goal is to use SIFT descriptors to locate desired objects in images that may contain other objects, occlusion and general clutters. It is assumed that the SIFT points on objects will arise primarily from a planar surface and that they can be matched in other views by homography transformations. All required functions are available in OpenCV version >= 4.4.0. The instructions below are based on OpenCV 4.5.2. An outline of the procedure can be found at:
https://docs.opencv.org/4.5.2/dc/dc3/tutorial_py_matcher.html

The basic workflow is to create and compute SIFT features, match key points through a matcher, refine matches and estimate the homopgraphy matrix accordingly. At the end, we need to assess if enough matches are found for considering an object as being "detected".

You are asked to use the "brute force" matcher (BFmatcher). We recommend using bf.knnMatch that returns top-k matches by filtering out weak matches according the ratio between the best and the second best matches. A snippet of code looks like:

```
    import cv2
    bf = cv2.BFMatcher()
    matchs = bf.knnMatch(src_dpts, dst_dpts, k=2) # If k is set to 2.
```

You may refer to https://docs.opencv.org/4.5.2/dc/dc3/tutorial_py_matcher.html for more details. We suggest to set k=2 but you are encouraged to try other k values for the best matching results.

Homography estimation code is also available in OpenCV; hints on how to use it can be found in https://docs.opencv.org/4.5.2/db/d27/tutorial_py_table_of_contents_feature2d.html. You are asked to use RANSAC to estimate matches. RANSAC is already implemented in OpenCV, you merely need to choose this option.

As this assignment is mostly implemented using built-in functions, you are asked to display some intermediate results to show the internal work flow of the program.

a. SIFT features: show the detected features overlaid on the images (*both the locations and directions, not 128-d vectors*). Also give out *the number of features detected* in each image.

b. Graphically show the top-20 scoring matches found by the matcher before the RANSAC operation. Provide statistics of how many matches are found for each image pair.

c. Show total number of inliner matches after homography estimations. Also show top-10 matches that have the minimum error between the projected source keypoint and the destination keypoint. (Hint: check the mask value returned by the function estimating the homography)

d. Output the computed homography matrix. (The built-in homography finder also applies a non-linear optimization step at the end; you can ignore or disable this step it if you wish.)

Image Data

The assignment folder contains a "HW3_Data" folder with five images. *src_0, src_1,* and *src_2* contain objects to be detected; *dst_0* and *dst_1* are target images. You need to detect each object in each target image, respectively. Namely, you should show the matching results before and after RANSAC for the following 6 image pairs: *(src_0, dst_0), (src_0, dst_1), (src_1, dst_0), (src_1, dst_1), (src_2, dst_0), (src_2, dst_1)*

What to Submit

You should submit a compressed file including your source code (should be well commented) and report. The report should include:

1. A brief description of the programs you implement.
2. Show the results of intermediate steps as listed in the descriptions above.
3. An qualitative analysis of your test results: how well does the method work? Does it work equally well on the different examples? If not, why might the performance be better in one case then the other? Note that the main goal is to locate the objects in the given images. We could transform the entire object image and overlay on the target image, using the computed homography, but this is not asked for. It will suffice to make your judgment based on results of feature matching (after homography computation).