CS 477/577 - Introduction to Computer Vision

Assignment Thirteen (Extra/Bonus)

(There are 13 assignments, but the total assignment grade is based on 12 assignments. If you have been doing well on assignments 1-12, then this assignment is optional).

Due: 11:59pm Wednesday, Dec 11 (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, and the instructor will not start grading before 9AM Thursday morning.

Weight: Approximately 6 points

This assignment can be done in groups of 2-3, but this is not required.

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.

If you worked in a group, 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 what the individuals would have done on their own.

General instructions

You can use any language you like for this assignment.

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

- Understanding basic stereo in humans, astronomy, and machines.
- Understanding the fundamental matrix (grads)

Assignment specification

This assignment has three parts, two of which is required for both undergrads and grads, and one that is required for graduate students for full marks.

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)

In this problem, we will implement simple stereo processing. There is Matlab code in various tool boxes and miscellaneous sources to do this sort of thing, but **you should not use them**. This would defeat the purpose of the assignment.

The image sets

http://kobus.ca/teaching/cs477/data/stereo-pair-1.tiff
http://kobus.ca/teaching/cs477/data/left-1.tiff
http://kobus.ca/teaching/cs477/data/right-1.tiff

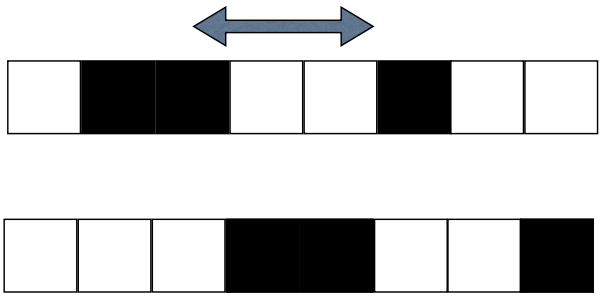
and

http://kobus.ca/teaching/cs477/data/stereo-pair-2.tiff
http://kobus.ca/teaching/cs477/data/left-2.tiff
http://kobus.ca/teaching/cs477/data/right-2.tiff

(also in D2L) are sets of left and right views of a "random dot stereogram". One image shows the pair together, and if you look at it in the right way, you should be able to see the 3D effect. (Try focusing between beyond the images (and relax). Some people suggest that you are looking at them almost cross-eyed. You may need to vary the distance to the screen to get things to pop into place). Getting the 3D effect is not that easy to do the first time, and this assignment does **not** require you succeed. But it is worth trying to make it work, as it will help you understand what is going on, as well as provide you some sense of the correct answer.

If you manage to see the stereo effect, then you will know that there are only two interesting "distances", background and foreground. We will use that assumption later.

Your first task is to write a program that computes the disparity between the image locations. You can assume the simple stereo setup that we worked with in class. In particular, the match for a point (x,y) is (x', y). In other words, to find a match for a point in the one image, you search along the corresponding row in the second image. To find a match, proceed as follows. Consider a maximum disparity of 15 (either to the left or the right). We want to step through the possible shifts and test each one for a best match. In Matlab notation this is -15:15. For each test shift, we want to ask how well the neighborhood of our point matches the neighborhood of the match under consideration. We can do this by taking the dot product of two vectors centered on each that are long enough to capture the similarity, but not so long that they are no longer local. Try 21 to start (you might find it easier to think of "21" as -10:10 in Matlab). Have a look at the figure before reading on.



Test similarity at each candidate disparity and keep the best one.

Suggested similarity is the dot product of the normalized vectors

As suggested in the figure we can measure similarity by taking the dot product of the vectors of length 21 centered on the test alignment. You may find that normalizing the vectors first (to unit length) works better, but for this you need to test that the vector is not zero first, and skip normalization, least you divide by zero.

You will notice that you cannot do much at image edges, and if you were to try to do so, you will be spending a lot of time debugging out of bounds indexing. So, you can assume that the disparity on the left and right margins is zero. Restrict your left-right search as appropriate. (Specifically, if the disparity is in –D:D and the matching vector indexes –C:C, then I think restricting your horizontal index from 1+margin to n-margin, where margin=D+C should work (in Matlab)).

Visualizing disparity. Your program should record the value of the shift for the best match for each point. Your program should display your disparity map as an image, appropriately scaled, for each of the two pairs. Except for bad matches due to noisy data or edge effects, points infinitely far away have zero disparities, and disparities for closer points are either negative or positive as a consequence of which view is truly from a left camera versus a right one and imaging conventions. For visualization, we will simply use positive disparities. Thus, you can either record the absolute value of the disparity, or reason about whether the setup will give you negative or positive ones and negate if needed, or you could be really clever and not even look for disparities in the wrong direction. Regardless of your strategy, create figures that are generally black for zero disparity, and brighter for higher absolute values of disparities (closer points). Put the two disparity maps into your report with an informative caption (**$**).

Hard code the image file names so that if the grader were to run your program, they would see disparity maps for both data sets. You can assume that the grader has copies of data files.

Your program should also output the average of the largest 10% absolute value disparities, to estimate the single interesting distance relative to the background. Report this disparity (**$**). Finally, supposing that pixels are 0.025cm, and that the focal length is 2cm (my guess of the human eye size), and the distance between eyes is 10cm, what distance would be associated with the surface using the formula developed in class? (You can assume that in the distance equation developed in class that d-D is approximately *d* as *D* is much smaller than *d* **($)**.

Part B (required for both ugrads and grads)

People generally believed that the earth was in the center of the universe until recent centuries. A key academic argues was the following: If the earth went around the sun, the stars should shift positions. This reasoning is correct, but this was not observed until 1838 because the shift is **very** small (and corresponds to the stars being almost imaginably far away).

Recall how your thumb, held at arms length, shifts in apparent position as you switch from viewing it with your left eye to right eye. In the same way, as the earth goes around the sun, observing the star at distinct times is like having two eyes, millions of miles apart.

Consider viewing a close star a few months apart. You can determine that you are looking exactly in the same direction by aligning stars in your two photos. In fact, since most stars are significantly further away, the only interesting number is the shift ("disparity") of the few nearest star with respect to the background stars. Let's suppose that the optical system has a focal length of two meters. Also, considering that the distance from the earth to the sun is 93 million miles, it is not unreasonable that the distance between the two "cameras" is 80 million miles. (That is one **big** stereo rig!). Finally, your measurement for the disparity is 6.2 micro meters. (A micro meter is 1 millionth of a meter). That is pretty small --- no wonder earlier attempts to notice it failed. Based on this, how far away is the star in: a) miles (**$**); and b) light years (**$**). You must show your work. Visit https://en.wikipedia.org/wiki/List_of_nearest_stars_and_brown_dwarfs to name the star ($).

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

Part C (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 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. If 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 and least squares.

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

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.