***2812ICT: PHOTOGRAPH STITCHING FOR THE CREATION OF PANORAMAS***

By Sebastian Perry S5132483

# Abstract

This document will contain the process which was followed in order to attempt the creation of python software which is able to fulfill the expectations laid out in the project proposal, such that a loose panorama could be made out of a selection of pre-existing images.

Throughout the project the expected output shown halfway through the Recognising Panoramas research paper (by M.Brown, and D.G Lowe) was used as an end goal of achievement.

David Lowe (Google Scholar, n.d.) and Matthew Brown(Google Scholar, n.d.) respectively



# Introduction

Today people the world over practice the tradition of preserving memories through the use of photography, for subjects on a scale too large panorama photographs have proven a popular solution with many cameras and phones offering this as just another option which can be set and used.

These Methods however require the user to be in the location, unharried by chaotic outside influences, with a steady hand that can slowly pan across the environment; if these requirements are not met the results can be horrific.



S.Perry demonstrating the eldritch horror of an incorrectly taken panorama

What if instead someone could take several separate images (a much easier task), or not even be there at all maybe just find multiple images of the same location and stitch them together to create a panorama after the fact; the possibility just stated is what will be the focus of this paper.

(E.G. the creation of a body of code which can stitch multiple images into a somewhat finished panorama)

# Previous Work

When researching approaches to the goal, a research paper titled ‘Recognising Panoramas’ by David Lowe and Matthew Brown was discovered. After reading the report it was decided that the report and its methods would be an excellent fit for the goals set out at the beginning of this project. In addition to the report various tutorials on the use of OpenCV encountered, as well as personal work produced during study of the Griffith 2812ICT course were useful in providing not only examples of similar methods but a small body of code which could be repurposed for the project allowing for a reduction in the amount of time required to achieve the desired results.

# Technical Approach

The following will be an account of the technical approaches which the produced body of code uses (in order) to achieve the goals set by this report:

1. **Image management and basic structure** – At the beginning of execution the program will load a set of images pre-arranged images into an array. It will first pop the starting image of the array before it through the remainder of the array, popping off the next picture, attempting to stitch it to the base, and making the result the new base until there are no images left in the array. The first step of the loop is calling a function which starts the process off by gathering SIFT points of the current base and the next image.
2. **SIFT algorithm** – also known as ‘scale-invariant feature transformation’, SIFT is an algorithm used by the program to identify key features in a given image outputting descriptors which describe them is such a way that they are invariant to translation, scaling, and rotation. Now that these features have been generated between the two images, it must be ascertained if some of these features are present in both images.
3. **KNN feature matching and ratio test** – is the feature matching of choice for this project’s implementation, in our case we run it with a k of 2 meaning it returns two best matches for every given feature. After receiving these matches, they are then run through something known as a ratio test, in which each pair of features found by the KNN tree are tested to ensure that the distance between them is within a certain arbitrary ratio (which for this implementation has been chosen to be 0.7), if not they are removed from the set before sending it to outlier removal.
4. **RANSAC outlier removal through subset selection** – the RANSAC, when given a set of points removes outliers by selecting a subset of inliers and returning them. Its use is necessary as an image matching of this nature could have quite a few outliers present in the matches consisting of false matches made between similar looking natural scenery and other such sources. Because of its nature it will often favour points with a minimum spread along for it to single out and ignore the more spread outliers. Once this subset has been found it will be returned and judged if it is an appropriate match to the base in image by the count of matches in the set, if the count is below 10 the image will be rejected as it clearly can’t be an image of the same subject. If the image has 10 or more points the image will be prepared for stitching to the base image.
5. **Homograph estimation and perspective translation** – the discovered points are then sent into a function which is able to estimate what changes need to be made to the image in order to change it scale, and orientation to match that of the base image, these required changes get returned and fed into a second function which performs the required transformations onto the image making it ready for stitching.
6. **Stitch the image** – a new image is made of the base image and the other image with their similar features overlapping. Due to the new image not being perfectly sized to account for the exact size of the new image, sometimes there may be a black border present, the next step fixes this issue.
7. **Remove the black border** – the image is converted to greyscale allowing for the use a few simple cv2 algorithms which find the bounds of the new image (being able to easily distinguish it from the pure black background) and crop it to the appropriate size.

(**PLEASE NOTE:** the code mentioned in the previous section is supplied in the zip file this report was found in, configured to run experiment 4 with all required images to do so. Furthermore, the power point presentation included contains images demonstrating the effects of the technical approaches steps individually)

# Experiments

The following will be a list of experiments which were conducted on the finished program to gauge its performance, each experiment will consist of the input, the output, and the result.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Experiment 1**: This experiment consists of two images taken from S.Perry’s backyard.

Input:



Output:



Result: the panorama was successfully created.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Experiment 2**: This experiment consists of three images taken from S.Perry’s backyard.

Input:

A picture containing outdoor, stone

Description automatically generatedA picture containing outdoor, ground, rock, stone

Description automatically generated

Output:



Result: the panorama was successfully created.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Experiment 3**: three images gathered from the internet of a location in Paris.

Input:

A picture containing sky, outdoor, house, road

Description automatically generatedA picture containing ground, outdoor, sky

Description automatically generated

Output:



Result: the panorama was successfully created.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Experiment 4: This experiment consists of 4 images taken from S.Perry’s backyard, and 1 random image found on the internet.

Input:

A picture containing outdoor, stone

Description automatically generatedA picture containing outdoor, ground, rock, stone

Description automatically generatedA picture containing rock, outdoor, building, stone

Description automatically generatedA sunflower with a blue background

Description automatically generated with low confidence

Output:



Result: the panorama was successfully created, it was able to reject the foreign image.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Experiment 5: This experiment consists of the 5 images from experiment 4 in a different order.

Input:

A picture containing outdoor, rock, building, stone

Description automatically generatedA picture containing outdoor, rock

Description automatically generatedA sunflower with a blue background

Description automatically generated with low confidenceA picture containing rock, outdoor, building, stone

Description automatically generatedA picture containing outdoor, building

Description automatically generated

Output:



Result: the program was unsuccessful at creating the panorama

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Conclusions

As shown in the experiments above, a body of code has been produced which can successfully take in a selection of photos and use them to output a panorama. The program was able to successfully match the images, enabling it to reject the unrelated image of the sunflower (images closer to the desired component images were not tested however it is very likely that if such an eventuality created issues, the match count requirement could simply be increased to adjust). The program even showed that it would allow for the creation of panorama’s using images sourced from the internet, opening up possibilities of being able to make panoramas of any location for which a collection of photos can be sourced from the internet. With that being said the experiments did show flaws In the program, some images would generate with visible seams, furthermore the successful generation of a panorama hinges entirely on the images being provided from left to right due to the fact that the code had no mechanism of detecting orientation with respect to the base and stitch accordingly.

# References

*Brown, M. and Lowe, D.G, 2003. Recognising Panoramas. Vancouver, Canada: Department of Computer Science, University of British Columbia.*

*[online] Available at:*

[*http://matthewalunbrown.com/papers/iccv2003.pdf*](http://matthewalunbrown.com/papers/iccv2003.pdf)

*[Accessed 1 September 2021].*

*Docs.opencv.org, n.d. OpenCV: Epipolar Geometry.*

*[online] Available at:*

[*https://docs.opencv.org/3.1.0/da/de9/tutorial\_py\_epipolar\_geometry.html*](https://docs.opencv.org/3.1.0/da/de9/tutorial_py_epipolar_geometry.html)

*[Accessed 21 September 2021].*

*Google Scholar, n.d. Profile Picture.*

*[image] [online] Available at:*

[*https://scholar.googleusercontent.com/citations?view\_op=view\_photo&user=8vs5HGYAAAAJ&citpid=2*](https://scholar.googleusercontent.com/citations?view_op=view_photo&user=8vs5HGYAAAAJ&citpid=2)

*[Accessed 9 October 2021].*

*Google Scholar, n.d. Profile Picture.*

*[image] [online] Available at:*

[*https://scholar.googleusercontent.com/citations?view\_op=view\_photo&user=WZfM0qsAAAAJ&citpid=4*](https://scholar.googleusercontent.com/citations?view_op=view_photo&user=WZfM0qsAAAAJ&citpid=4)

*[Accessed 9 October 2021].*