Programming Project #5: Automatic Image Stitching

CS498: Computational Photography

Due Date: 11:59pm on Monday, Nov. 11, 2013

Photos by Russell Hewett

Overview

In this project, you will create a program to automatically stitch two or more photographs into a single image. In doing so, you will explore correspondence using interest points, robust matching with RANSAC, and homography. You can also investigate cylindrical and spherical projection and other extensions of photo stitching and homography as bells and whistles.

To stitch two images, you must first establish point correspondences. In the first part of this assignment, you will do so manually. In the second part, you will use automatically computed interest points and SIFT descriptors and RANSAC to establish the correspondences. With four or more correspondences, you can compute the homography matrix H that maps between coordinates of two planes. Then, you can create a single surface (e.g., a plane or cylinder) and fill the pixels of that surface by blending from the original images. For the core project, you can just choose the plane formed by one of the two images as this surface. That's it! But get started early, because it may not be as easy as it sounds.

You may not use some built-in matlab functions, such as cp2tform, imtransform, tformarray, tformfwd, tforminv, and maketform. You may use interp2 and general linear algebra functions, such as svd. You may not use outside code that is not linked directly in this project description.

Part I: Semi-Automated Stitching (60 pts)

For this first part, you will manually select the corresponding points and write code to compute the homography (warping) and compositing. Each step is described in detail

below. Create a function
mergedImage = userstitch(im1, im2)
that stitches two images after asking the user to provide at least four correspondences.

Choosing Photographs
You're strongly encouraged to use your own photographs. You are also welcome to try using samples by Russ Hewett (small, large). The photographs must be of the same scene, and the translation of the camera must be small compared to the depth of the scene. The pictures should overlap quite a bit but not entirely (e.g., 30%-70%). Examples of scene include panoramas, waterfalls, and interesting indoor environments -- anything with interesting texture that is not well-contained within a single photograph.

Manual Point Correspondence
For a pair of photographs, click on four or more (I suggest more) pairs of points that correspond to the same place on the same object in the scene. You can do this using cpselect or a tool that you write on your own with ginput.

Computing the Homography
Write a function
H = computeHomography(x1, y1, x2, y2)
where (x1, y1) and (x2, y2) are corresponding pairs of points (e.g., x1(i), y1(i) in the first image correspond to x2(i), y2(i) in the second). The homography H is a 3x3 matrix that maps from one plane to another in 2D homogeneous coordinates (x/w, y/w, w). Because homogenous coordinates are defined up to a scale, H has only eight degrees of freedom (DOF). You should solve for H using the "Normalized Direct Linear Transform" method described in class. First, compute transforms T1 and T2 that translate and uniformly scale the points of each image to have zero mean and unit norm. Compute Hn for the normalized points by solving a system of linear equations using singular value decomposition (svd in Matlab). Finally, unnormalize Hn by H=inv(T2)*Hn*T1.

Compositing
The last step is to create a single image that combines the different parts of the scene captured by each photograph. Write a function
mergedImage = stitchPlanar(im1, im2, H)
that projects both image onto a planar surface and blends. The easiest way to do this is to create two larger images in the same coordinate space as the first image -- one for each projected image. Copy pixels from the first image, and warp pixels from the second image using the recovered homography (hopefully, this still feels familiar from the morphing project). Adjust the intensity of the 2nd image so that its average intensity matches the average intensity of the 1st image in the overlapping region. Then, create a mask (e.g., based on whether a pixel is closer to the center of image 1 or image 2) and blend. For extra points, you can create two images, one of the projection from each image and then to blend using Laplacian pyramid blending.

Part II. Fully Automated Stitching (40 pts)

Before working on Part II, make sure that you have Part I working. Now, it's time to fully automate the process. First, you should automatically compute interest points and descriptors, using provided code. Then, you should find initial matches to those descriptors and, finally, use the RANSAC algorithm to find a good set of matching points with the homography that corresponds them. Create a function
mergedImage = autostitch(im1, im2) that combines these functions and the ones developed in Part I to automatically stitch provided images.

Computing Interest Points
To compute interest points, use the SIFT code provided by Andrea Vedaldi:
[frames, descriptors] = sift(im);
I strongly recommend using the latest of the pre-compiled binaries if possible. Read the included documentation for how to use the sift.m function. You may also want to use the other included display functions such as plotsiftframe and plotmatches.
The sift.m function will return "frames" (locations) and descriptors. From "frames", you need only the x and y coordinates which are found in the first two rows of "frames".
The descriptors are used to determine initial correspondences. If stitching two images, you can find the first nearest neighbor using L2 (Euclidean) distance and then check that the nearest neighbor is less than 0.7 times the distance of the second nearest neighbor. If stitching multiple images (for extra points), you could follow the method suggested by Brown and Lowe (2007).

Finding Good Correspondences with RANSAC
Unfortunately, not all of the initial corresponding points can be trusted. Fortunately, we can deal with some outliers using RANSAC. Write a function
H = computeHomography_RANSAC(x1, y1, x2, y2)
that robustly computes the homography from a set of corresponding points that include outliers. The basic idea is to randomly select four points, compute the homography from these four points, see how many other points agree (if a point from im1 projects very near its corresponding point in im2, then it agrees). Repeat many times and choose the homography with the most agreement. You can refine the estimate of the homography by recomputing it, given the points that agree.

For compositing, you can use your stitchPlanar function.

Bells & Whistles (Extra Points)

Laplacian pyramid blending (up to 20 pts)
To get a seamless blend, you should adjust for the overall gain and then use a Laplacian pyramid blend. For 15 points, do the 2-band blending; 20 points for multi-band blending. If you've already gotten extra points for re-implementing Laplacian pyramid blending for a previous project, you can reuse it here, but do a nice before/after comparison for at least two or three panoramas.

Fun with homography (up to 10 pts)
Try other uses of homography. Put your face on a billboard photographed from the street, or blend an image into a football field. Manually select the four corresponding points and use your computeHomography function and your warping/blending code to create the final image. To get 10 points, try it on two or more images.

Stitching multiple images (10 pts)
Show the result for at least one multi-image panoramic stitching. In this case, the multiple images can be done sequentially. This is a good one to combine with the stitching on non-planar surfaces task.

Stitching onto non-planar surfaces (up to 30 pts)
There's a limit to how much rotation you can deal with when projecting onto a plane. It's much more common to use cylindrical projection for large panoramas. It's also possible to project onto a sphere or a cube. You can get up to 20 points for doing one of these, and 5 points each for doing the others. Show at least one example for each time, preferrably with multiple images. Hint: the easiest way to do this is to figure out the (non-linear) projection from im1 onto the non-planar surface; then to map im2, project from im2 to im1 using H and from im1 to the surface. For cylindrical projection, you can assume that im1 and im2 have the same focal lengths and choose a focal length that works well (e.g., set f to the diagonal length of the image in pixels).

Stitching multiple images with bundle adjustment (25 pts)
Not for the faint of heart, implement and demonstrate the bundle adjustment described by Brown and Lowe (2007) for 25 pts.

Other ideas (up to 30 pts)
If you want to try something else and you want points for it, let me know ahead of time, and I'll tell you how much it's worth.

Deliverables

To turn in your assignment, place your index.html file and any supporting media in your project directory: http://web.engr.illinois.edu/~netid/cs498dwh/proj5/, where "netid" is your netid (e.g., dhoiem). Also, e-mail me the zipped code in "netid_proj5.zip" and include a link to the project page in the text. See project instructions for details. In the e-mail, tell me how many points you think you should get for the core project and any bells and whistles.

Use both words and images to show us what you've done. Please:

- Show your favorite panorama result. Use the same images to demonstrate manual and automatic stitching. For manual stitching (required), minimally include: 1) the two input images; 2) the manually selected points; 3) each image projected onto the planar surface (separately); 4) the final stitched image. For automatic stitching, additionally include: 1) the initial matched points (use plotmatches); 2) the matched points that are consistent with your homography; 3) the stitched image.
- Now, show at least three more results. You can use some of the images provided by Russ, but at least one should be with your own photographs (taken by you).
- Include code for manual and automatic stitching and the functions that they call.
- Describe and bells and whistles under a separate heading and include relevant code.

Scoring

The core assignment is worth **100** points, as follows:

- **60 points** for implementation of the manual stitching, including userstitch (10 pts), computeHomography (20 pts), and stitchPlanar (20 pts), and results (10 pts).
- **40 points** for implementation of automatic stitching, including autostitch, <computehomography_ransac< ttl="">(30 pts), and general results and clarity (10 pts).</computehomography_ransac<>

You can also earn up to **125 extra points** for the bells & whistles mentioned above.