Problem Set 5

Assigned: April 13, 2006
Due: April 27, 2006

Problem 1 *Panorama Imaging*

In this problem we will guide you to build panorama by yourself. You are not allowed to use any other software for this problem set. We ask you to generate intermediate results at each step. You are also NOT allowed to use MATLAB functions such as `cp2tform`, `tformarray`, `tformfwd`, `tforminv`, `maketform` or `imtransform`.

Step 1. Take pictures

Take at least four pictures that you will stitch into a panorama. You may borrow a digital camera and tripod from Ce (32-D460). Here are some hints for taking the pictures.

- Make sure images overlap by $\sim 40\%$.
- Keep the same viewpoint.
- Use a tripod.
- See with distinct features (not repetitive).
- Try to keep the same exposure (use manual mode if available on your camera, but check aperture and shudder speed to make sure the the image looks nature).
- Avoid lenses with too much distortion (fish eye and other lenses that do not preserve straight lines). Avoid the wide-angle range of zooms because they tend to have such distortion.
- Down-sample the images to be smaller than 800x600 before processing. Debug with even lower resolution, e.g. 400x300.

- No moving objects in the scene for the first trial. You may shoot moving objects and try to display them properly later for extra credits.

Step 2. Mark correspondences

While it is possible to create a panorama generation program that is entirely automatic, using feature detectors and descriptors to find correspondences, in order to simplify the problem, we will skip that step and allow you to mark correspondences by hand.

Select the center picture as the reference, and mark correspondence between each image and the reference. If it is impossible to mark correspondence to the reference, mark to the nearest and then you will propagate the correspondence in a later step. For this case you may want to build a tree structure of the images, where the reference is the root and each picture can correspond to the reference picture by tracking the tree to the root.

To mark correspondence between a pair of images, you can use your favorite tool. One suggestion is to use MATLAB function `subplot` to display the two images side by side, and `ginput` to click the feature points. You may click one feature point at image 1, and then click the corresponding feature point at image 2, and so on. At the returning matrix from `ginput`, the odd rows will be the feature points at image 1 and the even rows be the corresponding feature points at image 2.

Step 3. Solve for homography

Write code to compute the homography between two images given pairs of correspondences. Setup a set of linear equations as explained in lecture to relate the 8 coefficients of the homography matrix to the correspondence pairs. Recall that the ninth coefficient (bottom-right) can be set to 1. Use the matlab command \ to find a least-square solution. Display the 3x3 homography transform matrix for each pair. Both transforms and inverse transforms are needed for generating panorama.

(*Extra credit*): refine the correspondences. For each pair, search in the local neighborhood of one image (e.g. 5x5 pixels) for the point that minimizes the SSD (sum of square differences) of the pixels in a small window (5x5 too) with the other image. If you are hard core, use sub-pixel accuracy.

Step 4. Compute the coordinate of the panorama

To compose the final panorama we need to generate the coordinate system for the big panorama picture that can embrace all the images warped to the reference frame. You may warp the corners of each picture to the coordinate of the reference frame, and then choose the minimum and maximum of the corners to determine the frame of the panorama. Use MATLAB function `line` to display the frame of each picture in the panorama in blue, and the reference frame in red.

Step 5. Warp

Since you already know the corners of each frame in the panorama, you can use MATLAB function `inpolygon` to generate a binary mask for each frame to approximately indicate the pixels that will show up from that frame at the panorama. Then apply the homography transform (before doing that you need to translate the coordinate back to the reference frame) and use `interp2` to generate the warped image. Be careful with the pixels that are outside the input picture. Modify the binary mask accordingly. You also need to generate the mask and the translated image for the reference frame, though you don't have to go through this process. Display the masks and the warped frames.

Step 6. Blend

You now have n images at the final resolution and n corresponding binary masks. You need to blend them together.

First, implement a naïve solution where you use the binary mask to blend the pictures in an arbitrary order (e.g. from left to right). Start from an empty image and for each warped picture, overwrite the pixels where the mask is 1. You will probably have artifacts at the boundary, but it will allow you to verify that the images are actually aligned.

(6.882 only) In order to blend the panorama seamless, we want to use pyramid blending as seen in problem set 2. We need to compute appropriate masks and deal with the fact that we have more than two images to blend. Our solution will be based on distance maps that indicate how far each pixel is from the boundary of the input images. You can make your life easy and use a Manhattan distance where the distance to the boundary simply becomes $\min(x, y, \text{width} - x, \text{height} - y)$. Use the same warp procedure to generate the distance map for each picture in the final frame (aligned with the reference picture). Put value 0 for pixels outside the picture. Now for

each pixel, we have n values that store the distances to the boundary of each input.

(a) Implement simple blending where, for each pixel, you use the color of the input image with the biggest distance to the boundary. Instead of being at the boundary like in the previous blending, artifacts will now occur at pixels equidistant from the boundaries.

(b) Implement a smoother version where you use for each pixel the average of the n inputs colors weighted by distance. Do not forget to divide by the sum of the distance. The transition should be smoother, but ghosting might occur.

(*Extra credit*) Generalize the above two distance-based blending to multi-scale pyramid blending as in problem set 2.

Bells and whistles (extra credit)

Implement non-linear projections such as cylindrical and spherical panoramas.

Use multiscale Harris corner detection, feature vectors, and the RANSAC algorithm to automatically stitch panoramas.

Implement ghost removal. Compute the variance of each pixel in the warped frame to detect motion in the image. Try to group such pixels into connected component to improve coherence. See `http://research.microsoft.com/users/mattu/pubs/Deghosting.pdf`

Implement lens distortion correction. Use the formula in lecture 15. Mark points on straight lines in the image and solve for the parameters under least square.

Problem 2 *Project Proposal*

The deadline for the course final project is May 18, which is approaching fast. We want you to begin working on your project now.

By Tuesday, April 25, you should have a partner for the project (or decide to do the project by yourself; either is ok), and a subject area. We will ask you for these in class.

By Thursday, April 27, you need to turn in your project proposal (this homework problem). Please describe the project topic, goal, tasks needed

for completion, along with a timeline for completing those tasks. Describe any potential difficulties you might encounter, and your fall-back plans. We'd like you to persuade us that your project is feasible, and let us know why you find it exciting (which we hope you do).

We will review your proposals and give you feedback on them by May 2.