Aerial Image Mosaic

- Project Report -

Gladys ROCH
CentraleSupélec
Gladys.roch@supelec.fr

Abstract

This project implements an algorithm to stitch images together into a mosaic. The image set will include rotation and translations only. The general method is to firs,t detect characteristic features on each image (such as corners), then match features between images, the best matches are then used to estimate the rotation and translation that separates two images.

1. Introduction

Capturing multiple images from a same area and stitching them into a bigger image is used in multiple domains. It allows to construct high resolution images using cameras of limited resolution. This is often used in aerial imaging as shown in figure 1.

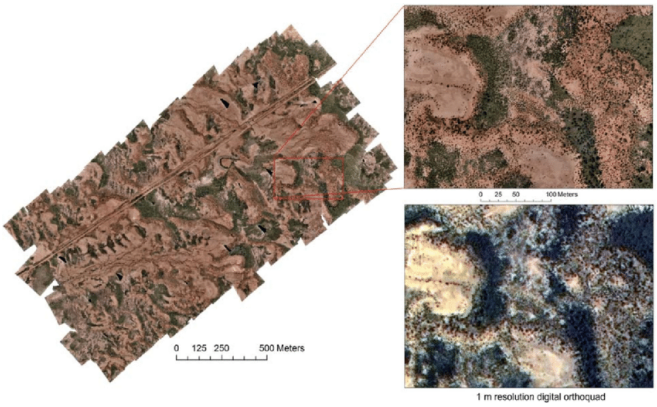

Figure 1. Example of high image resolution using image stitching [1]

1.1. Data

The sets of images to puzzle together come from the National Collection of aerial Photography (https://ncap.org.uk). The image dimensions are 750 x 750.

The first set of images to puzzle together is presented in figure 2. These 4 images have a high degree of superposition which reduce the computation load as each image can be stitched to any other (no disjoint images). Moreover, the exposition and contrast are uniform across the set, which allows to skip a preprocessing step.

The second set of images is more complex as it depicts a coastline including sea reflects and moving boats. (See figure 3).

Figure 2. Image set for mosaic reconstruction [2]

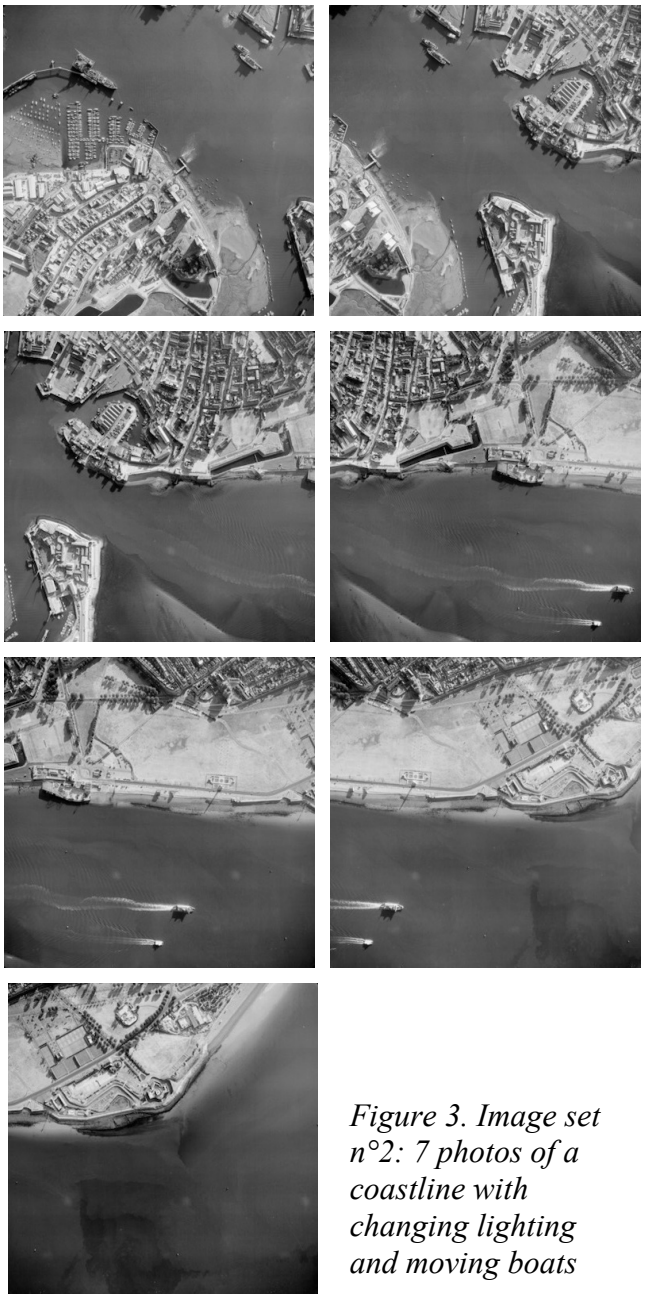

Figure 3. Image set n°2: 7 photos of a coastline with changing lighting and moving boats

1.2. Stitching Process

The process of image stitching considered for this project is in two steps. First, I extract characteristic points on each image, and then I will find the transformation (translation + rotation) that minimize the superposition error.

2. Implementation - Methodology

2.1. Feature points detection and description

To stitch 2 images, the first step is to detect and describe key features on each image that will then be compared to find matches between to two images. I first used Harris Corner Detection combined with SIFT Detector before shifting to ORB detector and descriptor which is must faster and give more matching points afterward.

2.1.1 Harris Corner Detection + SIFT Detector

One characteristic of aerial images used for cartography is that all images are at the same scale. Therefore, I only need to find rotation- and translation-invariant features: corners. Harris Corner Detector is an algorithm that detects theses corners using the difference pixel-intensity for a displacement in all direction. The OpenCV python library implements a cornerHarris() function that give the eigenvalue-ratio. Thresholding for a suitable value gives the corners of the image (0-ever pixel is a corner, 1-no corners). Below are the corners detected using OpenCV and a threshold of 0.4

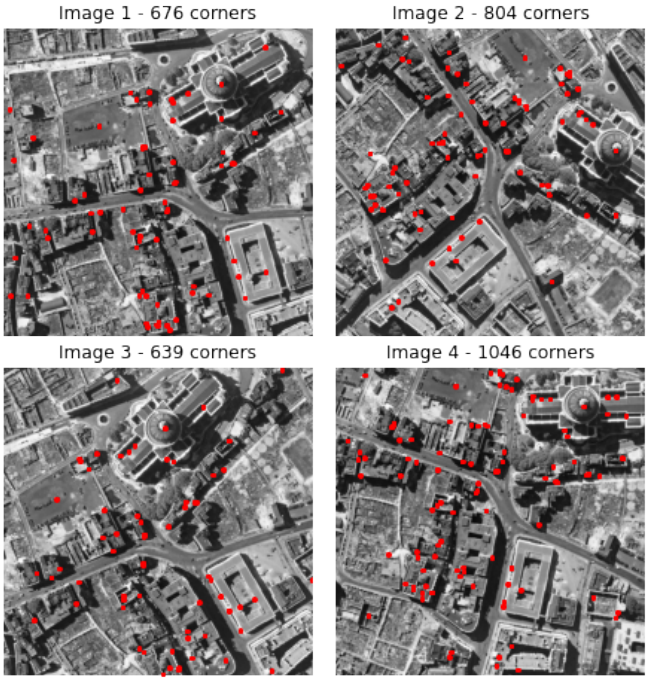

Figure 4. Corners using Harris Corner Detection and threshold of 0.4

To describe theses corners, I use the SIFT descriptor. It takes a 16x16 neighborhood around the keypoint, devided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histograms are created, to a total of 128 bin values. It is represented as a vector to form keypoint descriptor. To achieve robustness against illumination changes and rotation, normalization and thresholding are applied.

The computation of the SIFT description is time consuming.

2.1.2 ORB detector and descriptor

ORB builds on the FAST keypoint detector and the BRIEF descriptor. Both of these techniques are attractive because of their good performance and low cost.

FAST corner detector uses a circle of 16 pixels around the target pixel. Each pixel in the circle is labeled from integer number 1 to 16 clockwise. If a set of 12 contiguous pixels in the circle are all brighter or all darker than the intensity of candidate pixel (+threshold) then p is classified as corner.

For each key point detected by FAST, BRIEF builds a 128-long binary vector that describes it. Each bit of the vector is computed by comparing the intensity of 2 randomly selected pixels within a square patch centered on the target pixel. The random selection of pairs of pixels follows a Gaussian centered on the key point.

Because BRIEF is not rotation-invariant, ORB uses a variation of it called rBRIEF (Rotation aware BRIEF) descriptor. ORB also turns FAST into a scale invariant detector by using multiscale image pyramid.

The following figure shows the key points detected using ORB.

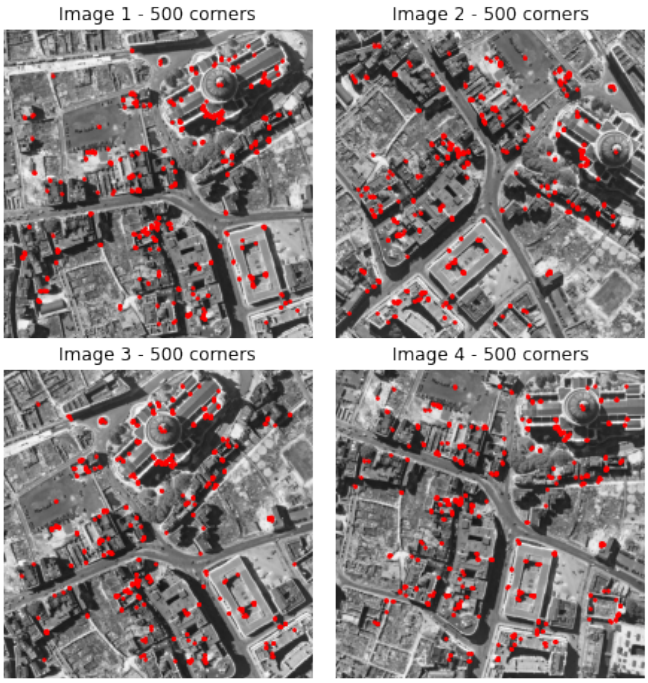

Figure 5. Corners using ORB Detector

2.2. Feature points Matching

To match features from 2 images, I use the Brute-Force Matcher from OpenCV. For each feature of image A, the BFMatcher computes the distance between its descriptor and all the descriptors from the image B and returns the keypoint with the smallest distance. For the SIFT descriptor the L2 Norm is good, however with the ORB descriptor is a binary vector, the Hamming Norm is sufficient.

$$d_H(\text{desA}, \text{desB}) = \sum_{i=0}^{n-1} (\text{desA}_i \oplus \text{desB}_i)$$

With the argument crossCheck=True, I keep matches only if they are reciprocal, i.e. when the the best match of keypoint kpA from image A is kpB (from image B), then the best match of kpB must be kpA.

The following figure shows the matches between image 1 and image 2 using the Brute-Force Matcher on the ORB descriptors. From the 500 key points on each image, BFMatcher detected

271 matches. The flowing figure shows the first 30 matches by order of Hamming distance.

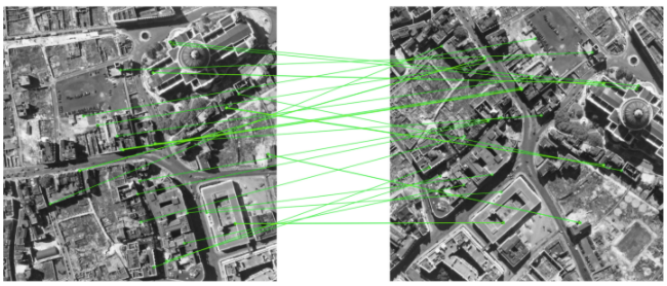

Figure 6. Matches between image 1 and image 2 using Brute Force Matcher

2.3. Translation estimation

The match with the smallest distance is the most likely to be exact, I choose to take this best match to stitch the images together. In the final image the 2 corners part of the best match will be superposed.

Once the matches are the next step it to estimate the rotation angle., he best rotation angle for a given matching corner pair will be estimated through a classical optimization method, such as least square regression. As the images have a lot of details, a relatively small region (100 pixels) should be sufficient to compute the loss for the optimization.

2.4. Rotation Estimation

The rotation angle between 2 images is estimated using the matches. Theoretically, 2 pairs of corners would be enough to estimate a rotation angle between images (it is the angle between the vectors formed by the 2 corners on each image). However, as can be seen on figure 6 some matches are wrong. To mitigate the effect of these outliers more matches are used. The RANSAC (RANdom Sample Consensus) algorithm is an iterative method used to estimate the parameters of a mathematical model. It allows for a robust optimization despite outliers, as shown in figure 7. This way we can use all the matches found to compute the homography.

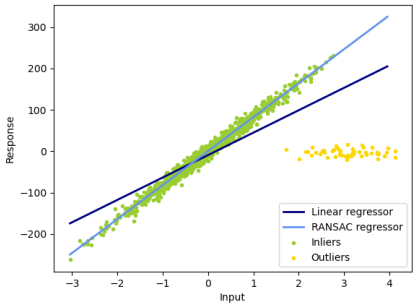

Figure 7. RANSAC algorithm for linear model

OpenCV implements a function that estimates the homography using RANSAC. The homography is the perspective transformation between the source and the destination planes it contains the information about the rotation angle. The homography is a 3x3 matrix.

$$H = \begin{matrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{matrix}$$

The findHomography function from OpenCV find the coordinates hij that minimize the projection error:

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

where x, y and x', y' represent the coordinates of the pixels image A and B respectively.

From the homography matrix estimated using the RANSAC method we can extract the rotation angle in the plane. It is a z-axis Euler angle (radian) is given by

$$\theta = arctan2(h_{21}, h_{11}) = 2\arctan\left(\frac{h_{11}}{\sqrt{{h_{21}}^2 + {h_{11}}^2} + h_{21}}\right)$$

The image B can now be rotated around their center by the angle θ. Before the rotation, margins are added to the images to prevent the loss of data.

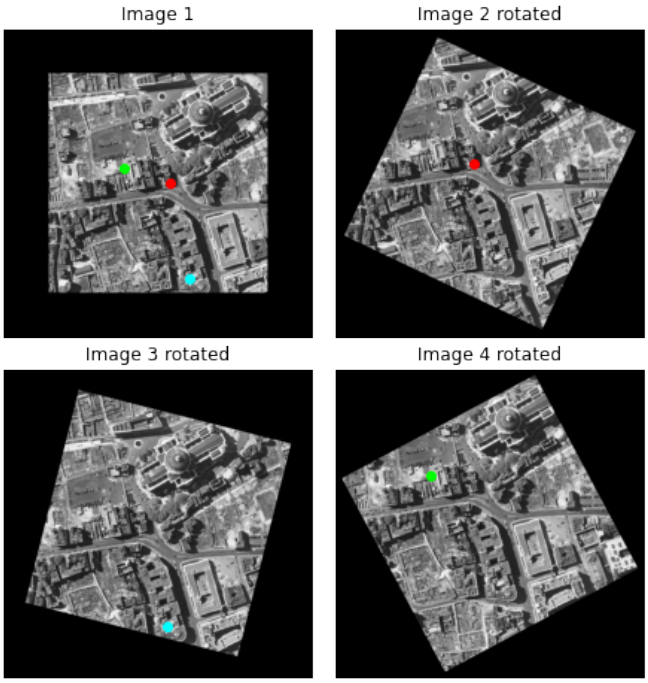

Figure 8. Rotated images around their respective best corner (all images are matched with image 1)

2.5. Stitching

The rotated images are finally stitched together.

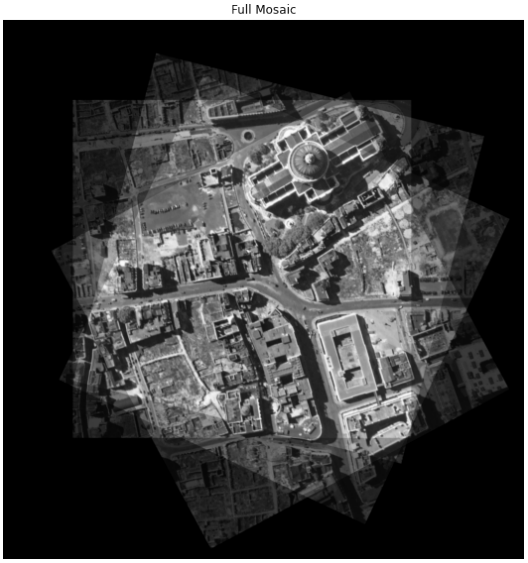

Figure 9. Image Mosaic set n°1

3. Results

The following image is the final mosaic for the set n°2. The method still works with body of water and moving objects such as boats, but the result is blurry, meaning the hard lines are not perfectly superposed. This comes from an error in the rotation angle. and

Unfortunately, any of the following options improved significantly the mosaic.

- changing the number of matches used (keeping only the 10 best matches) in the homography determination improved the rotation angle by less than 2%
- changing the threshold used by RANSAC to classify inliers/outliers did not affect the angle.
- using harris + SIFT corners gave less matches and therefore did not improve the final result.

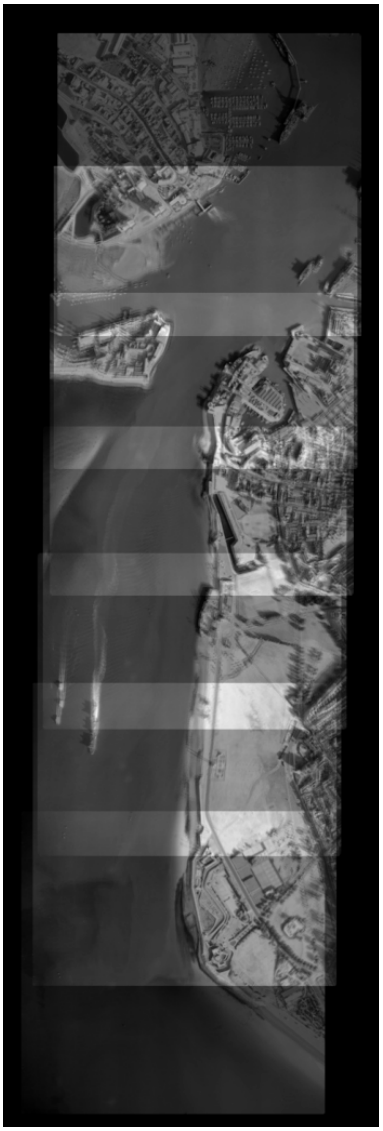

Figure 10. Image Mosaic set n°2

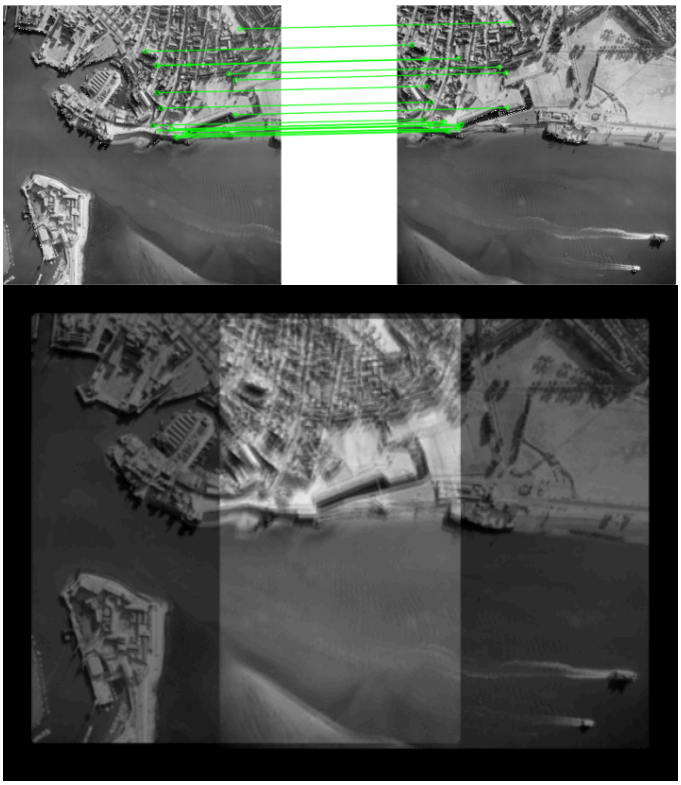

Figure 11. Focus on image 2 and 3 matches and stitching

4. Conclusion

The algorithm proposed works very well with images of constant contrast and colors (set n°1) but less so with shading. It is however resilient towards moving objects (boats) when they are a small part of the image. The model proposed is standard for image stitching and can be adapted for more complex hymnographies (with scaling and tilting).

References

[1] Laliberte, Andrea & Scientist, Remote & Winters, Craig & Specialist, Gis & Rango, Albert. (2008). A procedure for orthorectification of sub-decimeter resolution imagery obtained with an unmanned aerial vehicle (UAV).

[2] Yaqing Ding, Jian Yang, Jean Ponce, Hui Kong. An Efficient Solution to the Homography-Based Relative Pose Problem With a Common Reference Direction. ICCV 2019 - International Conference on Computer Vision, Oct 2019, Seoul, South Korea. ffhal-02266544f

[3] Işık, Şahin. (2014). A Comparative Evaluation of Well-known Feature Detectors and Descriptors. International Journal of Applied Mathematics, Electronics and Computers. 3. 10.18100/ijamec.60004.