B657 Assignment 2: Feature Detection and Projective Transforms

Spring 2014
Due: Sunday March 9

As we saw in class, projective transformations can be used to model changes across photos of the same scene that are taken from different viewpoints. In this assignment, we'll give you a set of images of the same scene taken across by several people from different viewpoints across the span of many months. Your job is to transform them so that it appears that they were all taken from the same viewpoint. We can then put the images together into a movie which shows how the scene changes over time, as if captured by a stationary camera (like a webcam). To do this, you'll first need to develop an automated technique to compute the transformation between two images, which involves estimating a homography using feature point detection and RANSAC. Then, you'll need to write code that applies a projective transformation to an image in order to "synthesize" the views from an imaginary static camera.

You may work on this project individually or in groups of two. If you work in a group, you should turn in a single project report and single set of source code files. Both members of a group will receive the same grade.

As with assignment 1, we recommend using C/C++, and we have prepared some skeleton code that may help you get you started (see details below). We recommend using C/C++ because computer vision algorithms tend to be compute-intensive, and your code may be frustratingly slow when implemented in higher-level languages (like Matlab, Java, etc.). However you may choose to use a different programming language, with the restriction that you must implement the image processing and computer vision operations yourself. You do not have to re-implement image I/O routines (e.g. you may use Matlab's imread and imwrite functions). You may use built-in or third party libraries for routines not related to image processing (e.g. data structures, sorting/searching algorithms, etc.).

Please use the Forums on the OnCourse site to ask questions about this assignment.

What to do

1. Download the C/C++ skeleton code and test images (available via OnCourse), and make the sample program on a Linux machine:

   ```
   tar -xzf a2.tar.gz
   cd a2/
   make
   ```

   The code has been tested on the CS Linux machines (e.g. tank.cs.indiana.edu), so we suggest using one of those. You may use another development platform (Windows, MacOS, etc.), but you may have to modify the skeleton code to get it to compile.

2. Implement a function that finds corners in an image, using the Harris corner detector that we discussed in class. Remember to smooth the image first to eliminate noise (use your Gaussian filtering code from A1 to do this).

3. Implement a technique for extracting a local descriptor for each corner detected in step 2. Here's one approach to try (although you may want to modify it based on your experiments in later steps). Look at the four small neighborhoods (each of size, say, 8x8 pixels) to the upper-left, upper-right, lower-left, and lower-right of the detected corner. Compute the gradient magnitude and gradient orientation at each pixel. Quantize the gradient orientation into, say, 8 bins. For each of these 8 quantized orientations, count the number of pixels in each neighborhood having that quantized orientation and

magnitude above some threshold. This produces a 4×8=32 dimensional descriptor for each detected corner point.

4. Write a function that takes an input image and applies a given 3x3 coordinate transformation matrix to produce a corresponding warped image. To help test your code, here's an example of what "lincoln.jpg" (which we've provided) should look like before and after a projective transformation,


with this transformation matrix:

$$\begin{pmatrix} 0.907 & 0.258 & -182 \\ -0.153 & 1.44 & 58 \\ -0.000306 & 0.000731 & 1 \end{pmatrix}.$$

Note that this matrix assumes a coordinate system in which the first dimension is a ~~row~~ column coordinate, the second is a ~~column~~ row coordinate, and the third is the extra dimension added by homogeneous coordinates.

Use bilinear interpolation to do this warping.

5. Now, write a function that estimates a homography (projective transformation) between two images. To do this, you'll first want code that finds corresponding features between the two images. (In other words, after detecting features in image I and J, your function should find pairs of points for which the distance between the 32-dimensional feature descriptors is below some threshold. Of course, many of these correspondences are going to be noisy, so one needs RANSAC to find the signal in the noise. We will discuss RANSAC in class on Monday Feb. 24).

6. Combine all above techniques to realize the image sequence warping application. Given a sequence of two or more images, the program should compute a homography between each image and the first image, and then produce a warped image that appears to have been taken in the first camera's coordinate system. We will provide some test data via OnCourse.

As with Assignment 1, this assignment is purposely left open-ended. For example, the above steps have several parameters that will need to be set and other design decisions that will need to be made. In these cases, you'll need to use your intuition and some experimentation to choose good value(s) or good approach(es).

Output. In addition to producing the sequence of warped photos, your code should also produce a visualization of all detected interest points for each input image. You might find "overlay_rectangle" helpful for doing this (provided in the skeleton code).

Test images. We'll put several test image sequences on OnCourse. Please show the results of your system on these image sequences in your project report. You may also want to try your code on simpler images to help test and debug your code.

Extra credit. We'll award extra credit for submissions that go beyond the requirements of the assignment; make sure to describe this work in your documentation file.

Academic integrity

You and your partner may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about C/C++ syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials in the documentation of your source code. However, the code that you (and your partner, if working in a group) submit must be your own work, which you personally designed and wrote. You may not share written code with any other students except your own partner, nor may you possess code written by another student who is not your partner, either in whole or in part, regardless of format.

What to turn in

Turn in two files, via OnCourse:

1. Your C++ source code compressed as a .zip or .tgz file. Make sure your code is thoroughly debugged and tested. Make sure your code is thoroughly legible, understandable, and commented. Please use meaningful variable and function names. Cryptic or uncommented code is not acceptable.

2. A separate text report in PDF format. Explain how you implemented it (e.g. what data structures you used), what design decisions and other assumptions you made. Give credit to any source of assistance (students with whom you discussed your assignments, instructors, books, online sources, etc.). You are encouraged to use write your report in Latex.

3. The output images for our test sequences. You can either submit a .zip or .tgz file containing a set of output images (PNG or JPEG), or you can combine the images into a movie (e.g. using the ffmpeg program).

Grading

We'll grade based on the correctness, style, and the project report. In particular: Does the code work as expected? Does it use reasonable algorithms as discussed in class? Did it produce reasonable results? Was it tested thoroughly? Does it check for appropriate input and fail gracefully? Is the code legible and understandable? Is the code thoroughly and clearly commented? Is the project report adequate?