Co-alignment of Aerial Push-Broom Strips using Trajectory Smoothness Constraints

Erik Ringaby*, Jörgen Ahlberg†, Per-Erik Forssén* and Niclas Wadströmer†
*Computer Vision Laboratory (CVL), Linköping University
Email: {perfo,ringaby}@isy.liu.se
†Sensor Informatics Group, Swedish Defence Research Agency (FOI), Linköping
Email: {jorahl,nicwad}@foi.se

***Abstract*—We study the problem of registering a sequence of scan lines (a *strip*) from an airborne push-broom imager to another sequence partly covering the same area. Such a registration has to compensate for deformations caused by attitude and speed changes in the aircraft. The registration is challenging, as both strips contain such deformations.**

Our algorithm estimates the 3D rotation of the camera for each scan line, by parametrising it as a linear spline with a number of knots evenly distributed in one of the strips. The rotations are estimated from correspondences between strips of the same area. Once the rotations are known, they can be compensated for, and each line of pixels can be transformed such that ground trace of the two strips are registered with respect to each other.

I. Introduction

In airborne remote sensing, push broom imagers are commonly used. Such an imager has a spatial resolution of $1 \times K$ pixels, and exploits the ego motion of the platform on which the imager is mounted to form an image. The imager in itself is thus equivalent to a scanning sensor, but without the scanning mechanics. The platform, typically a fixed-wing aircraft, does not move in a perfectly straight line, and moreover rotates slightly around all three axes. This causes distortions in the resulting image (a sequence of lines), which thus needs registration and orthorectification to be useful for most applications. Hardware for navigation (GPS, INS) and sensor stabilisation can be used to some degree, but are not always available.

Our primary applications are change detection and anomaly detection [1], while mapping and signature-based target detection are secondary. For change detection to be possible at all, the image lines of the two (or more) acquired data strips must be registered with high precision. In order for any kind of detection (change, anomaly, target) to be georeferenced, the imagery must be registered to other available orthorectified images.

In this paper, we propose a method for mutual registration of push broom data strips. The method is an adaptation of the method developed at CVL for rolling shutter video sensors.

The outline of the paper is as follows. The scenario, the sensor, and the data are described in Section 2, the registration method is described in Section 3, results are shown in Section 4, and conclusions are drawn in Section 5.

II. Sensors and data

The imager, ImSpec, is a visual and near-infrared (391–961 nm) hyperspectral imager from SpecIm [2], with 1024 pixels in each scan line and a maximum of 256 spectral bands. Due to limitations in read-out electronics' data rate, the number of spectral bands might need to be reduced to meet requirements on the number of lines to be acquired per second. In this experiment 60 spectral bands are recorded, which is more than enough for our applications. The imager is mounted nadir-looking in a small fixed wing aircraft as shown in Fig. 1. Data was acquired by flying over approximately the same land strip twice, a rural area at the Swedish Army Ground Combat School premises at Kvarn outside Linköping. The flight altitude was 1000 meters, yielding a pixel footprint of around 0.5 meters. The aircraft was equipped with GPS, but for unknown reasons the GPS data was not logged to the hard disk during the flight. Two resulting data sets are shown in Fig. 2.

For the purpose of strip alignment, we view each ImSpec strip as one image. For visualisation, and correspondence finding, a mean of three wavelengths approximately corresponding to blue, green and red are used.

III. Registration method

Our algorithm estimates the 3D rotation of the camera to compensate for the misregistration due to the aircraft's rotation. The algorithm can be summarised in the following steps:

1) Initial strip alignment using SIFT-features and a global homography model
2) Dense point correspondences with KLT using the initial alignment
3) Separation of the strip into segments where rotation is assumed smooth
4) Rotation estimation
5) Image rectification

A. Homography estimation with SIFT-features

Even though the pilot tries to fly over the same area for each strip, the paths may differ (due to e.g. small rotations during the flight). The spatial location of the first row may also differ if the data gathering started at different locations. The aircraft speed may also differ between different strips.

Fig. 1. The sensor installation in the aircraft. Top: The sensors are installed in the metal box beside the pilot. Bottom: Computer and storage equipment in the back seat.

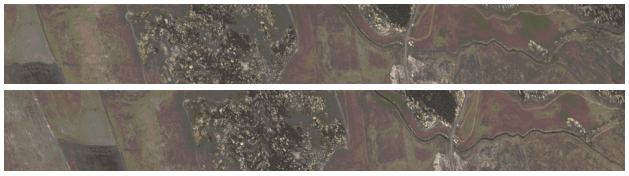

Fig. 2. Two strips from approximately the same area.

To get a coarse initial alignment of the strips we use SIFT descriptors [3] together with RANSAC to estimate a homography, see Fig. 3. In each strip, SIFT finds between 10 000 and 25 000 features. From these we select the 300 with the highest ratio score, and use these to estimate a homography using RANSAC. We have used an inlier threshold of 50 pixels on the symmetric transfer error [4].

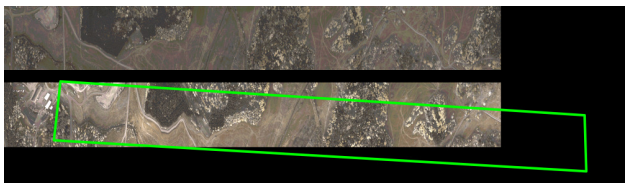

Fig. 3. Initial homography alignment found with SIFT and RANSAC. Top: strip 1, bottom: strip 2, with transformed bounding box from strip 1 overlaid.

B. Point correspondences with KLT

In order to estimate the camera rotation we need point correspondences between the strips. We obtain them by tracking points with the KLT-tracker [5], [6]. The KLT tracker uses a spatial intensity gradient search which minimises the Euclidean distance between the corresponding patches in the different strips. We use the scale pyramid implementation of the algorithm in OpenCV. Correspondences from KLT benefit from the initial homography alignment, and also from first scaling the forward axis of each dataset by a factor of two. The first image is initialised with a regular grid of points which are tracked and re-tracked. This means that when a point is tracked from the first strip to the other one, it is tracked again and if it returns to the original position the point is regarded as a match [7]. This procedure removes outliers efficiently.

C. Separation of the strip into segments

The rotations during a flight are assumed to vary smoothly, and thus the strip can be split into several segments. At each line that splits two segments, we introduce a *key-rotation*, i.e. a knot in a linear spline that interpolates aircraft rotations. The number of key-rotations needed may differ depending on how long the strip is, and how big the aircraft rotations were (i.e. if the strips differ much). We have chosen to have N key-rotations uniformly spaced in the first strip and to use local linear regression to calculate where these rotations approximately should be in the the other strips, see Fig. 4.

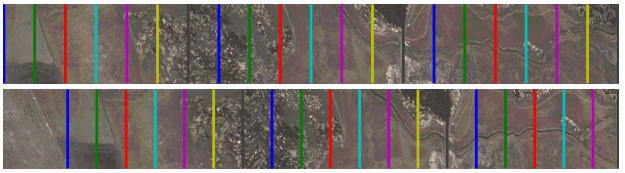

Fig. 4. Rotation distribution with N = 21

D. Rotation estimation

We model the distortion of a strip as a sequence of rotation homographies:

$$\mathbf{H}(t) = \mathbf{K}\mathbf{R}(t)\mathbf{K}^{-1}\,, \tag{1}$$

i.e. we neglect the translational component of the aircraft motion. This means that we model the sensor as rotating purely about its optical centre, and thus the imaged ground patch is modelled as being on the interior surface of a sphere. This is bound to cause some distortions in the reconstruction, but if the radius of the sphere (i.e. the focal length in $\mathbf{K}$) is large enough (compared to the strip length), this distortion is small.

We optimise for a sequence of rotations $\mathbf{n}_1,\ldots,\mathbf{n}_N$ using the MATLAB optimiser `lsqnonlin`. The optimisation makes use of a cost function

$$J = \epsilon(\mathbf{n}_1,\ldots,\mathbf{n}_N)\,, \tag{2}$$

with image correspondences $\mathbf{x}_k \leftrightarrow \mathbf{y}_k$ as constant parameters. The rotations $\mathbf{n}_1,\ldots,\mathbf{n}_N$ are represented as

three element vectors, where the magnitude corresponds to the rotation angle, and the direction is the axis of rotation, i.e. $\mathbf{n} = \varphi\hat{\mathbf{n}}$. This is a minimal parametrisation of rotations, and it also ensures smooth variations, in contrast to e.g. Euler angles. The vector $\mathbf{n}$ can be converted to a rotation matrix using the matrix exponent. Because we are dealing with only rotations this can be simplified to Rodrigues formula.

In order to make the optimisation more stable, the first and last key-rotations in the first strip is set to identity rotations. To help the global optimisation with an initial guess, smaller segments (areas between vertical lines in figure 4) is locally optimised and used as input to the global optimisation.

We have also augmented the cost function with a regularisation, where we put an extra cost on large changes between consecutive rotations. This trajectory smoothness constraint is done by adding a term in the cost function:

$$J = \epsilon(\mathbf{n}_1, \ldots, \mathbf{n}_N) + \alpha \left(\sum_{l=1}^{N-1} 1 - \hat{\mathbf{n}}_l^T \hat{\mathbf{n}}_{l+1} \right) . \quad (3)$$

This regularisation is necessary, otherwise the optimiser may find very strange trajectories, see Fig. 6.

E. Image rectification

When the key-rotations have been estimated, they can be used to assign a rotation to each row using SLERP (Spherical Linear intERPolation) [8]. With this we know how each point should be displaced in order to rectify the scene.

We have chosen to perform the rectifying interpolation in three steps: First, we create an all-zero RGBA image. Second, we apply the rectification to each pixel in the strip image. The 3×3 closest grid locations are then updated by adding vectors of the form (wr, wg, wb, w). Here r, g, b are the colour channel values of the input pixel, and w is a variable weight that depends on the grid location $\mathbf{y}$, according to:

$$w(\mathbf{y}) = \exp(-.5(\mathbf{y} - \mathbf{x}')^2/\sigma^2) . \quad (4)$$

Here $\mathbf{x}'$ is the sub-pixel location of the pixel, and σ is a smoothing parameter, which we set to $\sigma = 0.15$. Third, after looping through all pixels, we convert the RGBA image to RGB, by dividing the RGB values by the fourth element. The whole operation is quite fast, and its parallel nature makes it well suited to a GPU implementation. The number of channels may be changed to correspond to the correct number of bands.

Alternatively, the irregular grid of pixels can be resampled to a regular grid, by defining a triangular mesh over the points, and sampling the mesh using bi-cubic interpolation. This is done by the function `griddata` in Matlab. This method gives a good result but is much slower than the previous proposed method.

Finally, it is also tempting to use regular, or *inverse interpolation*. We can now loop over all values of output pixels, and use inverse mapping to find the pixel locations in the distorted image, and cubically interpolate these, see Fig. 5, right. This is fast but it does not give as good a result as the previously proposed methods because the interpolation takes place in the distorted image.

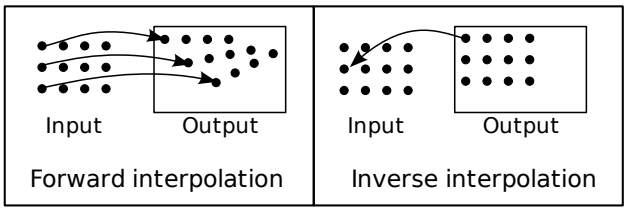

Fig. 5. Forward and inverse interpolation. Left: In forward interpolation, we rectify the locations of all pixels to obtain an irregular grid. Neighbours are now correctly defined by the irregular grid. Right: In regular interpolation, one uses the inverse mapping to find out where to sample points. Neighbours are then defined by the regular grid where the distorted pixels lie.

IV. Results

The results presented here are from one of the three strips acquired during the flight. Fig. 6 shows the result when the regularisation parameter is set to zero. The strips are overlapping but it is difficult to make use of the images. When using $\alpha = 60\,000$ as regularisation parameter the trajectory is much smoother and the images look more like the original scene, see figure 7.

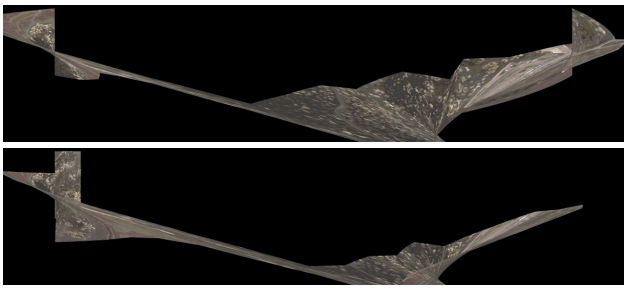

Fig. 6. Result for aligning the two strips from the first flight path, without regularisation of large deviations in rotation.

V. Conclusion and future work

In order to exploit the hyper-spectral imagery for the mentioned targeted applications, there are additional steps that need to be taken. Where the strips are overlapping in figure 8 left, many false change detections are made. The pixel registration needs to be more exact, which can be achieved by exploiting navigation data (GPS, INS) from the aircraft as well as a suitable motion model. Also additional imagery can be used for registration. For registration, simultaneously acquired imagery from a staring sensor with high spatial and low spectral resolution (for example a consumer digital camera) can be used, possibly enabling registration exact enough for spectral change detection.

In general, we got better results the more rotations we added. The downside was much higher computation time, and additionally the optimisation had an increasingly harder time to converge.

Fig. 7. Result for aligning the two strips from the first flight path, with $N = 21$, and regularisation parameter $\alpha = 60\,000$.

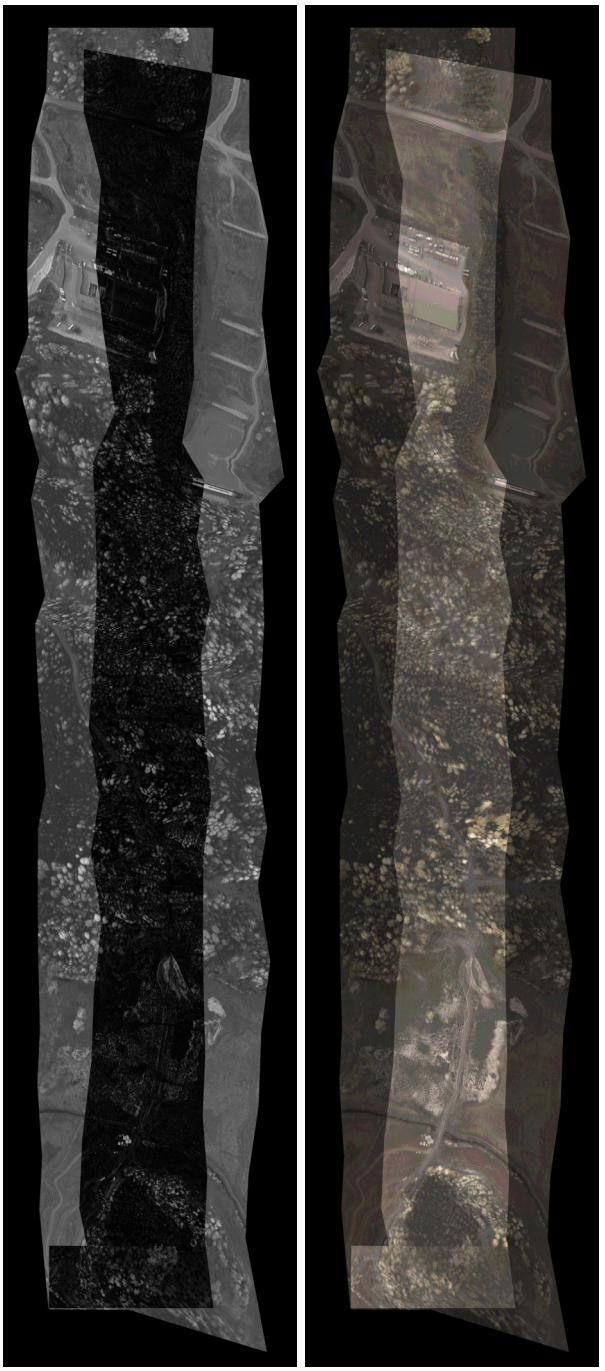

Fig. 8. Left: The difference image for the two strips in figure 7. Right: The average of the two images.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Thomas Svensson at the IR Systems Group at FOI for the data. The flights were performed by SkyMovies AB.

REFERENCES

[1] J. Ahlberg and I. Renhorn, "Multi- and hyperspectral target and anomaly detection," Swedish Defence Research Agency (FOI), Tech. Rep. FOI-R--1526--SE, December 2004.

[2] T. Chevalier, "Samverkande sensorer. illustrerat exempel med hyperspektral kamera och 3d-laserradar," Swedish Defence Research Agency (FOI), Tech. Rep. FOI-R--2325--SE, September 2007.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, January 2004.

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[5] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679.

[6] J. Shi and C. Tomasi, "Good features to track," in *CVPR94*, Seattle, June 1994.

[7] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *IEEE ICCV*, 2007.

[8] K. Shoemake, "Animating rotation with quaternion curves," in *Int. Conf. on CGIT*, 1985, pp. 245–254.