Orthophoto & Image Matching

Name: Nishit Patel
Email: n.t.patel@student.utwente.nl
Student No.: s2872129
Date: 23rd June 2022

Contents

1. INTRODUCTION **2**

OBJECTIVES AND STUDY AREA: 2

2. ORTHOPHOTO GENERATION **3**

3. ORTHOPHOTO REFLECTION **6**

RADIOMETRIC PROBLEMS: 9
GEOMETRIC PROBLEMS: 10
RMSE: 11

4. ORTHO-MOSAIC GENERATION **13**

IMAGE MATCHING ALGORITHM: 13
PYTHON IMPLEMENTATION & PARAMETERS: 13

5. ORTHO-MOSAIC REFLECTION **17**

Figures

FIGURE 1: STUDY AREA (UNIVERSITY OF TWENTE CAMPUS AREA, ENSCHEDE, NETHERLANDS) 2
FIGURE 2: DISTORTION IN BUILDING SHAPES IN THE ORTHOPHOTO GENERATED USING TERRAIN LAYERS (DTM) 3
FIGURE 3: COVERAGE AREA PROBLEM WITH DSM (ORANGE AREA IS DSM WHILE RED IS POINT CLOUD) 3
FIGURE 4: BLOCK STRIP CONSISTING OF THREE IMAGES AND POINT CLOUD DATA 4
FIGURE 5: PARAMETERS IN THE ORTHO RESAMPLE DIALOG BOX (ERDAS IMAGINE) 5
FIGURE 6: LEFT ORTHOPHOTO 6
FIGURE 7: MIDDLE ORTHOPHOTO 7
FIGURE 8: RIGHT ORTHOPHOTO 8
FIGURE 9: HOTSPOT ERRORS CAUSED DUE TO REFLECTION OF SUN IN THE GLASS 9
FIGURE 10: HOTSPOTS ON CARS DUE TO SOLAR REFLECTION 9
FIGURE 11: SQUASHED PART ON TOP OF THE BUILDING 10
FIGURE 12: DOUBLE MAPPED HOUSE 10
FIGURE 13: FLAT ROOF ON THE BUILDING 10
FIGURE 14: RMSE TABLE FOR MIDDLE AND RIGHT ORTHOPHOTOS 11
FIGURE 15: THIRD GCP POINT OUTSIDE THE COVERAGE AREA IN LEFT ORTHOPHOTO 12
FIGURE 16: 50 GOOD MATCHES IN THE LEFT-MIDDLE IMAGE PAIR (TOTAL GOOD MATCHES: 387) 14
FIGURE 17: 50 GOOD MATCHES IN THE MIDDLE-RIGHT IMAGE PAIR (TOTAL GOOD MATCHES: 1269) 16
FIGURE 18: 50 GOOD MATCHES BETWEEN THE LM ORTHOMOSAIC AND MR ORTHOMOSAIC (TOTAL GOOD MATCHES: 255) 16
FIGURE 19: FINAL ORTHO-MOSAIC OF THE STUDY AREA 17

1. Introduction

Objectives and Study Area:

The study area for this project was the University of Twente Campus in Enschede. The goal of this project was to create orthophotos for one of the image strips provided in the data. Because the orthophoto has been orthorectified and corrected for distortions, it can be used in a variety of GIS applications. As the study area was the university campus, potential applications of the generated orthophotos include aiding the planning and design initiatives at the university.

Further, using the generated orthophotos and image matching algorithms, an orthomosaic of the strip was generated. Orthomosaic is a collection of orthophotos stitched together to obtain a single image of a large area. This eases up the integration of orthophotos to a particular application domain when a user needs more than one orthophoto to cover the entire study area.

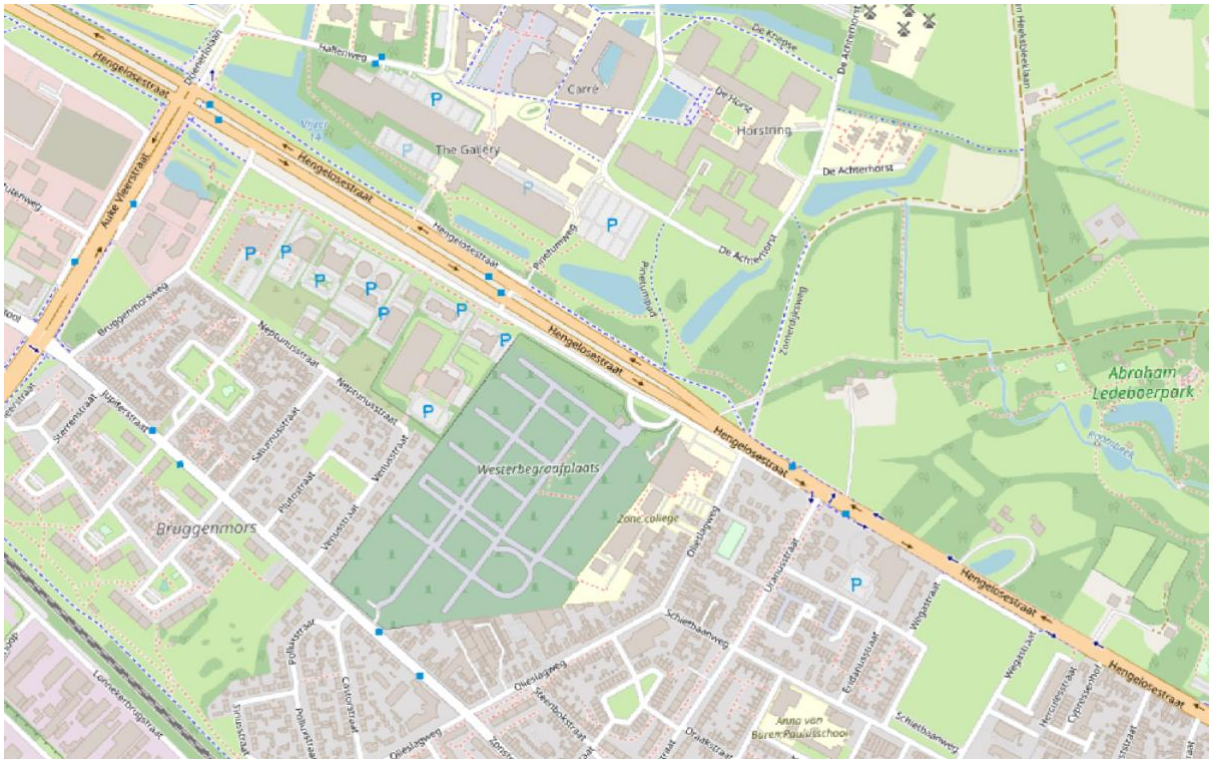

Figure 1: Study Area (University of Twente Campus Area, Enschede, Netherlands)

2. Orthophoto Generation

The Orthophoto generation process was carried out using the ERDAS Imagine. As an input, the second strip of images was selected. There were three images in the strip: 'IMG 18.jpg', 'IMG 22.jpg', and 'IMG 26.jpg'. The 'point cloud' data was chosen for the elevation input layer because the DTM was causing distortion in building shapes and the DSM coverage was much less than point cloud. Also, I had never worked with point cloud data before, thus, this was a good opportunity for me.

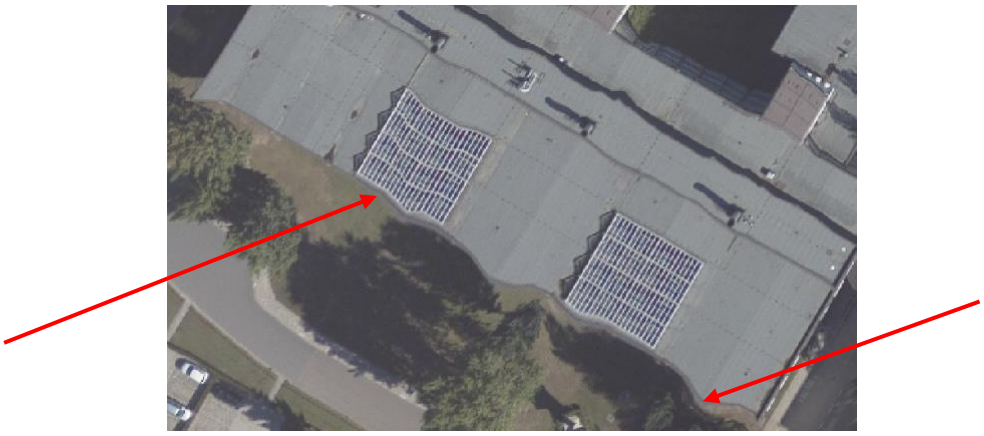

Figure 2: Distortion in building shapes in the orthophoto generated using Terrain Layers (DTM)

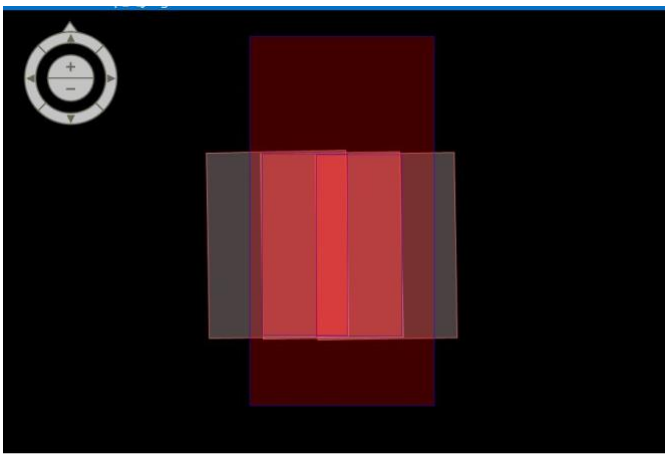

Figure 3: Coverage Area problem with DSM (orange area is DSM while Red is Point Cloud)

However, the provided point cloud data for the second strip didn't cover the images in the vertical direction. Thus, the point cloud data of the first strip and the second strip had to be merged using 'Cloud Compare'. The screenshot of the resulting block file is attached below.

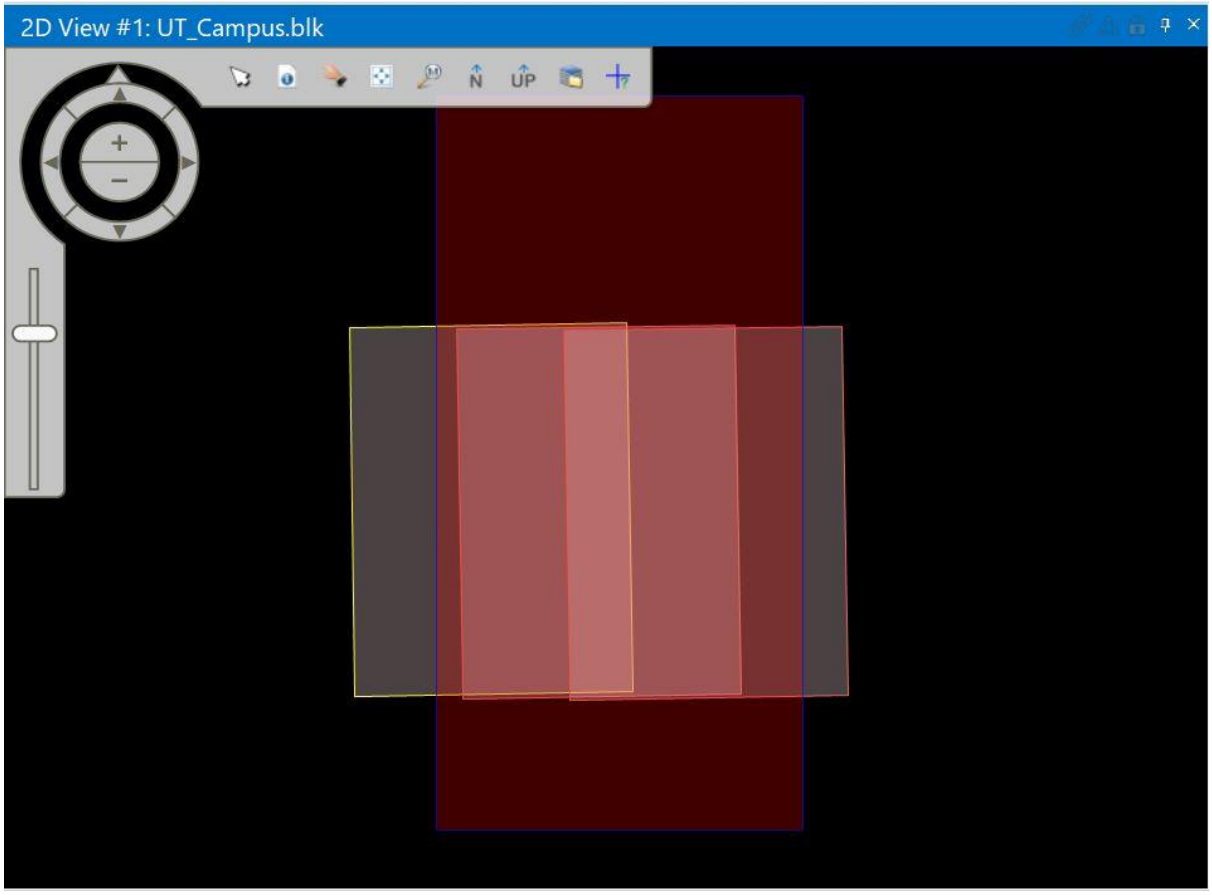

Figure 4: Block strip consisting of three images and point cloud data
(Note: The point cloud data is highlighted in red in the strip)

For the Ortho Resample process input parameters, the 'active area' was set to 100% to cover everything. The spatial resolution was set to 20 cm to avoid long computation times (both in ERDAS and later during image matching in Python), and it is also larger than the original image's GSD (~7 cm) (however it is more than factor of 1.5 which is assumed to be optimal in orthophoto generation).

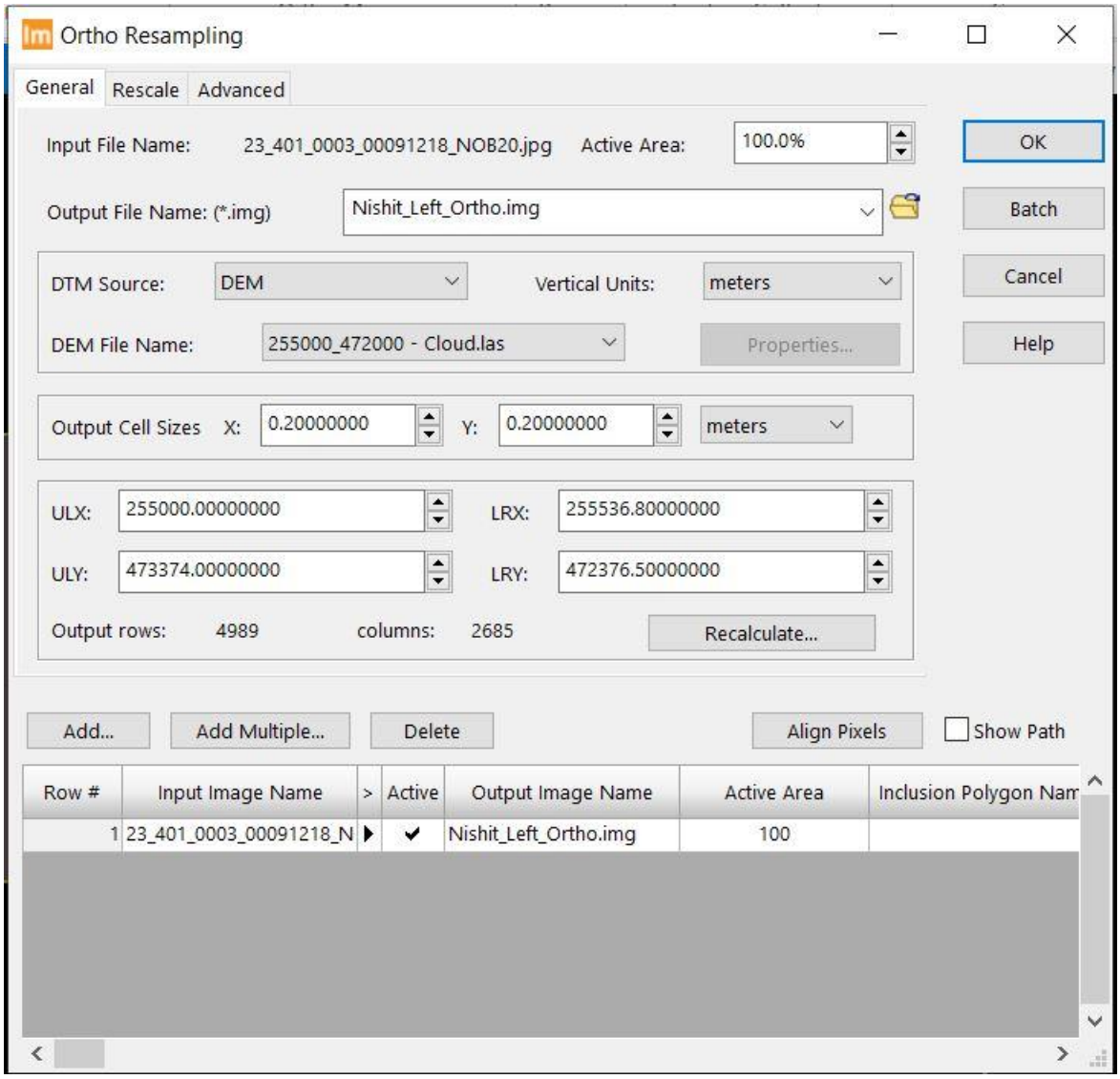

Figure 5: Parameters in the Ortho Resample dialog box (ERDAS Imagine)

Further, the extent parameters were also tuned to avoid black strips on the left side of left orthophoto and on the right side of the right orthophoto where the area in images is not covered by the point cloud.

3. Orthophoto Reflection

The three generated orthophotos:

Figure 6: Left Orthophoto

Figure 7: Middle Orthophoto

Figure 8: Right Orthophoto

As seen in the above orthophotos, there is a black strip on the borders of each orthophoto. This is due to the fact that the images in the second strip are a little titled and the point cloud layer

is perfectly rectangular. These black areas can be cropped to improve the overall appearance, but they were left in the output to demonstrate the effects of differences in image orientation and elevation layer when generating orthophotos.

Table 1: Radiometric Problems and suggested improvements

Radiometric Problems:
As seen in the below images, due to high reflecting material, there are some hotspot radiometric errors caused in some objects. Correcting these could be a complicated process but one approach could be to delineate the objects (if they are of interest in the application domain) and then apply filtering to remove pixels that have noise values.

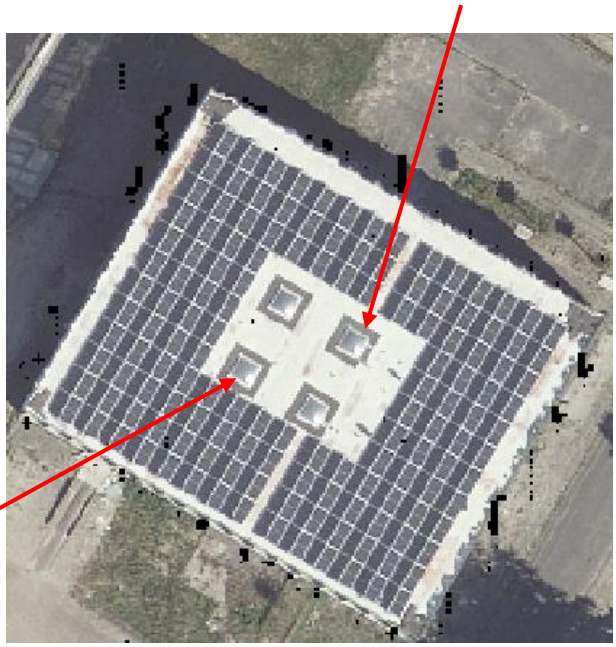

Figure 9: Hotspot errors caused due to reflection of sun in the glass

Figure 10: Hotspots on cars due to solar reflection

Table 2: Geometric Problems and suggested improvements

Geometric Problems:
As seen in fig. 11, the top floors of the building are squashed. This is because of the misalignment of the point cloud. Due to obscured areas, we can see double mapping happening in fig. 12. Further, in fig. 13, due to difference in viewpoints we can observe flatter roofs in some house buildings. The observed geometric errors can be attributed to either the flight parameters or the quality of the point cloud data. It is difficult to recommend improvements to flight parameters such as more overlap because the associated cost factor may increase. In terms of point cloud data, more accurate point cloud data might be able to eliminate some of the geometric errors.

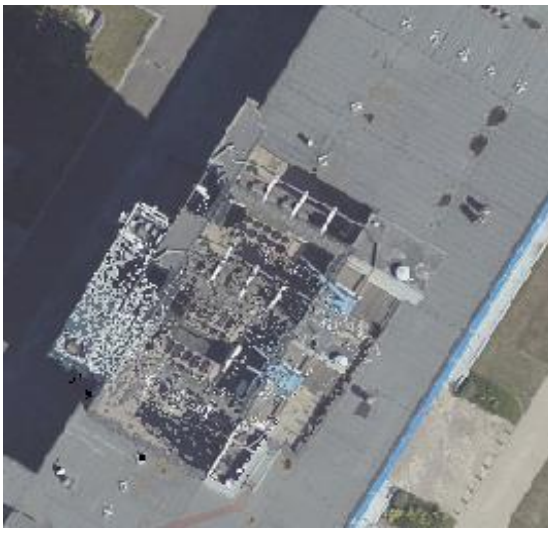

Figure 11: Squashed part on top of the building

Figure 12: Double mapped house

Figure 13: Flat roof on the building

RMSE:

POINTNAME	GCP		Ortho M		Difference x	Difference y	RMSE x	RMSE y	RMSE Middle Orthophoto
	x_1	y_1	x_m	y_m					
Point 1	255269.343	473281.877	255269.13	473282.38	0.212999999	-0.503000001	0.120330737	0.458841693	0.474357656
Point 4	255580.58	473183.119	255580.536	473182.38	0.043999999	0.739			
Point 2110	255187.495	472767.269	255187.5964	472767.3032	-0.1014	-0.034200001			
Point 1002	255535.7992	472571.8781	255535.8174	472572.0827	-0.018193	-0.204556			
	GCP		Ortho R		Difference x	Difference y	RMSE x	RMSE y	RMSE Right Orthophoto
	x_1	y_1	x_r	y_r					
Point 4	255580.58	473183.119	255580.536	473182.38	0.043999999	0.739	0.196210648	0.453584219	0.494203665
Point 1002	255535.7992	472571.8781	255535.8174	472572.0827	-0.018193	-0.204556			
Point 2	255813.14	473161.529	255813.3592	473161.8046	-0.2192	-0.275600001			
Point 1003	255892.7377	472616.0339	255892.4157	472615.6351	0.321992	0.398791999			

Figure 14: RMSE table for Middle and Right Orthophotos

As it can be interpreted from the above image, the RMSE of middle orthophoto came out to be 0.47 m and that of right orthophoto came out to be 0.49 m. It was not possible to calculate the RMSE of the left orthophoto as it only covers 2 points.

Figure 15: Third GCP point outside the coverage area in Left Orthophoto
(Note: the points are made larger in size on purpose for better visualization)

The RMSE value (~ 2 pixel) is a bit higher than expected especially in the y direction. One explanation for this high value is the temporal changes in some of the areas surrounding the GCP and since the corresponding orthophoto points were extracted by visual inspection it might give more error than expected.

4. Ortho-Mosaic Generation

Python was used for the process of orthomosaic generation. Since there were three generated orthophotos, first left and middle orthophotos were stitched together to make a LM mosaic then the middle and right orthophotos were stitched together to make a MR mosaic. Finally, both LM mosaic and MR mosaic were stitched together to get the orthomosaic of the whole study area.

Image Matching Algorithm:

For the task of image matching, SIFT algorithm was selected. The main advantage of using SIFT over algorithms such as Harris Detector is that SIFT is scale invariant. Thus, irrespective of the scale differences of the features in different images, the algorithm would be able to detect it.

The algorithm detects keypoints using the difference of Gaussians. It combines the scale pyramid approach with the difference in different levels of smoothening to detect keypoints which are distinct to scale changes. To generate the descriptor of the identified keypoints, the algorithm calculates the gradients in the local neighborhood (4 * 4) of the keypoints and computes an orientation histogram. This orientation histogram is discretized at 45-degree values (i.e., 8 bins). Thus, each keypoint would have a 128-dimensional descriptor associated with it.

Once the keypoints and descriptors are detected, we can match them based on the descriptor distance. And once we have the descriptors matched, we can find the transformation matrix between the source and the destination image (after dealing with the outliers using RANSAC algorithm).

Table 3: Python Implementation & Parameters

Python Implementation & Parameters:

First the SIFT object was generated:

```
sift = cv2.SIFT_create(10000)
```

Here, the parameter ‘number of features’ is set to 10000 as putting it on default might create more than 262144 training descriptors which will result in Assertion errors. This algorithm was used to compute and detect the keypoints and descriptors. Out of all the detected keypoints and descriptors, Lowe’s ratio test was used to filter out the bad matches. At last, 387 good matches were found.

```
keypoints_1, descriptors_1 = sift.detectAndCompute(grr_r, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(grl_l, None)

bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)
matches = bf.knnMatch(descriptors_1, descriptors_2, k=2)
good = []
for m, n in matches:
    if m.distance < 0.4*n.distance:
        good.append(m)
```

Here, the parameter 'cv2.NORM_L2' tells the matcher to find matches based on "Euclidean Distance". Also, in the bf.knnmatch function, the parameter 'k=2' sets up the keypoint and descriptor pairs in a way that we can later use the Lowe's ratio test.

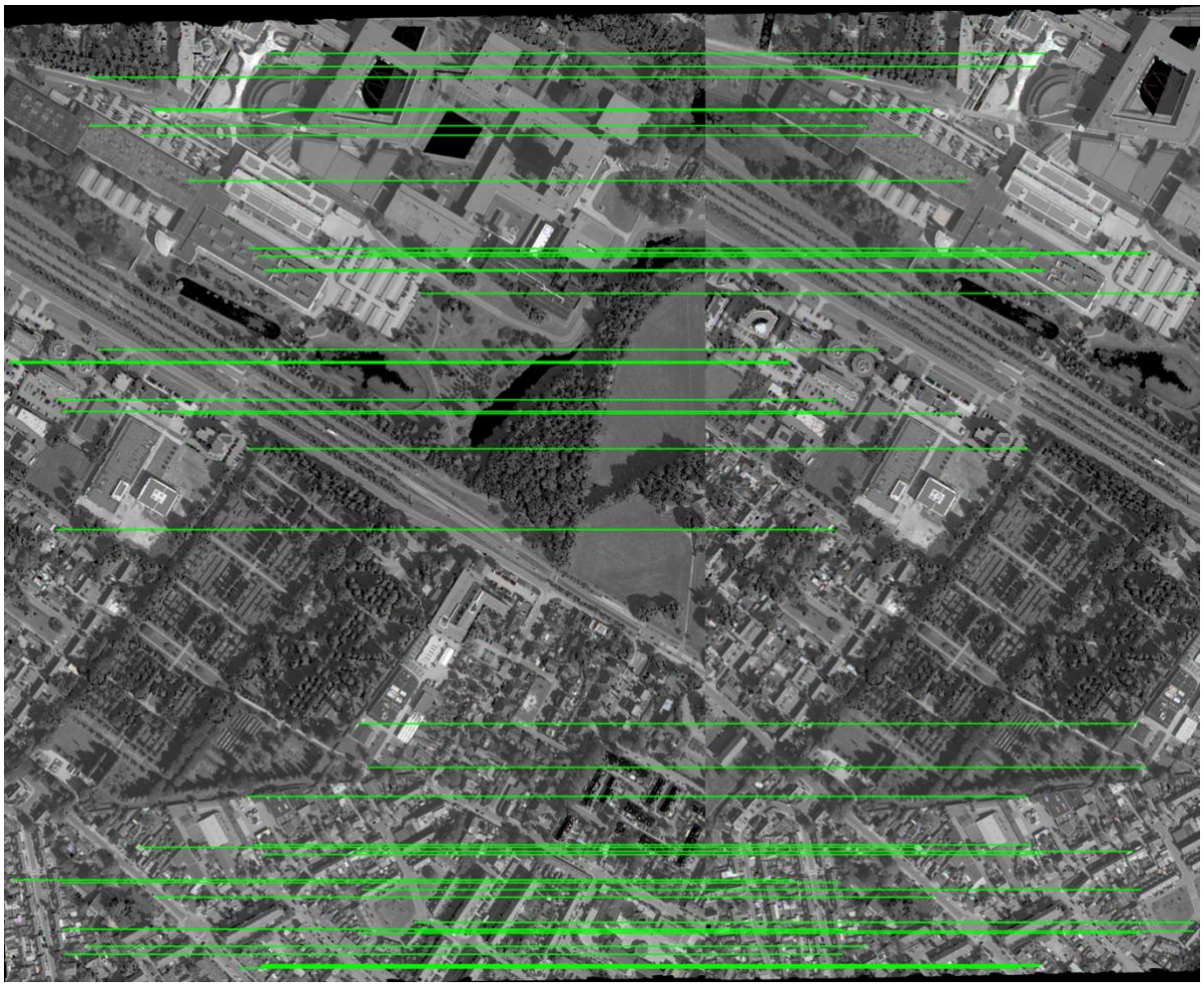

Figure 16: 50 good matches in the left-middle image pair (Total good matches: 387)

At this point we have the optimal matches. To deal with the outliers and compute the homography matrix, RANSAC method was used. (Note: RANSAC method needs at least 4 points to find the transformation matrix)

```
if len(good) >= 4:
    src = np.float32([keypoints_1[m.queryIdx].pt for m in good]).reshape(-1,1,2)
    dst = np.float32([keypoints_2[m.trainIdx].pt for m in good]).reshape(-1,1,2)
    (H, status) = cv2.findHomography(src, dst, cv2.RANSAC, 5.0)
else:
    print('bad')
```

The resulting homography matrix:

$$\begin{bmatrix} 1e(+00) & 3.914e(-05) & 3.935e(+02) \\ 7.712e(-05) & 1e(+00) & -2.401e(-01) \\ 2.564e(-08) & 8.144e(-09) & 1e(+00) \end{bmatrix}$$

This process was repeated for the middle-right image pair and then at the end the LM (left-middle) orthomosaic and the MR (middle-right) orthomosaic were combined to give the final orthomosaic.

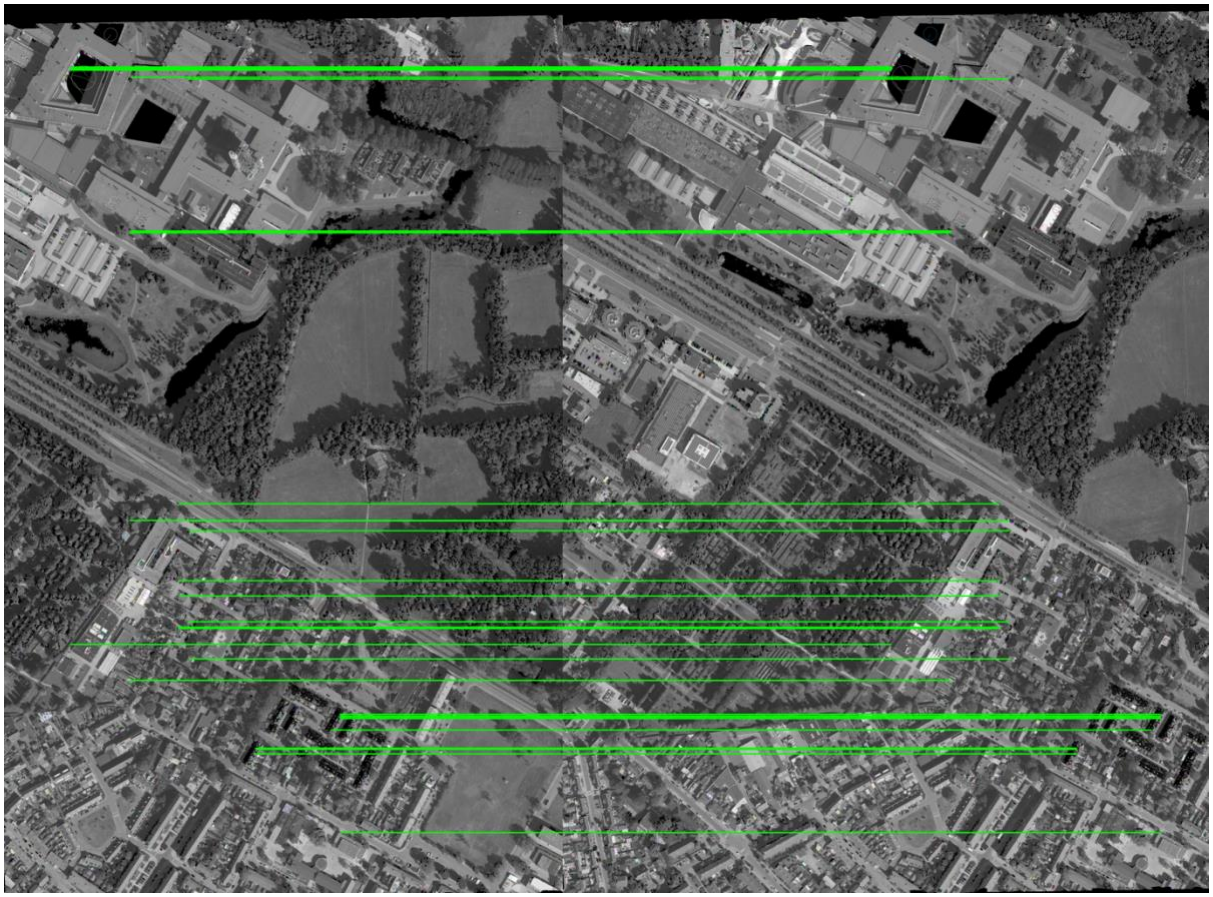

Figure 17: 50 good matches in the middle-right image pair (Total good matches: 1269)

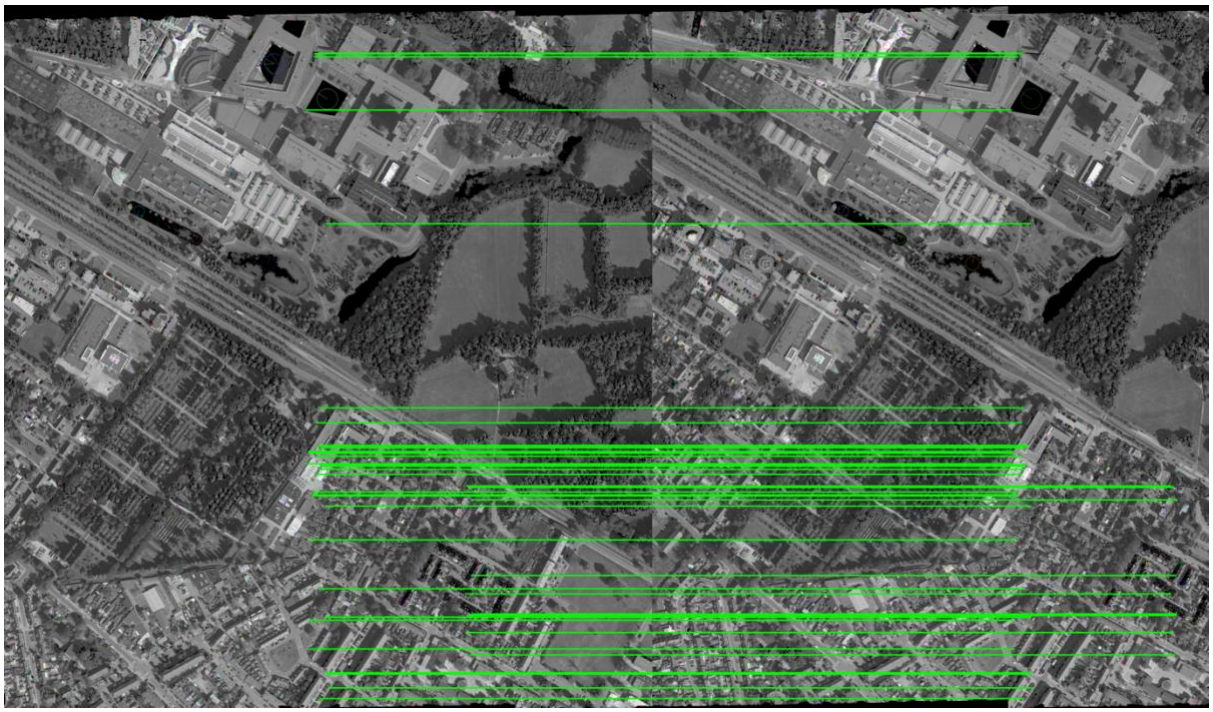

Figure 18: 50 good matches between the LM orthomosaic and MR orthomosaic (Total good matches: 255)

5. Ortho-Mosaic Reflection

The final orthomosaic output:

Figure 19: Final Ortho-Mosaic of the study area

One of the problems that was observed was related to the extent of the generated orthomosaic. When the .tiff file was opened in QGIS, it was observed that the orthomosaic is smaller than the study area. It contains the entire area, however the extent has somehow decreased. This problem was traced back to a weird reduction of the output array (for LM orthomosaic and MR orthomosaic) caused by the .png export. This issue can be solved by exporting the LM orthomosaic and the MR orthomosaic as .tiff, using the extent of the raster created by clipping

the area covered by the corresponding orthophotos. However, this was not carried out because of the shortage of time.

Apart from this, there were no observed errors concerning the orthomosaic process. There was no overly visible seamline or difference in radiometric resolution. Obviously, the radiometric and geometric errors observed in individual orthophotos will also translate to the orthomosaic but that can be dealt with before inputting the orthophotos to the orthomosaic generation process.