# **ELEC 474 Machine Vision Project**

## **AutoStitch: Matching and Merging an Unordered collection of Images**

Queen’s University

Department of Electrical and Computer Engineering

2021-12-1

Zhenyue Yu

20093332

### List of Code Modules:

def FindBestMatch(images, idx) original

def generateHomography(img1, img2) original

def blendingMask(height, width, b, sw, left\_biased=True) non original

def panoramaBlending(dr, src, wd, side) non original

both of blendingMask and panoramaBlending are from (https://github.com/ndvinh98/Panorama/blob/master/stitch.py)

def warpImages(img1, img2) original

def sideDetermine(pts, pts1\_, wd, t) original

def blendAndCrop(img1, img2) original

def splitImageList(list\_images) original

def mergeTwoLists(list\_images) original

def multiStitching(list\_images) original

I declare that this submission is entirely my own work, in my own words, and that all resources used in researching it are fully acknowledged and all quotations properly identified.

### Step1: Match Features

The first step is to load all the images from one of the image sets provided (i.e., office2). I read multiple images on the same folder using a for loop with imread function. The images are stored into an image list(images1). The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell. In order to find a set of images that can overlap with each other, the number of key points and matches should be determined. SIFT feature descriptor is used to extract features and generate matches. A ratio of 0.65 is set to filter the matches. The distance ratio between the two nearest matches of a considered key point is computed and it is a good match when this value is below a threshold. A matrix “good” is used to store all matches that are satisfied the requirements.

The function FindBestMatch(images, idx) I designed will take image list and index of the list as two inputs and output a list of image that have similar features.

To start with, the target image is selected depending on the index of image list and its key points and descriptors are obtained from class::SIFT method detectAndCompute. The next step is to loop through all the other images in the list and compare and calculate the number of good matches. A threshold of 4500 is set to determine the images that have high overlaps with the target image. In other words, images with the number of good matches more than 4500 will be stored into an image list and drop other images otherwise. Then the last step is to test the method and randomly choose several target images (I choose the first image as my target image). The result is shown below:

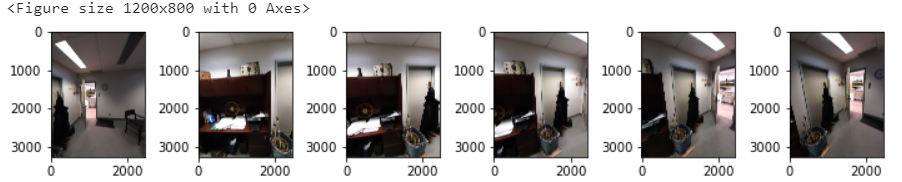


Figure 1: The result of target image (the first one) and its 5 good match images (threshold is 4500).

### Step2: Estimate Transformation

The function I designed generateHomography(img1, img2) will take two images as inputs and calculate the transformation between them. I applied the same approaches that I have finished in lab4: given two input images, detect and compute their key points and descriptors; use cv2.FlannBasedMatcher() to find matches with the descriptors; apply ratio filter and format points for use in transformation functions; apply cv2.findHomography() to calculate the homography from the reference image.

### Step3: Merge Images

The function I designed warpImages(img1, img2) will calculate the result image size and then do a translation. The geometric image transformation method cv2.warpPerspective is applied to transform the source image. I split the result image list into two sub lists by splitImageList(list\_images), left and right, to separately calculate size and merge them together. The other two functions panoramaBlending(dr, src, wd, side) and blendingMask(h, w, b, sw, left\_biased=True) are used to blend the result image.

### Results

A screenshot of a video game

Description automatically generated

Figure : The merge result (choosing the first image as the target image)

The program usually takes five minutes to execute one image set and obtain the result. It cannot execute all the 29 images from the image set due to the lack of key points between each image. Basically, it can find several images (four to five) that can perfect overlap with the target image. Then, up to six images can be merged together. However, the threshold needs to be changed for each target image and the user has to change the threshold manually each time the target image is changed. The program needs to be improved so that it can merge more images at one time and the threshold should be determined automatically.

A collage of a room

Description automatically generated with low confidence

Figure : The result of target image (the 15th one) and its 5 good match images (threshold is 5000).

Graphical user interface

Description automatically generated with medium confidence

Figure : Figure 2: The merge result (choosing the 15th image as the target image)