Panorama Report

Michael Cruz
mi484725@ucf.edu
University of Central Florida
Orland, Florida

1 Introduction

The aim of this project is to create panoramas from multi-image datasets. The images from all datasets were captured by rotating a camera around a stationary origin (i.e the camera center or the camera itself) and taking photos with slightly overlapping sections. To stitch these images together creating a panoramic, the infinite homography is calculated between each image and either the previous or next adjacent image in the set, depending on the image's location in the sequence with respect to the center image. These homographies are saved and used to transform each image, and subsequently stitch the transformed images together. The exact methodology and algorithms used to perform these tasks will be discussed in detail in the following sections.

2 Dataset

There were two datasets provided to us to use for the project in folders titled mov2, and mov3. Additionally two other datasets were collected to use in the experiment. The first dataset was captured with a Eufy home security camera with rotation capabilities. Ten photos of my backyard are used to create a panorama and these photos are located in the backyard folder. The images required pre-processing in the from of cropping to remove the timestamps and Eufy logo from the images. The last dataset of images was acquired using the street view layer of google maps. Ten images were collected of the Meiji Jingu temple in Japan with the Windows Snippet tool. By accessing the street view and rotating the camera, acceptable images with no camera translation were collected.

3 Methods

To create a panorama several steps are required. First, five photos from each dataset were selected at even intervals to be used in the panorama. I selected only five images to streamline the process of making panoramas with different datasets. To use a different image set, five images from the set need to be chosen and the paths of those images must be changed in the code where the comments indicate the changes. Subsequently, the selected photos go through each of these steps:

(1) Feature detection
(2) Feature matching
(3) Homography estimation
(4) Image warping
(5) Stitching warped images

Each step will be discussed in detail in the next sections.

3.1 Feature Detection

For each of the selected images, 200 SUFR features are extracted, including the feature vectors and the corresponding coordinates of the strongest features in the image. I chose to extract 200 features because through experimentation, I discovered that extracting 100 features would sometimes lead to a low number of feature matches with adjacent images. However, when 200 features were selected, there were always at least 80 feature matches between all adjacent images. The feature vectors and coordinate matrices are returned from the function getFeatures(image) to be used in the feature matching step.

3.2 Feature Matching

The function getMatchingPoints() accepts a matrix of feature vectors and a matrix of corresponding coordinate locations from two adjacent images. Using these matrices, the function uses the matchFeatures() function to find matching features between the images and returns the indices of the coordinates of each match. Using those indices, the coordinates of the matches are extracted from the coordinate location matrix of each image. Subsequently, a 1 is concatenated to each row of the matrix to make the coordinates homogeneous for future calculations. The two sets of homogeneous coordinates are returned to be used to estimate the infinite homography between the two images.

3.3 Homography Estimation

The infinite homography is the homography that solves the equation x'= Hx for all x and x' in two images. Therefore, H is a 3x3 homography that maps each point x to a point x' in the images. However, H is unknown. To find the optimal H value, the equation Ah = 0 is solved which is a homogeneous systems of equations where A is a matrix containing point correspondences between the images and h is H represented as a 9x1 column vector. To construct the matrix A, matching points between the two images (found in the feature matching step) must go through the following steps.

3.3.1 Modified RANSAC. The sets of matching points for each image contains both inliers, i.e points that are geometrically close to each other given a distance threshold, and outliers, i.e points that correspond to each other but are geometrically far away in distance. To ensure that H is the best fit homography for the two images, a modified version of the RANSAC algorithm is performed on the matching features by the estimateHomography() function. In the modified version, the the inliers are found among the matching feature pairs, which are often a smaller number of points than the full set of feature pairs. This subset of inliers is extracted from the matching pairs and the algorithm finds the subset of inliers, within the previous subset of inliers. This iterative process continues until either the number of candidate matching points is equal to 4, or the number of inliers during an iteration is equal to the number of inliers of the previous iteration, meaning there were no outliers found. The final set of matching feature points are used to construct the point correspondence matrix A.

3.3.2 Constructing the Point Correspondence Matrix. For each point obtained inlier obtained in the previous, a matrix is constructed representing the cross produce of corresponding points from each image. Each point x = [x, y, w] and x' = [x', y', w'] is computed into the following matrix:

$0^T, -w'_i x_i^T, y'_i x_i^T$

$w'_i x_i^T, 0^T, -x'_i x_i^T$

$-y'_i x_i^T, x'_i x_i^T, 0^T$

Then, each point correspondence is concatenated together by stacking the point correspondences on each other creating a 3nx9 matrix of correspondences. Through experimentation I found that keeping all three equations for each point matrix sometimes leads to blurry transformed images, like in Figure 1. Therefor, I decided to emit the last equation for each point correspondence which led to a 2nx9 matrix A, where n represents the number of points. The matrix A of all point correspondences is returned from the function constructCorrespondenceMatrix() and will be used to calculate the infinite homography between the adjacent photos.

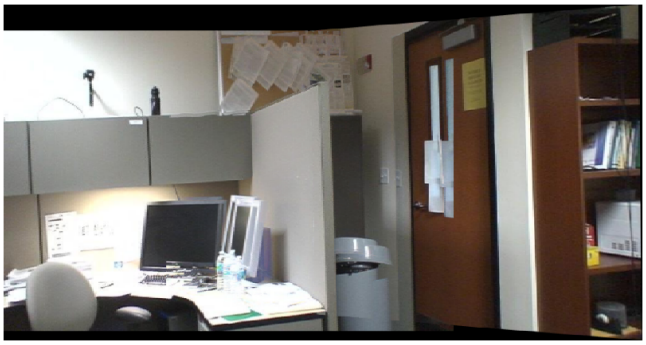

Figure 1: An image depicting as image transformed by a homography that used all three linear equations for each point in the point correspondence matrix

3.3.3 Constructing the Homography. The function calculateHomography() accepts the previously construct A matrix representing the point correspondences of two adjacent images and solves the equation (1) Ah = 0. To solve the homogeneous equation, the null space of matrix A must be found. This is accomplished by performing singular value decomposition (SVD) on the A matrix. The A matrix is not full-rank, and should be rank deficient by one. The result of SVD is three matrices: U, D, and V. The last column of the V matrix is a 9x1 column vector and represents the solution to h in equation (1). This column vector is reconstructed into a 3x3 homography and is returned to be used to transform the corresponding images.

3.4 Image Warping

The previous process of estimating the homography between adjacent images is done for each of the five photos. However, the process is completed in a specific way as to ensure that the third image in the set of selected images is the center image in the panorama. To do this, the following homographies were estimated, where x -> y means that x is mapped to y:

(1) Homography 1: image 1 -> image 2
(2) Homography 2: image 2 -> image 3
(3) Homography 3: Identity matrix
(4) Homography 4: image 4 -> image 3
(5) Homography 5: image 5 -> image 4

Additionally, the following steps were taken for images not adjacent to the center image:

(1) Homography 1 = Homography 1 * Homography 2
(2) Homography 5 = Homography 5 * Homography 4

By multiplying these homographies together, the images not adjacent to the center image are adjusted to account for the fact that they are being mapped to a transformed image, as opposed to images adjacent to the center image which are being mapped to the untransformed center image itself. The 5 homographies are concatenated into a 3x3x5 matrix that is used to transform each of the images. After the homographies are fit, they are stored as a list of 2D projective geometric transformation via the projtform2d() function. Each image in the sequence is warped using their respective transformations and the warped images are stitched together to create the panorama.

3.5 Stitching Warped Images

This section of the code was created with the help of another code source on creating panoramas published on MathWorks. I have included the source code along with the resulting images. This is the only part of the assignment that was not created exclusively by me. However, I only used the code as a skeleton for my intended purposes and altered it to warp and stitch the images together. First, the x and y limits of each transformed image are calculated and stored in two lists using the outputLimits() function. Subsequently, the maximum and minimum x and y values are obtained from the set of transformed image coordinate limits. Those maximum and minimum values are used to calculate the height and width of the panorama. Then the base panorama that all the transformed images are stitched onto is initialized to a zero matrix with the previously calculated height and width as size parameters. Along with the panorama, a base output image is created using the imref2d() function so that the transformed images will be stitched side by side, retaining the 3D world coordinate system from which the images were taken. For each image in the set, the image is transformed using the homographies previously estimated, then a mask is created that allows adjacent images to be blended together in the panorama instead of being placed side by side. Finally using both the warped images and the masks, each image is stitched onto the panorama canvas one-by-one using the imblend() function.

4 Results

The results of the experiment were successful in creating panorama images using multiple datasets of images. I created four panoramas from the datasets. The backyard panorama is the worst one most likely due to the quality of the images and possibly a slight tilt of the camera. The other panoramas were created seamlessly and the results are displayed at the end of the paper. There were some difficulties in obtaining the results, which I will discuss in the next section.

Figure 2: A subset of images from the mov2 dataset

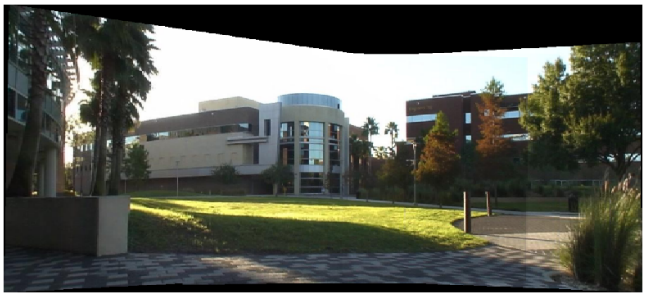

Figure 3: The resulting panorama of the mov2 dataset

Figure 4: A subset of images from the mov3 dataset

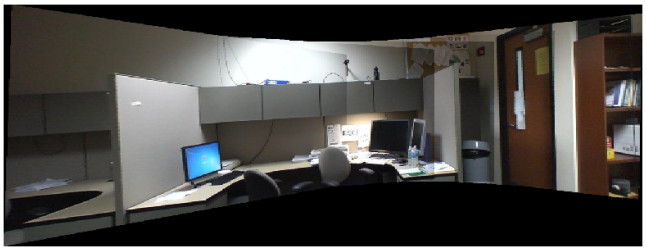

Figure 5: The resulting panorama of the mov3 dataset

Figure 6: All of the images from the backyard dataset starting at the top left, and progressing right then down

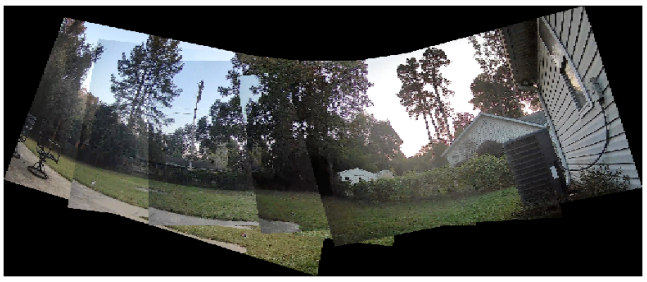

Figure 7: The resulting panorama of the backyard dataset

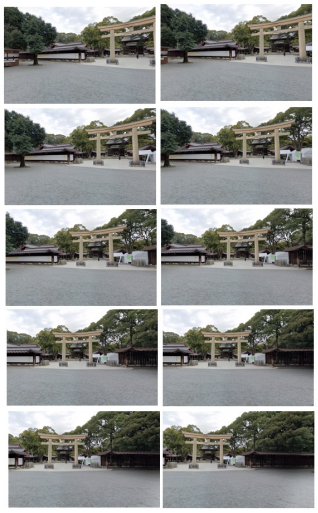

Figure 8: All of the images from the Meiji Jinga dataset starting at the top left, and progressing right then down

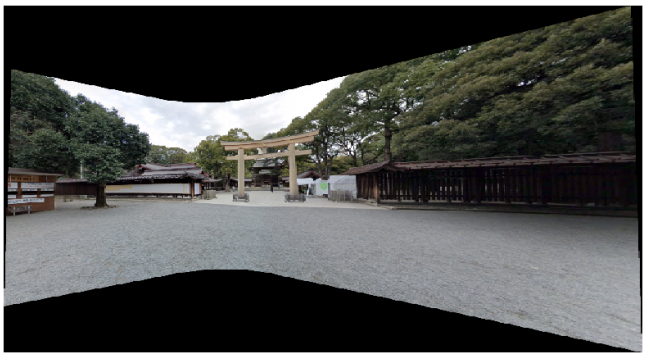

Figure 9: The resulting panorama of the Meiji Jinga dataset

4.1 Issues

There were some aspects that affected the quality of the resulting panorama. For instance, the distance between adjacent images played a role in the quality of the final image. Selected images that were rotated too far apart made lines appear in the panorama where the images were stitched together. I found that using images closer together made not only each individual transformed image look better, but it gave the panorama the ideal hyperbolic shape. Another issue I had was creating the homographies at infinity. My first attempts included taking all matching point correspondences and using 100 elements from the set to create the point correspondence matrix. However, this often led to transformed images that seemingly went to infinity. I also tried taking only 4 elements from the set of matching points, which fixed some of the transformed images but negatively affected others. Finally, by doing a modified version of the RANSAC algorithm, I obtained images that truly looked projectively transformed. One of the last issues I ran into which I briefly discussed was the point correspondence matrix itself. Whenever I kept all three linear equations for each point, some transformed images would come out extremely blurry, or somewhat skewed, but by keeping only the first two linearly independent equations, the problem was mitigated.