Fall 2016 - CS 477/577 - Introduction to Computer Vision

Assignment Seven

Due: 11:59pm (*) Wednesday, December 07.

(*) There is grace until 8am the next morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted.

Weight: 7 points

This assignment must be done individually

Some parts of what you need to do for this assignment is quite close to code available on-line, **but you are responsible for your own implementation**.

General instructions

Unless there is a good reason others, you should use Matlab for this assignment. **If you want to use any other language, you must discuss this with the instructor at lease one week before the due date**. If permission is granted, you will have to sort out any image handling and numerical library support on your own or as a group. (The IVILAB has support for C/C++ , with example files geared towards this course that can be made available on request).

You need to create a PDF document that tells the story of the assignment, copying into it output, code snippets, and images that are displayed when the program runs. Even if the question does not remind you to put the resulting image into the PDF, if it is flagged with ($), you should do so. I should not need to run the program to verify that you attempted the question. See

http://kobus.ca/teaching/assignment-instructions.pdf

for more details about doing a good write-up. While it takes work, it is well worth getting better and more efficient at this. A substantive part of each assignment grade is reserved for exposition.

Assignment specification

This assignment has two parts, A and B that are required for both undergrads and grads, and one more part (C) which is required for grads. Undergraduates especially interested in the slide-frame matching problem can substitute C for A. Part C is not particularly difficult, and undergrads are encouraged to consider it, either for extra credit, or instead of A.

To simplify things, you should hard code the file names in the version of your program that you hand in. You can assume that if the grader needs to run your code, they will do so in a directory that has the files linked from this page.

Part A (Required for both undergrads and grads)

(Undergraduates especially interested in the slide-frame matching problem can substitute Part C).

RANSAC. This http://kobus.ca/teaching/cs477/data/line_data_2.txt contains 300 points, of which at least 1/4 can be assumed to lie on a pretty good line. Write a program that reads in these points, makes a scatter plot of them, finds a line using RANSAC, and plots that line on top of the scatter plot. Hand in your code to produce this plot. In addition, put the plot into a PDF, and report the line found, the error of the line based on what you found as the inliers ($).

You should use the perpendicular distance from the inlier points to the proposed line as your error measure. Don't forget to refit the line after establishing the inliers using homogeneous least squares.

Note: In class we mentioned two methods for determining inliers versus outliers (threshold, and best N%). You can use either here, but since I am telling you a lower bound on inliers, you may find using the best N% to be a bit easier. If you want to try using a threshold, and/or both methods, you could consult the scatter plot help you choose a threshold.

Tip: See http://en.wikipedia.org/wiki/RANSAC for more on RANSAC, including the standard formula that allows you to choose the number of iterations sensibly.

Part B (required for both ugrads and grads)

Homography. Implement both methods discussed in class for computing the homography between two planar images based on 4 or more point matches. For the first method you will need to set up the matrix along similar lines to what we did with camera calibration (HW 3). For DLT, follow the notes. The equations for both methods are solved using homogeneous least squares. Explain what you did in your writeup **($)**.

Test both methods using the data from Assignment Six. For each pair of images (slide and video frame), collect 8 SIFT keypoint matches using manual mouse clicking. Do not spend too much time getting perfect matches. In fact, if the points are completely free of error then it may be harder to understand what is happening. Visualize your results in your write up include nice explanation ($).

Now take a subset of 4 points from those 8 as input to a test program that computes the homography. Use the computed homography to map all 8 source points (from the slide) to the video frame. In the video frame images, display the 8 clicked points, and the 8 mapped locations differently (e.g. different color). Ideally they should be relatively close, and in particular, the 4 points used to establish the homography should be on top of each other. Hand in three pairs of images for each method showing the marked points in both, and the mapped points in the video frame **($)**.

Part C (Required for grads, optional for undergrads)

Homography and RANSAC. Now use RANSAC to improve the keypoint matching to search for a set of N keypoints that all agree about the homography. From your previous work on Assignment Six, you probably have some idea about what a good number for N is. Does RANSAC help? If you already were sufficiently conservative with N so that most matches are good, consider if you can get away with a bigger N when you use RANSAC. (A bigger N is better if you can retrieve some good matches you would have done without otherwise which then leads to a more accurate homography).

Provide revised figures with keypoint matches as you did for Assignment Six for the method you deemed best and worse, or, if that is not clear, choose two of them **($)**.

Comment on any difference in the two methods for solving for the homography **($)**. (Note: In general the DLT method is regarded to be more stable, but I have no idea if this will be the case here).

Tip: Ensure that the 4 matches in each group are different. If you naively randomly sample, then you might draw the same match twice. This will lead to a degenerate equation that might produce an error when you try to solve it. (This may happen on occasion anyway, so if you are able to trap that error and ignore it, your code will be more robust).

What to Hand In

As usual, the main deliverable will be PDF document that tells the story of your assignment as described above. Ideally the grader can focus on that document, simply checking that the code exists, and seems up to the task of producing the figures and results in the document. But you need hand in your code as well as follows.

If you are working in Matlab (recommended): You should provide a Matlab program named hw7.m, as well any additional dot m files if you choose to break up the problem into multiple files. Do not package up the files into a tar or zip file–this messes up the D2L conventions.

If you are working in C/C++ or any other compiled language you need to discuss this with the instructor at least one week before the due date: You should provide a Makefile that builds a program named hw7, as well as the code. The grader will type:

```
make    ./hw7
```

You can also hand in hw7-pre-compiled which is an executable pre-built version that can be consulted if there are problems with make. However, the grader has limited time to figure out what is broken with your build. In general a C/C++ solution will require nonstandard libraries, and you should discuss with the instructor how they can be provided as part of your submission, or assumed to exist on the system that is used for testing.

If you are working in any other interpreted/scripting language (again you need to discuss this with the instructor at least one week before the due date): Hand in a script named hw4 and any supporting files. The grader will type:

```
./hw7
```