# Main Assignment

## Data

For taking pictures we used two different cameras, a Canon SX100 (we pre-calibrated already provided by the instructor) and a Sony Cyber-shot DSC-HX9V (which we calibrated using Caltech’s Camera Calibration Toolbox).

## Cylindrical Projection

The first step in creating the panoramic image is to first project all of the images onto a cylindrical surface. This helps account for the slightly different perspectives obtained when turning the camera about its optical center while acquiring the images.

## RANSAC

To determine the homographies for the images we used the RANSAC method. We started by using the VL\_FEAT library to obtain SIFT keypoint descriptors for all of our images. For each pair of overlapping images we then ran the UBCMatch function provided by VL\_FEAT which matches two sets of descriptors based on a threshold. If the distance between two features multiplied by the threshold (1.5 for our application) is greater than the distance of the first descriptor to all other descriptors then it is considered a match.

With this set of matches we then perform the RANSAC algorithm by taking 4 random pairs out of the set of matches and computing a hypothesis homography from it. We then take this homography and test all the other matches against it to see if they still hold. We do this multiple times and keep the homography that had the greatest number of inliers.

## Stitching

To stitch all the images together into the final panorama we first select an image to be our fixed image that everything is tied to. We then compute a new homography for every other image to the left and the right of the fixed image by compositing the homographies that we calculated in the previous RANSAC step. This gives us a homography for each image that lets us transform the image into the coordinate system for the final panorama.

Each image is transformed and composited onto our final panorama using pyramid blending, described below. At the end of this process we also match the image on the far right to the image on the far left by repeating the image on the left and looking at the height difference of where it appears on the left and where it appears on the right. We then crop the image to remove the seam and perform a linear transform in the y direction using the equation y’ = y + ax to make the image on the left and right line up. We perform one final crop of the top and bottom of the image to account for blank space produced in the linear transform.

# Extras

## Blending

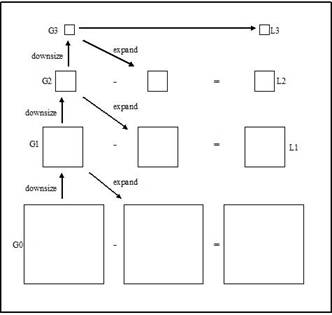
For the project we implemented Laplacian pyramid blending.

One advantage of Laplacian pyramid blending is that you blend at multiple frequencies. The more levels there are in the pyramid, the wider the range of frequencies that are blended. However, a significant drawback of Laplacian blending (or at least our implementation of it) is that it’s difficult to consistently recover the exact dimensions when expanding tiers. Our implementation, assumes an expanded tier will have 2\*n-1 by 2\*m-1 (where m and n are the original dimensions of the tier). This, of course, results in a loss of data (alternatively we could have assumed 2\*n by 2\*m, which would instead result in inflation of data and potentially degrade the quality of blending).

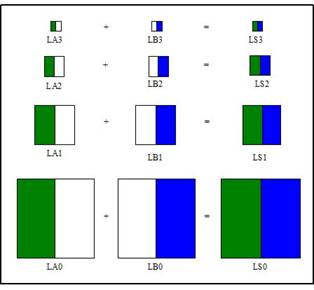
|  |  |  |
| --- | --- | --- |
| 1 level (feathering) | 5 levels | 7 levels |
|  |  |  |

5 levels yields much better blending than 1 level (the split between apple and orange is more pronounced), however, the loss of dimensions is noticeable. 7 levels, again, yields better blending than 5 levels but at the cost of more apple/orange missing.

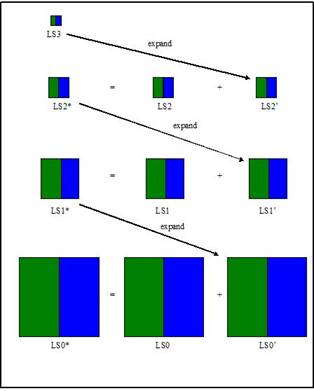
The first step in blending is to build Gaussian pyramids for the two images we’re blending. To build the Laplacian pyramid, traverse back down the Gaussian pyramid and expand the tier above (an image at a lower resolution) and expand it. We then subtract the expanded image from the image at the current tier and this approximately results in a frequency that we will later use for blending.



In the second step we apply a respective mask to each pyramid and alpha blend at each level. After that, we combine the separate Laplacian pyramids into one.



In the last step we expand and collapse tiers of the pyramid, which allows for multiple frequencies of blending to appear in the final resulting image.



# How to Run Program

Our program can be run from MATLAB by first loading in our source code and then executing the “main.m” file. There are several key parameters that need to be set to create a panorama with our program.

1. The location of the VL\_Feat library. Line 5 of “main.m” requires the path to the folder containing the VL\_Feat library so that it can run the setup when the program starts.
2. The camera parameters (f, k1, k2) are set on line 10-12 of “main.m”.
3. The path to the folder containing the images that you want to stitch together must be provided on line 13 of “main.m”.
   1. The images in this folder should be named so that when put into alphabetical order each image is a rotation to the right from the image that comes before it alphabetically. That is, if you define your access of rotation pointing up from the ground in the positive direction, then the difference between image “01.jpg” and “02.jpg” should be a negative rotation about this axis.

### Optional

1. The number of iterations that RANSAC is run for as well as how many random pairs are used to compute each hypothesis homography can be adjusted by changing the parameters on lines 4 and 5 of “ransac.m”.
2. Laplacian blending can be altered as well. Within “stitchImages.m” on line X you can modify the number of tiers the Laplacian pyramid uses for blending. The default is set to 5.

# Citations

M. Brown, D. G. Lowe, “Recognizing Panoramas,” ICCV 2003.

Burt and Adelson, "The Laplacian Pyramid as a Compact Image Code," IEEE Transactions on Communications, vol. COM-31, no. 4, April 1983, pp. 532-540.