Spring 2019 - CS 477/577 - Introduction to Computer Vision

Assignment Eleven

Due: 11:59pm (*) Wednesday, April 17.

(*) There is grace until 8am April 21, as the TA will not grade assignment before that Sunday. However, once the TA starts grading assignments, no more assignments will be accepted.

Weight: Approximately 5 points

This assignment MUST be done in groups of 2-3. Likely you will simply stay in the group that you were in for assignment 10. If you want to switch, you need to sort this out with both your current group and your new group.

You should create one PDF for the entire group, but everyone should hand in a copy of the PDF and the code to help the TA keep track.

Make it clear on the first page of your PDF who your group is, and a brief explanation of how you worked together. Group reports are expected to be higher quality than what the individuals would have done on their own.

General instructions

You can use any language you like for this assignment, but unless you feel strongly about it, you might consider continuing with Matlab.

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.

30% of the assignment grade is reserved for exposition.

Learning goals

- Using RANSAC to fit relatively low dimensional models
- Understanding and computing planar homography
- Familiarity with geometric constraints (and RANSAC) making matching robust d
- Exploring computational ideas using synthetic data (grads)
- Additional experience with chicken and egg algorithms, using K-means as an example (grads).
- Understanding image stitching (optional)

Assignment specification

This assignment has five parts, three of which is required for both undergrads and grads, one that is required for graduate students, and one that can be substituted and/or . If students would like additional optional questions they should contact the instructor.

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)

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 the DLT method discussed in class for computing the homography between two planar images based on 4 or more point matches. Report **($)** RMS error of the transformation for 4, 5, and 6 randomly generated pairs of points inside [0,1]x[0,1] over 10 samples. Do your results suggest that your code is working **($)**?

Test DLT using the data from assignment 10. For each pair of images (slide and video frame), collect 8 matching points using manual mouse clicking. These points need not be keypoints, and in fact, you will find it easier to simply use points that **you** find distinctive such as the dot in an "i" or the corner of a box. However, 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 both ugrads and grads)

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 9, 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 9 for the method you deemed best and worse, or, if that is not clear, choose two of them **($)**.

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).*

[End common parts. As usual, ugrads can do grad problems for modest extra credit].

[Grad students should do either parts D or E. If you do both, modest bonus points are available]

Part D (one of two options required for grads)

Part of the learning objective for this question is to structure exploration of computational ideas using synthetic data. Generally, if you are trying out new ideas, you should begin with synthetic experiments where you know the answer, and precisely how it relates to your assumptions. As things get complicated, mediocre results from real data are hard to distinguish from bugs. Generally as we develop complex methods, we go through cycles of development, testing on synthetic data, and then testing on real data.

Explore using K-means to "segment" points into the lines they line on. You will have to general N lines, and the points on them according to an error model (e.g., Gaussian). You will also want to try adding outliers (e.g., points uniformly distributed over the image). Questions to ask include: 1) the effect of increasing noise; 2) the effect of increasing the number of outliers; and 3) the effect of K. For example, what is the effect of knowing N versus having to infer it. Can it be inferred by trying several values of K that includes N?

Careful with your conclusion here, as increasing K will generally reduce error, even if you are not happier with the results. It is instructive to look at the plot of error versus K.

You should use your judgment about what to try and what to report. The reader should acquire a good sense of how this works, and not be feeling as though there were obvious things that should have been tried, which, in grad school, means the advisor sends you back to do another experiment.

Part E (one of two options required for grads)

Implement image stitching using SIFT, homography, and RANSAC and demonstrate that it is working using some pictures that you choose. Most folks will find it easy enough to simply take some pictures with their phone, but if this is not easy for you, let me know and I will provide some. The pictures you use need to have some overlap, and some textured areas in the overlapping regions.

Possibly the hardest part is making the bigger picture from the smaller ones, **once** you have the homography. If you think about painting the smaller images into a bigger "canvass" you will realize that the first one is easy, but the canvass pixels for the subsequent ones correspond to small region in one or more images that is not exactly a pixel. Perhaps a good strategy for development is to first paint with the color of the mapped pixel rounded to the nearest integral values, and then try to improve the result with the appropriate weighted average. If this makes a difference, be sure to tell the TA about it.

You should report what you did in some detail, and provide at least two examples of the result of your stitching program.

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. So you need hand in your code as well.