CS 477/577 - Introduction to Computer Vision

Assignment Eleven

Due: 11:59pm Wednesday, May 04 (FIRM (*)).

(*) There is no official grace because due to UA rules assignments cannot be due beyond the last day of classes. We will try to grade as soon as possible after the due time, and once we start grading, no more assignments will be accepted. However, we will not bother formally closing the submission site.

Weight: Approximately 5 points

This assignment can be done in groups of 2-3. You should create one PDF for the entire group, but everyone should hand in a copy of the PDF to help me 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. If the group consists of both graduate and undergraduate students, make it clear to what degree undergraduates contributed to any graduate parts or extra parts. Group reports are expected to be higher quality than individual ones.

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) for making matching robust.
- Exploring computational ideas using synthetic data (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. 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. The file linked here (also in D2L):
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 9. 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 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 D1, or D2. If you do more than one, modest bonus points are available. Also, in the final, optional bonus assignment, you will be able to choose any of the selections you did not choose previously]

Part D1 (one of two options required for grads)

To firm up our understanding of the **fundamental matrix**, we will study matches across two arbitrary views of the same scene. We will ignore parts of both images that do not occur in the other image for matching. You are encouraged to use your own images if you are so inclined. However, we are now into t the epically busy time of year, so in case you would like to simply use existing ones, you can use the ones linked below. The two cameras and scenes can be arbitrary. The cameras could be the same physical camera moved between two positions, even with focal length changes, or even two different cameras. The main restrictions for the purposes of this assignment is that cameras must have reasonable translation between them, and the scene should not be planar. (In these special cases the relationships between points are a homography, and not interesting if we want to study the fundamental matrix.)

Links to images in case you want to use mine (also in D2L).
http://kobus.ca/teaching/cs477/data/FM-inside-1.jpg
http://kobus.ca/teaching/cs477/data/FM-inside-2.jpg

1. Using skills developed in HW3, efficiently get the coordinates of 20 matching image points. In you do not think you have a good system for this, I suggest using a drawing program to insert modest sized open circles onto both images to set up visually distinctive “targets”, and numbers beside the circles. Then you can plot the points into the image later to check how accurate they are. Another thing that I find helpful is to have a way that coordinates of clicked points go into a file. My personal method is based on some code I wrote a very long time ago, but I assume that there are good alternatives (let me know!). Hand in your two txt files with the coordinates on matching rows being the same image coordinate as seen by each of the two cameras **($)**. In your report, provide both of your images with your marked points clearly visualized as such **($)**.

2. **Derive** a way to compute the fundamental matrix **($)**. I think the easiest way (and it is fairly straightforward) is to notice that for every matched pair **x** and **x'**, the basic equation provides an equation in the 9 unknowns of F. While you can do fancy things to get away with 8 points, the simplest formula assumes that you have 9 or more equations. In this assignment we will use more than 9 unless you particularly want to explore further[1].

   Try not to overthink the math. Each match gives you one equation, which can be interpreted as a dot product.

3. Use your formula to solve for F using 12 points, and report how both these points (training) and the other 8 (or so) points that you did not use (training), fit the fundamental constraint. You should report error in terms of RMS over the values that should be zero if everything were working properly **($)**. For debugging you may want to implement the next part in parallel.

4. Implement the capability to do the following given image-pairs, point-pairs, and the derived F. For each point in one of the images, draw the epipolar line for it in the other image. Be sure to also include your marked points in such a way that they are not obscured by any of the lines. Hand in the result using an F derived from all your points (**$**). Is there any obvious structure in the lines? Explain what is going on **($)**.

   Hint. To draw the epipolar line for x, you can compute the vector Fx, and view $x'^{T}(Fx)=0$ as the equation of a line.

   If you think it will help you. I have linked the Matlab function "draw_line" links below (also in D2L). This is not the same function as "draw_segment", as it draws the line extended to the image boundaries. These bits of code are not very robust. They are "free software", but not as in "free beer", nor as in "free speech", but as in "free puppy".
   http://kobus.ca/teaching/cs477/code/draw_line.m

Part D2 (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 instructor about it.

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

---

[1] Using as few points as you can get away with is important for RANSAC, but the matrices so found often will not be very accurate. This is why it is very important to refit the inliers if the model is not that stable as is the case for going after the fundamental matrix.

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.