CS223b: Introduction to Computer Vision
Assignment 3: Image Stitching

Due date: Thursday, February 3rd 23:59 PST
You may work in teams of up to 3 students.
Submission via e-mail with subject "Assignment 3" to cs223b@gmail.com

1. Image Stitching

For this assignment, you will be implementing a panoramic image stitching package, using a subset of the techniques presented in the lecture on image stitching. You may use Matlab or C++. If you're working in C++, you may want to use OpenCV.

We strongly recommend that you follow the steps laid out here, since this can be a large assignment if you start from scratch, and we think that following these discrete steps will minimize your workload. This handout seems long because we've tried to give you a lot of information to help you stay on track.

With that said, you are free (encouraged) to explore other creative solutions, provided that you clearly document them in your submission.

1.1 Image acquisition

Use the webcams you were provided with for ASSIGNMENT 1 to collect a series of several images that you'll stitch together. Start at the far left of your scene, and for each new image, rotate the camera about its center until around half of the image is "new". That is, you should have about 50% overlap between each pair of successive images. The more you constrain your camera to rotate only about its vertical axis, the happier you'll be when you're building your panorama.

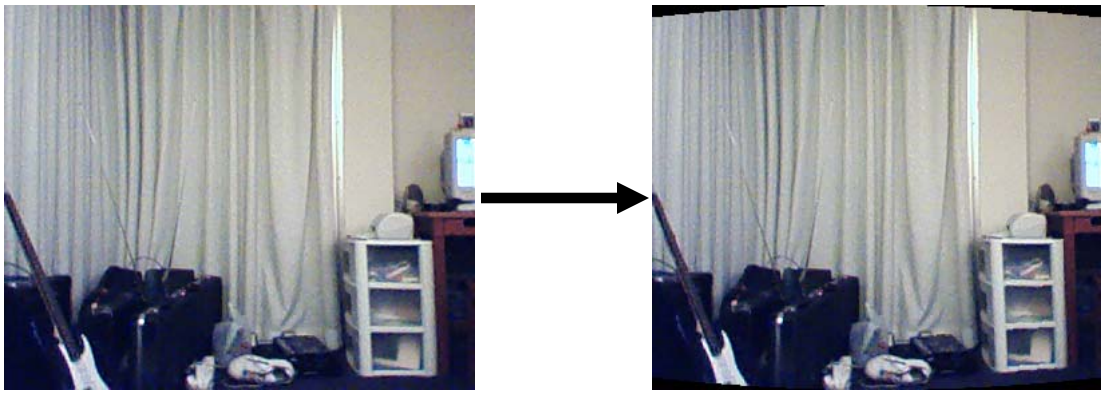
You may use a different camera if you like, but you will need to know the focal length of the camera, and you may find that working with higher resolutions is computationally demanding.

Here are some example images that worked well using the techniques required for this assignment:

1.2 Cylindrical warp

Write a function that takes an input image and a focal length and warps the image into cylindrical coordinates.

You results should look something like:



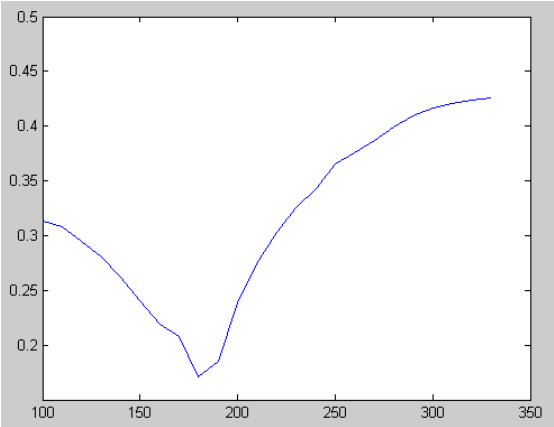
Run your function on the images in your image sequence. From now on, you'll be working only with the warped images.

1.3 Coarse alignment

For each pair of images, you will need to first generate a coarse estimate of the proper alignment between the images.

Write a function that takes two images and returns a coarse estimate of the horizontal alignment of the images.

We recommend that you do this by sliding the left (cylindrical) image across the right (cylindrical) image at discrete intervals (e.g. 15 pixels), and finding the shift that minimizes some error metric (which you should choose). When the first image in the above image sequence is slid across the second image, the error as a function of image shift looks something like:



You should come up with an error metric that gives you a clean result like this in most cases. You can assume a reasonable minimum and maximum shift.

Note that this is not the technique that was presented in class; a more sophisticated technique uses image pyramids to find the optimal transformation between two images at successively higher resolutions. You may implement image pyramids for extra credit, but finding a coarse alignment just by shifting is acceptable for this assignment.

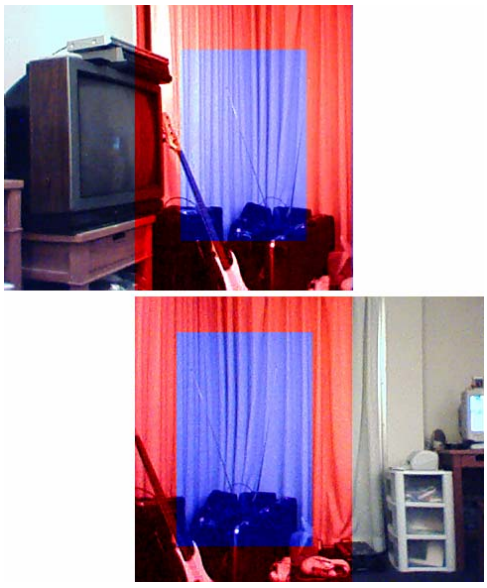
1.4 Optical Flow

Now that you have a rough estimate of the horizontal shift between two images, you should use optical flow computation to find a more precise transformation. For this assignment, you will only be required to compute a horizontal and vertical shift; you do not have to worry about rotation, skew, etc.

You will probably want to align the images according to the transformation you found in 1.3, then extract the overlapping region from each image. These are the regions that you'll use for computing optical flow.

Write a function that takes two such regions and computes a (u,v) shift that optimally aligns them.

It's fine if you simplify the problem by ignoring a few pixels at the boundaries of the overlap, so you don't have to worry about the regions "shifting off of each other". To illustrate this graphically, this figure shows two images that have been approximately aligned (using the technique you implemented in section 1.3), with their approximate regions of overlap highlighted in red. The regions highlighted in blue are the "trimmed down" overlapping regions that you might use for flow computation.



For this component of the assignment, you can use several approaches, including the Lucas-Kanade flow algorithm, the optical flow technique discussed in your textbook, or any of OpenCV's optical flow methods. Note that even if you're not using C++ for this assignment, you might still want to look at the OpenCV source for inspiration.

You are encouraged to compute your optical flow on grayscale images; this will be much easier. Your final results should be in color, but computing your shift vector on grayscale images is fine.

1.5 Blending

After computing the flow vector between two images, you have all the information you need to line them up in a panorama. However, if you just shift one of the images and drop it on top of the other, there will be an obvious seam at the boundary between the images. **You should implement a function that takes two images and a two-dimensional alignment vector and returns a new, larger image in which the images have been stitched together and a few pixels around the seam have been nicely blended.**

You can use a simple linear blend, where you pick a fixed blend region (say 10 or 15 pixels) and for each pixel in that region, computed a weighted average of the corresponding pixels in the left and right images. Pixels at the left of the blend region should look mostly like the corresponding pixels in the left image, etc.

1.6 Panorama Generation

The last step is to write a function that glues all the parts together. Your main program should take a list of image files, ordered from left to right, and perform all the above steps for each successive pair of images (1 and 2, 2 and 3, 3 and 4, etc.). The blending step should always use the accumulated panorama as the "left" image, so you gradually build up your panorama from left to right. You will also probably want to cut a few pixels off from the top and bottom of your panorama to get rid of the ugly gaps induces by cylindrical warping.

This figure shows a glorious panoramic view of Dan's apartment, computed from the image sequence presented in 1.1, using the techniques presented in this assignment.

Deliverables:

- A series of numbered source images
- The result after stitching your first two images together, along with the offset you computed for these two images for part 1.3 and the flow vector you computed for part 1.4
- Your final panorama
- Your source code, in Matlab or C++
    - If you use Matlab, please provide us with a simple script that we can run that loads your images, runs whatever functions need to be run, and shows the resulting panorama.
    - If you use C++, please provide us with a project file or Makefile so we can compile your code "out of the box", and a command line that we can run or magic button we can click that loads up your images, builds a panorama, and outputs or displays the result.
- A document telling us what algorithms you used, what interesting problems you encountered and/or solved, and whether you implemented any optional features that you want us to know about

Finally, please state the percent effort that each individual member of your team has contributed to this assignment (numbers should add up to 100%). Submit everything with the subject "Assignment 3" to cs223b@gmail.com .

Extra credit

There are lots of opportunities for extra credit on this assignment. It's fun to make panoramas from your own images, so enjoy, and be creative.

Some possible "extras" include:

- Use a fancier stitching technique... a weighted blend still leaves some seam or some ghosting, especially with a hard-coded blend region. "Gradient domain fusion" – fusing the derivatives of the images, rather than the images themselves – is a popular technique.
- Use image pyramids to compute optical flow, rather than the shifting approach we presented above.
- Normalize the images cleverly to avoid the large variations in brightness that are present in the sample panorama.
- Use a feature-based flow technique instead of a dense flow technique.
- Compute affine flow – rather than just a 2d vector – and warp the images accordingly, to account for rotations and shearing effects between images.

References

A really nice summary of the overall process of creating panoramas is available at:
http://www.ddewey.net/pics/vpt/technical/