Northwestern | McCORMICK SCHOOL OF ENGINEERING

Department of Electrical Engineering and Computer Science
EECS 332 – Introduction to Computer Vision
Fall 2017

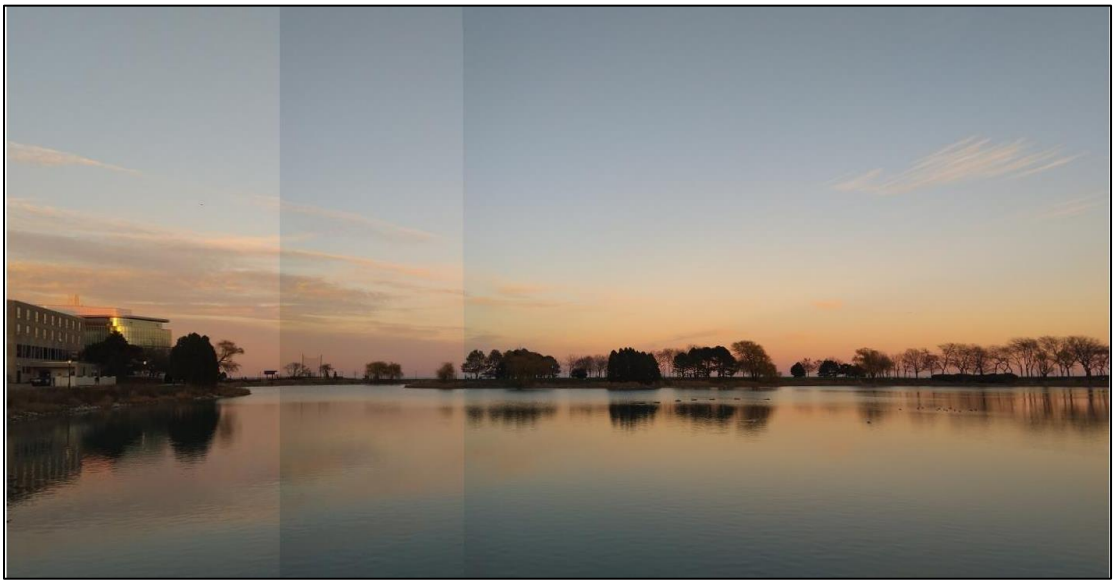

Image Mosaic
Final Report

Name: Karan Shah
Submitted to: Professor Wu
Report Due: 12/6/2017

Table of Contents

Executive Summary

The field of Computer Vision is a field with quite lot possible applications. In general, it provides the technology and methods used to compute imaging-based automatic inspection and analysis. Such applications can include automatic inspection, process control or robot guidance. Some of the applications that are used using computer vision are detection of objects, tag and categorize images, detect and recognize faces or analyze images for computer aided diagnosis in hospitals.

In this course, there was a huge possibility to learn new methods and get a good idea of the computer vision area. During the 1st week, an introduction into binary image processing was given. Black and white images where used to detect segment with the connected-component labeling. The next step were the morphological operators. The working of Erosion, Dilation, Opening and Closing operators were covered. Given those elements, it is possible to detect the boundary of a segment. Sometimes the image is too dark and not a lot of details can be detected. For such situation, the histogram equalization can be used. Given the concept of a histogram for all grayscales, by summing and normalizing it, the histogram can be equalized. As result, more details can be detected.

After binary image processing, colored image processing was introduced. For understanding working of color detection, a fundamental introduction into color spaces (RGB, YMC, HSI) and color segmentation was covered in the course. In addition to color, also different kind of edge detectors, like the Kenny edge detector, were implemented in machine problems. To not only detect edges, but also lines and circles, Hough Transform was implemented.

In addition to basic image processing algorithms, more advanced topics were introduced as well. An emphasis was set to face and motion detection. Different ideas were covered, starting with skin and couture detection over the Viola-Jones-Face-detection and ending with the Bayesian Network for estimation of face motion.

Introduction

Everybody knows this situation: you see a beautiful panorama and want to take a picture of that. Unfortunately, the whole panorama does not fit in one image. Or maybe you want to take a detailed photo of a room. In such cases it would be great to make multiple images and stitch them all together. In “The Image Mosaic Project”, Francesco Spadafora and me designed a project which enables the user to form a panorama from different images. We wrote a MATLAB-function that takes an arbitrary number of images as input and stitches them to a panorama.

The report contains the algorithm and the general overview taken to tackle the problem at hand. The results and future work is explained. The executive summary describes the general overview about the field of computer vision along with the personal view regarding the course taken at Northwestern University.

Design Overview

This report details the design and implementation of Image Mosaic Project for EECS 332: Introduction to Computer Vision at Northwestern University. For capturing more details, one picture is not enough, and the user needs to take multiple images of different segments. For seeing all the details in one image, the different images must be snitched together by an additional software that creates a panorama. Today operating systems for smartphones have a built-in function for that known as a Panorama. Different software's are available for creating a panorama from multiple images in PC. But, while looking at these software's we found MATLAB has no such function. Thus, our goal was to write a code that performs this function in MATLAB. In the following sections, we describe the algorithm along with the results obtained while running the algorithm.

Algorithm

Figure 1 shows the general overview of the algorithm. The main steps of the algorithm are loading of images, feature extraction, finding the size of panorama image, stitching the different images to form a panorama and then cropping the image to remove the excess part of the image. Let us see each step-in detail.

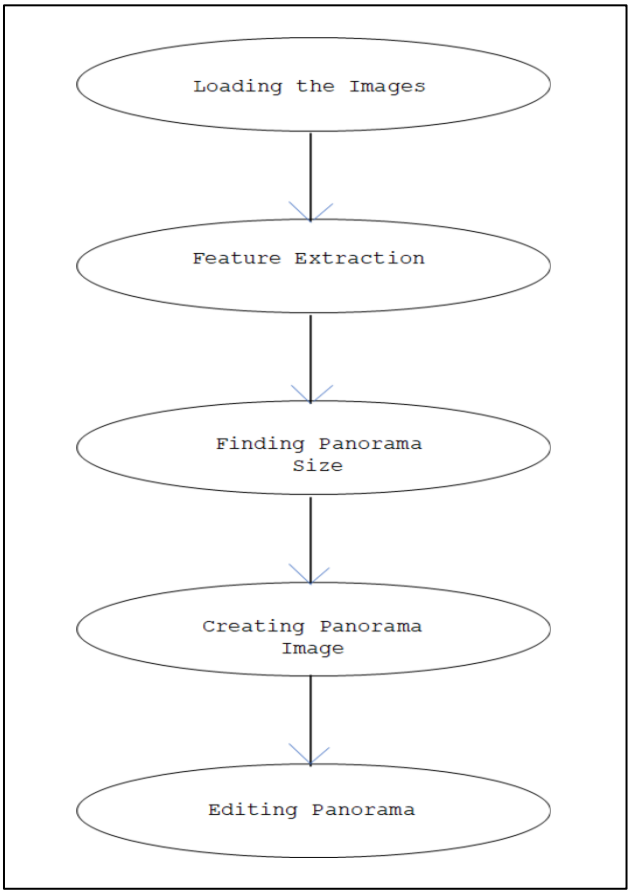

Figure 1: Algorithm

Loading of Image

The function *load_image* deals with loading of image from a specified folder and reading them. The input for this function is the file path where all the images are stored. The output of this function is the number of images selected and an array of the images selected.

```
function [file_names, image_num] = load_image(fileFolder)
%Loading of Images
```

Figure 2: Load Image Function

Feature Extraction

The function *feature_image* is used for detection of features between two different set of images. We use inbuilt MATLAB function for feature extraction and projective transform.

```
function [image_size, proj_transform] = feature_image(I, file_names, image_num)
gray_image = rgb2gray(I);
feature_points = detectSURFFeatures(gray_image);
[features, points] = extractFeatures(gray_image, feature_points);
```

Figure 3: Feature Image Function

Two neighboring images are read and converted to grayscale images. The in-built function *detectSURFFeatures*, detects and extracts features. For every image, an index is created. The function *estimateGemoetricTransform* creates a transformation matrix. This is used to reshape the images such that they fit next to each other.

Size of Panorama

The idea of this sub function is to create a panorama picture for the output. From the parameters obtained from *feature_image*, the minimal and maximal sizes in x and y direction are detected. With the x and y values detected, an empty panorama full of zeros is initialized.

```
function [panorama, x_min, x_max, y_min, y_max, width, height] = panorama_size(
                        (proj_transform, image_size, I)
```

Figure 4: Size of Panorama Function

Creation of Panorama

This sub-function is creating stitched image. The parameters are obtained from *feature_image*, and *panorama_size* function. The built-in functions *Vision.alphabender* and *imwrap* help in the creation of the panorama. The different images are stitched and blended. The output image is a deformed panorama image.

```
function [panorama] = panorama_creation(panorama, x_min, x_max, y_min, y_max,
            width, height, image_num, file_names, proj_transform)
```

Figure 5: Creation of Panorama Function

Cropping of Panorama

This is the last function in the algorithm. This function is used to crop the image and display a beautiful panorama. This function works in a greedy algorithm way. A rectangular area is selected by giving the input as the number of rows and columns to be corrected. This rectangular area is then placed on the deformed image and a cutout is obtained.

```
function [panorama_complete] = panorama_edit(panorama,
          row_correction, column_correction)
```

Figure 6: Cropping of Panorama Function

Result

We implemented our algorithm for the Image Mosaic system. We tried our code on 5 set of images and each time we had to change the correction parameters to obtain a better panorama (Panorama example 1). But in some cases, we were able to see the difference between two images due to difference in contrast. Below are the sample images and their output. For each set of images, we have shown the input montage, feature matching and the output.

Panorama Example 1

For the 1st example, we took images of bonfire near the lake at Northwestern University. Figure 7 shows the montage of input images. Figure 8 shows the features matched between the input images. In the next step, the deformed image is formed and after that the final cropped image is shown in figure 10. In this image, due to better contrast, we are not able to see the edges between the images when they are stitched together.

Figure 7: Input Montage

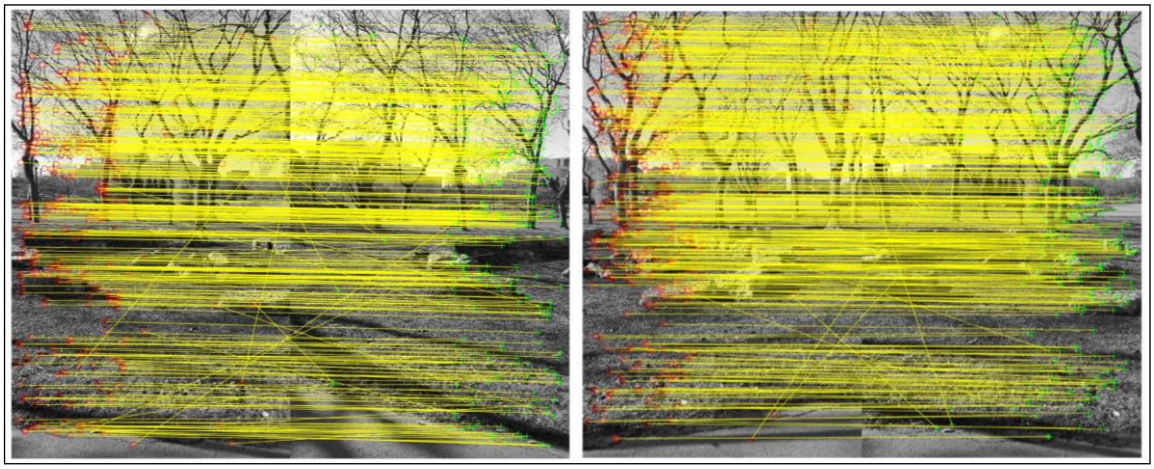

Figure 8: Feature Matching

Figure 9: Panorama – deformed and cropped

Panorama Example 2

Figure 10: Input Montage

Figure 11: Feature Matching

Figure 12: Panorama – deformed and cropped

Panorama Example 3

Figure 13: Input Montage

Figure 14: Feature Matching

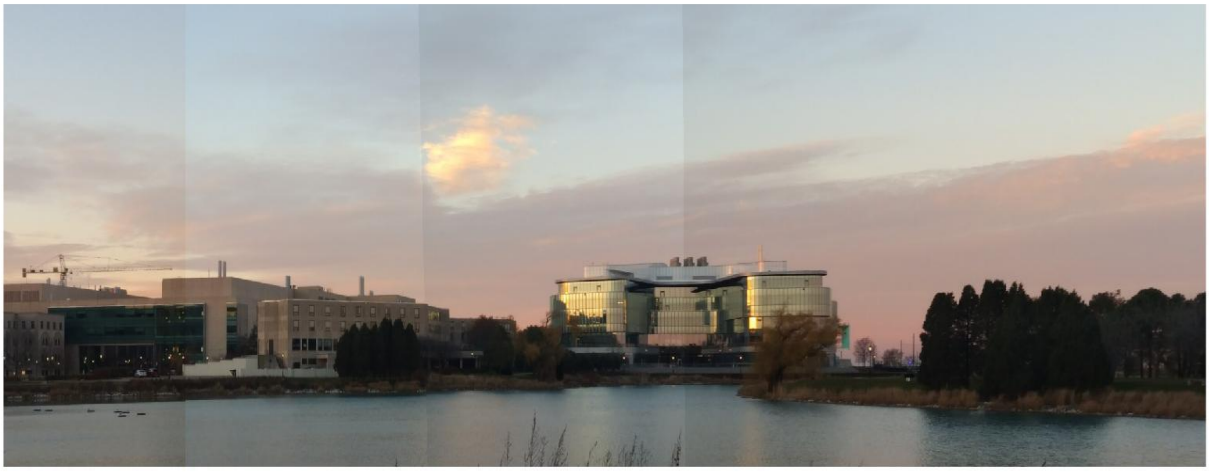

Figure 15: Output Image

Panorama Example 4

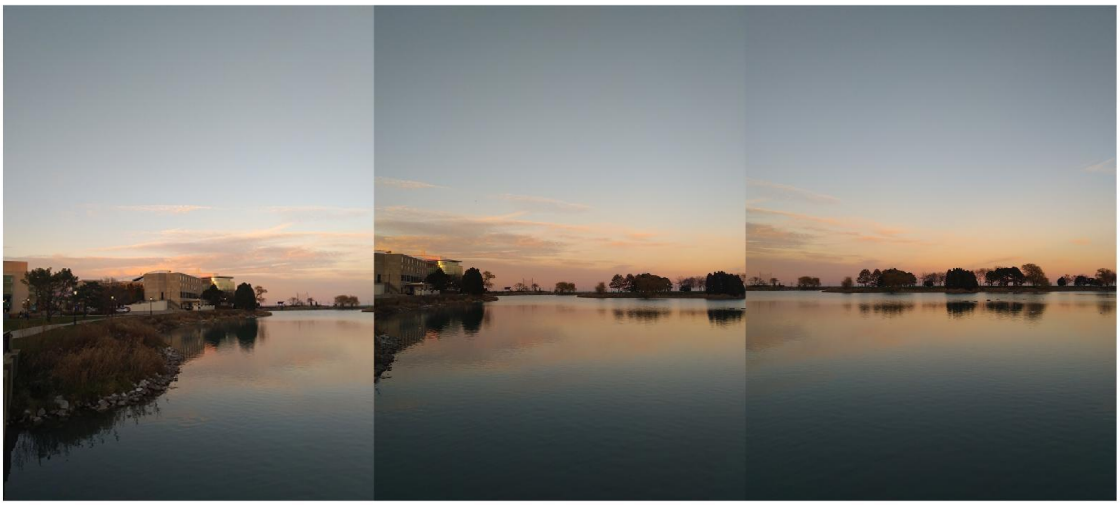

Figure 16: Input Image

Figure 17: Feature Matching

Figure 18: Output Image

Future Steps

The biggest next step for this project is refinement of our algorithm. Although we were able to create a functional prototype, there are plenty of areas that could be made more efficient or where more features could be added. Much of these issues stem from the rapid manner in which we developed the system, but could be avoided with more careful planning. For example, in some cases we are able to see the edges when two different images overlap and are stitched together. The feature matching function was not accurate to our liking due to which, the edges could be seen in deformed image. Small improvements such as an efficient algorithm and accurate functions would provide a cleaner user experience.

Another major future step would be the addition of a contrast enhancing function. When the images have different contrast rate, we are able to see the edges even in the final images. As of right now, if the images are of different contrast, the edges will be seen. This can be changed by smoothing the borders of the image and minimizing the differences in contrast and illumination and equalizing the colors.

One major function we did not have time to develop but would like to see added is automated image interpretation and resizing. For the image interpretation, we want the system to interpret which image should go on the left most side and so on. This can be achieved using machine learning and using in-built MATLAB functions. The second part is the image resize code. MATLAB has constraints on the size of the image it can handle. The resize function should be designed in such a way that the images are scaled down while maintaining the original quality of image and passed through the algorithm. This way, we can bypass the size constraint and obtain a good quality output image.

The last thing we would like to work on is the maximal cut-out of the panorama also known as cropping of panorama. As of now, we used trial and error method, to find the optimal cut-out of the panorama by manually setting the correction parameters. We tried designing an automated function which crops out the excessive deformed part of the image and displays the panorama. We couldn’t work more on this as we were short on time but would surely implement in the future.

Conclusion

Our final code has some limitations compared to the output we originally set out to get. However, we were very satisfied with the output that we did manage to complete. We were able to demonstrate the application with different set of input images having different contrast and details. With more development time, it should not be very difficult to achieve some of the functions mentioned in future work to the existing code.

References

1. http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/imagemosaic.html.
2. R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models, SIGGRAPH 1997, pp251-258
3. Slides during Class
4. https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html
5. http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/imagemosaic.html

Course Feedback

I really liked this course and was glad that I took it. I had some basic knowledge about image processing but after taking this course my interest has gone up. I know now the mathematical concepts behind the important concepts in image processing and video processing. This class was a satisfactory introduction to this topic.

The professor helped the students and encouraged the students to try different things in this field. The course was more based on practical's and assignments than theory which I think is was very good. The topics for final project were also fun and each of them has their plus point when being implemented. I would like if the professor explains more about automated learning i.e. how can we integrate computer vision and machine learning.

Personally, I liked the organization of the course. From beginning the syllabus of this course was clear. Some of the machine problems were easy but the in general it challenged me intellectually. Last but not the least, I would like if there was a bigger classroom as sometimes, it became overcrowded. Over all, I enjoyed the course and would also recommend it to other people.

Appendix A: Code

This appendix outlines the code used in the project. We used MATLAB 2017 for execution of our code. For full code please visit: https://github.com/kks3851/ImageMosaic

Main Program

```
clc;
close all;
clear all;

file_folder = 'C:\Users\Karan Shah\Desktop\Intro to Comp
Vision\Project\Final\code\'; % file path
row_correction = 6; % parameters for cropping
column_correction = 15;

[file_names, image_num] = load_image(file_folder); % loading of
images
[I, map] = imread(file_names{1,1});

[image_size, proj_transform] = feature_image(I, file_names,
image_num); % feature extraction
[panorama, x_min, x_max, y_min, y_max, width, height] =
panorama_size(proj_transform, image_size, I); % finding the size
of panorama
[panorama] = panorama_creation(panorama, x_min, x_max, y_min,
y_max, width, height, image_num, file_names, proj_transform); %
creation of panorama
[panorama_complete] = panorama_edit(panorama, row_correction,
column_correction); % cropping of deformed part
figure;
imshow(panorama_complete) % final output
```

The first 3 lines of the main program are used for closing extra windows and clearing the workspace. The *file_folder* defines the path of the images. The variables *row_correction* and *column_correction* is used for cropping of deformed panorama. The images are loaded and read using *load_image* function. Then features are extracted using *feature_image* and given to *panorama_size* for finding the size of the panorama. After the size is found, *panorama_creation* is called, and a deformed panorama is found. To correct the deformed image, *panorama_edit* is called,S and we get a proper panorama image.