Statistical methods for georeferencing and terrain modeling

Master Thesis

Victor Poughon
July 2015

DTU Space
National Space Institute

Abstract

Earth observation offers global insight into local phenomena for monitoring, observation, maintenance and decision making. Among the many available solutions, airborne images are a solution at a unique resolution, price and availability combination. The need for accurate georeferencing and terrain modeling arises as an essential step in data processing of aerial images. Working from a data set consisting of 117 aerial images acquired with a Nikon D700 camera, this project investigates the complete data processing pipeline. The digital products considered are orthographic images and digital terrain models. After a theoretical exposition of the methods, implementation of a complete georeferencing system is described. The results are analysed for quality through inspection of the resulting digital products and for stability using the bootstrap method. Finally, recommendations concerning problem design, algorithm selection and software engineering are presented. This Master Thesis is a final year project for the Earth and Space Physics and Engineering master program at DTU Space.

Contents

Chapter 1

Introduction

Observation of the Earth from the air or from space offers a unique and global insight into local phenomena. Although images of the Earth today come primarily from satellites, and flying drones are undergoing rapid development, airborne imagery remains the appropriate solution for many combinations of price, resolution and availability. In particular, high resolution ground imagery at a reasonable price is made available by on-demand aerial data acquisition.

The field of photogrammetry has been studying the analysis and processing of aerial images for many decades. The initial focus was on reconstructing the geometry of the observed objects, with the inherent challenge of determining the ”lost” dimension of the imaging process, when going from the 3D world to a flat 2D image. More recently, the field of computer vision has been converging to the same problems, with a sometimes different approach. The result is an extremely rich body of literature and previous work to draw from, which is both a blessing and a curse.

Commercial applications of aerial images consists in the fabrication of so-called *digital products*. The typical digital product is the orthorectified map, typically available on the web to be easily consulted. The second important digital product is the terrain model. A terrain model is a 3D representation of the shape of the imaged object, in this case the ground (with potentially trees, cars, buildings, bridges and other elements). Maps and terrain models have a lot in common. They are both extremely useful and complementary for an entire class of maintenance, monitoring, observation and decision helping tasks. Most importantly, the processing necessary to the fabrication of those products requires knowing accurately the positions and orientations of the cameras at the moment each image was taken. This task is called *georeferencing*, and arises because of uncertainties in other positioning and orientation methods, such as GPS and IMU, which do not provide enough accuracy.

Georeferencing is the key element of the canonical aerial image processing pipeline studied in this thesis. It is inseparable from terrain modeling because the method of reconstructing the camera positions, and the object positions in the real world is essentially the same. It happens simultaneously during one single optimization procedure, where the parameters of the problem are chosen to best fit the available observations.

This study is realized in the context of AscendXYZ. AscendXYZ is a map production and image analysis company and provided the data for this project. This project is also a Master Thesis and represents a final year project at DTU.

Chapter 2 details the georeferencing problem and formalizes mathematical notation. Chapter 3 is an exposition of the proposed data processing pipeline, with detailed steps from images to digital product. Chapter 5 presents the results of the implementation, analysis and interpretation. Focus will be put on implementation decision as well as recommendations for building a complete georeferencing system.

I wish to extend many thanks AscendXYZ for providing data for this project, valuable advice and feedback and a work space. Thank you to Lars Stenseng and Poul Kjeldager Sørensen for supervising my thesis and giving feedback and guidance. Finally, my sincere gratitude to DTU and DTU Space for giving me the opportunity to study Earth and Space Physics and Engineering for two years. It was a very enjoyable and rewarding experience.

Chapter 2

Problem formulation

2.1 Objective

The main objective of this project is the analysis of the production of digital maps from aerial images. This process involves image processing, mathematical modeling, numerical optimization and photogrammetry. Its inherent complexity means that the implementation of such a system requires software engineering considerations.

The problem is therefore triple: First, propose a complete method to georeference aerial images. Second, implement the method to validate the theoretical work. And third, analyse results extensively to provide detailed answers to accuracy, speed and correctness concerns.

The two statistical tasks to be accomplished are georeferencing and terrain modeling. They are non separable, because they happen simultaneously during the optimization process, and therefore must be considered as one operation. The two digital products of interest are digital maps, in the form of orthophotos, and terrain models in the form of DTM or point clouds. Although the production of digital products is not the focus of this work, this thesis contains basic implementations useful to evaluate the results of the georeferencing procedure.

The input of the georeferencing problem is the set of images acquired from the data collection campaign. Additional input data can be useful is present, such as altimeter data, camera calibration parameters or LIDAR scans. Conceptually the images have been acquired at some location in space over the terrain. This is illustrated in figure 2.1. To produce digital maps, the positions and orientations of the cameras have to be known with high accuracy. But typically only low accuracy GPS positions and inaccurate or incomplete orientations are available. The cameras' six degrees of freedom coordinates are then adjusted simultaneously to optimize the

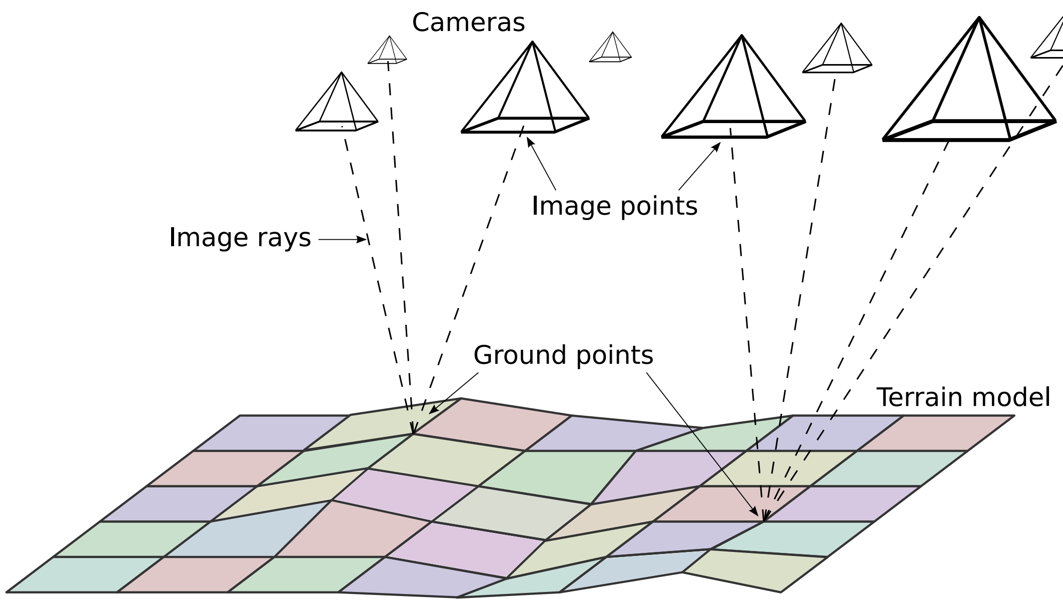


Figure 2.1. Visual representation of the problem geometry

image rays geometry. For this reason the problem is called *bundle adjustment* in photogrammetry and computer vision.

The output of the considered problem is whatever is necessary to the production of digital products. In general this will be the adjusted cameras positions and orientations, as well as a collection of points on the ground with varying density.

2.2 Definitions

Let the cameras positions and orientations be specified by vectors C_i of size 6

$$C_i = \begin{bmatrix} X & Y & Z & \omega & \phi & \kappa \end{bmatrix} \ \forall i \in 1 \ldots m \tag{2.1}$$

A point X on the ground has three coordinates

$$X_i = \begin{bmatrix} X & Y & Z \end{bmatrix} \ \forall i \in 1 \ldots n \tag{2.2}$$

It may be visible in one or more image, in that case its image coordinate, expressed in meters on the camera's sensor is

$$x_{ij} = \text{proj}(X_i, C_j) = \begin{bmatrix} x & y \end{bmatrix} \tag{2.3}$$

where $\text{proj}(X_i, C_j)$ is the *projection function*, which transforms a 3D world point X_i into a 2D image points for a specific camera position C_j. There may be up to $m \times n$ such image points, but in typical flight overlaps this number will be closer to $3N$.

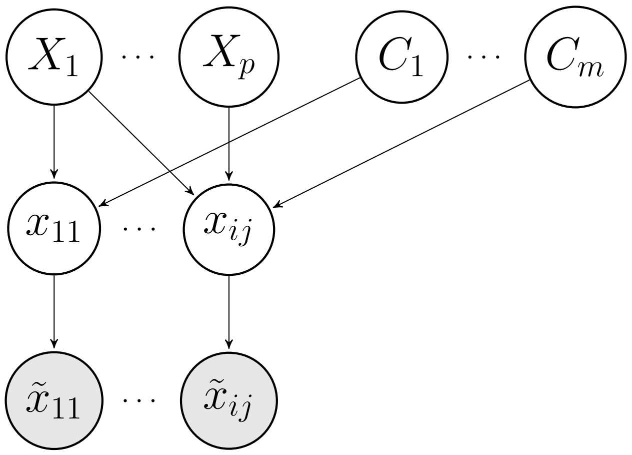


Figure 2.2. Random variables' dependence in a bundle adjustment problem. Only noisy image coordinates $\tilde{x}_{ij}$ are observed.

Using computer vision, *image features* are extracted from images corresponding to the image of the same object in different images. This process is good but not perfect. As a first approximation it is modeled as a random variable following a Gaussian distribution with fixed covariance:

$$\tilde{x}_{ij} \sim \mathcal{N}(x_{ij}, \Sigma) \tag{2.4}$$

The global independence relationships of the random variables of the bundle adjustment problem are represented as a Bayesian network in figure 2.2. Only the noisy image coordinates $\tilde{x}_{ij}$ are observed and not all combinations are available, only those where images overlap. Finally, the projection function can be non linear when considering lens distortion effects. The combination of non linear effects, large dimensionality and non unique solutions are at the core of the difficulty of automatically georeferencing images.

2.3 Data set

Most of this project's practical work uses images from the Alinta Stockpile data set (figure 2.3). This data set is from an aerial flight over a stockpile site from Australia. It consists of

- 117 images from the on board camera.
- Calibration reports including measured focal length, principal points and lens distortion.
- Lidar point cloud from an aerial scan acquired the same day as the images.

- Ground truth terrain model and adjusted camera positions from commercial photogrammetry software *Photoscan*.
- Ground control points.

This data set was chosen because it is highly representative of a typical aerial photography campaign. It requires no specific treatment and any solution with generalize to other data sets. Furthermore, it offers additional advantages such as calibration information, ground truth from commercial photogrammetry software and ground control points.

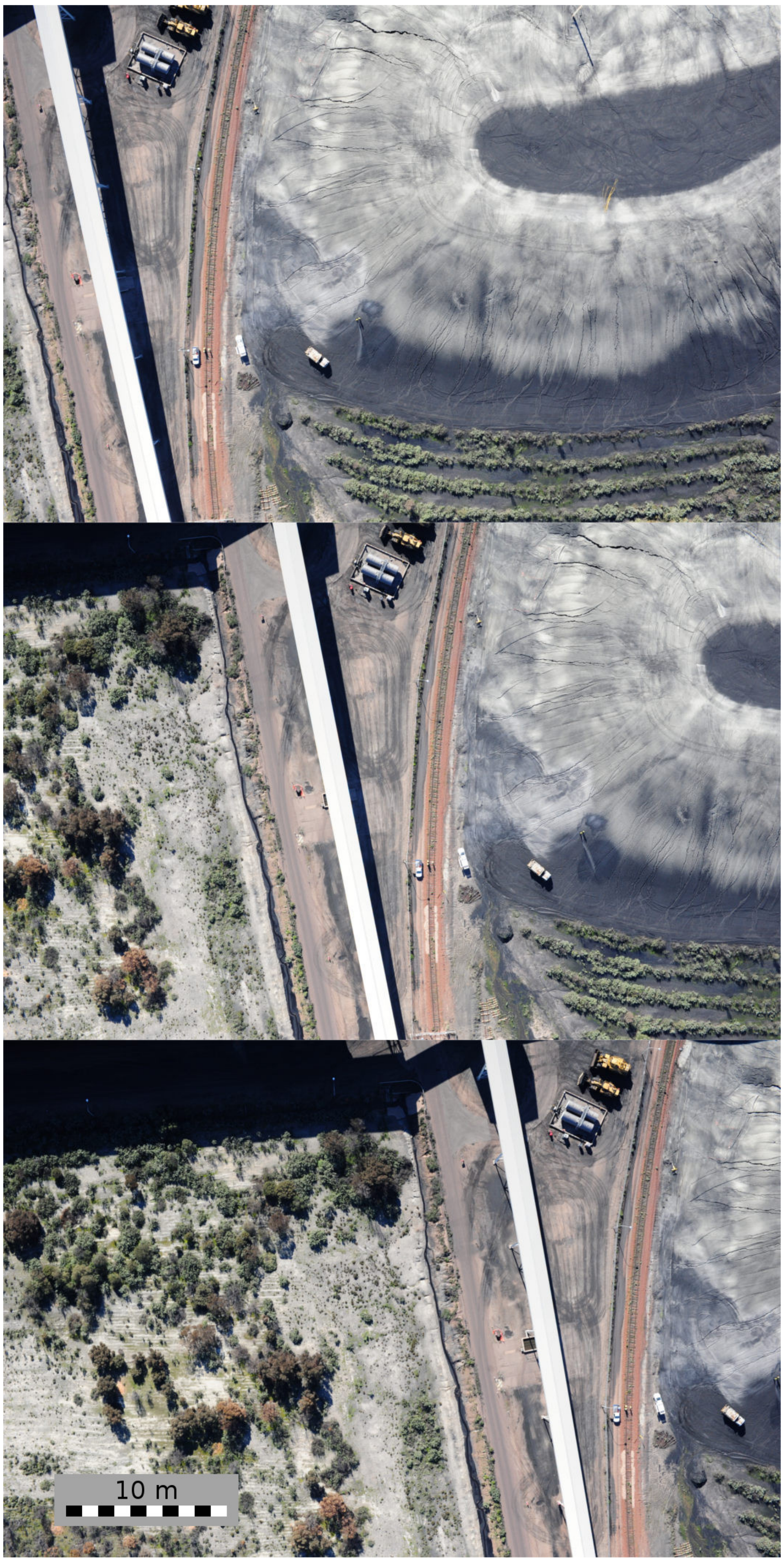


Figure 2.3. Typical sequence of three images from the Alinta Stockpile data set. Most points are visible in at least three images in the flight forward direction.

2.4 Previous work

Mapping from aerial images is one of the most important application of photogrammetry. Progress in computing power and cheap sensors has shifted the focus from a careful data acquisition and processing campaign to more widely available data sets obtained and processed on consumer hardware.

It is important to note that the two communities of photogrammetry and computer vision have approached the problem from two different angles, and their technologies are only now starting to converge. On the one hand, the traditional approach of photogrammetry is called bundle adjustment, or bundle block adjustment. On the other hand, computer vision refers to Structure from Motion, abbreviated SfM. The photogrammetric philosophy is to carefully plan out a campaign in advance, carry it out precisely and perform extensive quality control on the results. However, computer vision researchers focus more on the ”point and shoot” approach. It is typical to expect a fully automated and reliable procedure, to the point that a method can be considered unusable if it requires to much user intervention.

In any case, very impressive results have been achieved by both communities. Some examples include: ”Building Rome in a day”[2], in which the authors construct 3D models of city buildings from publicly available internet images or combined bundle block adjustment and orbit determination[22]. The methods generalize to other areas of active research such as video stabilisation [12], simultaneous localization and mapping (SLAM) [19], or multi-core processing [31].

The best text on the details of bundle adjustment is definitely ”bundle Adjustment - A Modern Synthesis” [27]. It exposes extensively the details of the method through numerical optimisation, parametrization, network structure, implementation strategies, the issue of gauge freedom and quality control.

There are a variety of commercial software solutions available today for aerial images processing. For example Agisoft Photoscan is the solution used at AscendXYZ.

2.5 AscendXYZ

AscendXYZ operates in the field of low cost, high quality map production and image analysis. This is done using the power of cloud computing to work on large data sets, with as many automated processes as possible. Enabling acquisition companies to upload and handle data worldwide.

Clients are on the one side aerial image acquisition companies and on the other, clients that use aerial and mapping data in their everyday business. In the past few years mapping systems have gone from being something only large engineering companies could do to being something a private person can do. This mainly is due to pricing of the equipment for image acquisition, and this equipment also requires a larger airplane to facilitate the data acquisition.

Data acquisition has moved to UAVs and small manned airplanes like Cessna 172 that AscendXYZ uses. AscendXYZ is also in their final tests of getting a box certified that can be strapped onto a Cessna 172, that requires no airplane modifications which will lead to even more small data acquisition companies. Yet the mapping pipeline services and tooling for working with consumer based camera mapping are lagging behind and there are only a few players in this market and their software all has disadvantages due to them wanting to be a general image stereo matching software for general modeling.

Therefore it is of create value for both AscendXYZ and others in this field to research better tooling for generating point clouds and ortho-rectified maps. AscendXYZ also sees a lot of interest in LIDAR data. It is of value to AscendXYZ to have more research done into clarifying the differences of LIDAR point clouds and point clouds from stereo matching.

Chapter 3

Theory and method

This chapter outlines the complete data processing pipeline of georeferencing, from image formation to digital products algorithms. It also introduces image features through keypoints, descriptors and matching algorithms.

3.1 Image formation

The first necessary element of theory, is understanding the process of image formation. How is an image formed from objects in the real world, to pixel values on the camera detector? Accurate modelling of this process is divided in two steps. First, the perspective projection, which represents the transformation from the three dimensional world to the two dimensional detector. Second, the distortion effects in the camera lens are modeled to account for impurities in the manufacturing process.

3.1.1 Perspective projection

Knowing the coordinate of an object point, the image coordinate of that point can be computed, if the following is known:

- Focal length (interior orientation)
- Position and orientation (external orientation)

The fundamental principle of the perspective projection, is that the image point, perspective center and object are aligned. The transformation from world to image coordinates is then:

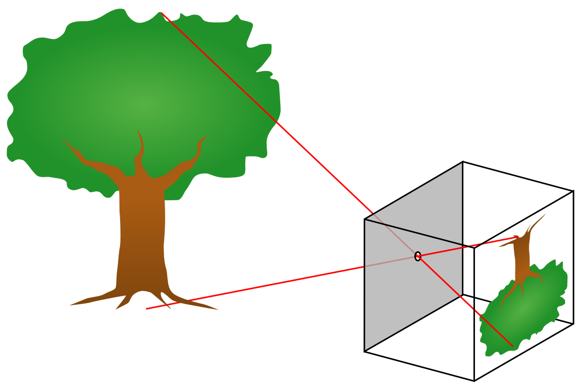


Figure 3.1. Illustration of the ideal pinhole camera model. Image from Wikipedia [28].

- Translation by X_c, the position of the camera
- Rotation around the X axis by angle Ω
- Rotation around the Y axis by angle ϕ
- Rotation around the Z axis by angle κ
- Projection on the image plane (using the focal length)

In the ideal case (without a principal point), the above five transformations are elegantly expressed in homogeneous coordinates:

$$\hat{x} = P \cdot R_\kappa \cdot R_\phi \cdot R_\Omega \cdot T \cdot X \tag{3.1}$$

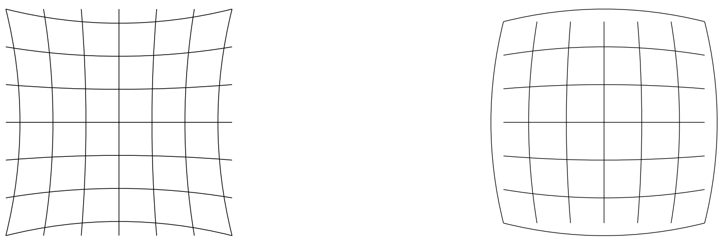
Where $\hat{x}$ is the image coordinate, X is the world coordinate, and:

$$
\begin{aligned}
R_\Omega &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Omega) & \sin(\Omega) & 0 \\ 0 & -\sin(\Omega) & \cos(\Omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
R_\phi &= \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
R_\kappa &= \begin{bmatrix} \cos(\kappa) & \sin(\kappa) & 0 & 0 \\ -\sin(\kappa) & \cos(\kappa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
T &= \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
P &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-1}{f} & 0 \end{bmatrix}
\end{aligned}
\tag{3.2}
$$

This perspective projection explains the so-called *parallax effects*. Thus it is impossible to inverse with only one image of an object, because information is effectively lost. It is needed to see an object in at least two images to perform triangulation and determine all three of its coordinates. Terrain modeling generalizes this procedure by reconstructing multiple objects from multiple cameras in one single optimization problem.

3.1.2 Lens distortion

Lens distortion is the result of imperfect manufacturing. It generally comes in two kinds: pincushion and barrel distortion, as illustrated in figure 3.2. Brown's model [10] can be used to effectively model lens distortion, and take it into account during processing.



(a) Pincushion lens distortion **(b)** Barrel lens distortion

Figure 3.2. Illustration of the two most common types of lens distortion (image from Wikipedia [29].

Given the measured coordinates x_{meas} and y_{meas} of a pixel point, the corrected position is given by:

$$\begin{aligned} r^2 &= (x_{meas} - x_p)^2 + (y_{meas} - y_p)^2 \\ d_r &= K_1 r^3 + K_2 r^5 + K_3 r^7 \\ x_{corr} &= x_{meas} - x_p + \frac{x d_r}{r} + P_1\left(r^2 + 2x^2\right) + 2P_2 xy \\ y_{coor} &= y_{meas} - y_p + \frac{y d_r}{r} + P_2\left(r^2 + 2y^2\right) + 2P_1 xy \end{aligned} \tag{3.3}$$

Where x_p and y_p are the principal point coordinates and P_1, P_2, K_1, K_2, K_3 are the parameters of the model.

This highlights the role of calibration. For high quality modeling and reconstruction, it is necessary to obtain those parameters for each sensor and lens combination used. Note that the higher order coefficients can be ignored to control the model complexity. Another solution is treat the distortion parameters as unknowns of the optimization problem. However this greatly increases the problem complexity and can lead to more difficult solving.

3.2 Image features

This section exposes the theory behind image features. In general, an image feature is a precise location in an image that is reasonably unique. It can be for example a corner, a high contrast region, a low entropy area, etc. The purpose of feature detection is to associate features in two or more images identifying the same point in object space, even though the images are taken in different conditions (angle, lighting, exposure, zoom level, etc.).

The theory behind feature recognition is an extremely well developed area in computer vision. They are at the core of many application including camera calibration, object detection and 3D reconstruction. The OpenCV library includes eleven different algorithms related to image features (in publication order): MSER [18], SIFT [17], FAST [24], SURF [8], Star [4], Brief [9], ORB [25], BRISK [15], AKAZE [6], KAZE [7] and FREAK [5]. The following is not a complete survey of published work. Instead, it exposes the general structure of image features algorithms which is composed of the three steps: keypoints detection, description and matching.

There exists such a variety of solutions because of the heuristic nature of the problem and the differences in the output's requirements. Typically requirements include invariance to scale, rotation, translation, perspective and parallax effects, and more generally to the transformation that is to be reconstructed.

3.2.1 Keypoints

The first step of features detection is finding keypoints. A keypoint is a point of an image that has some unique property, like a specific response to a filter. Its coordinate is expressed in pixels units. An example of keypoints found by the KAZE algorithm is shown in figure 3.3. There are many different approaches to finding keypoints, and the Harris corner detector [11] is one of the first published feature detector. It focuses on finding edges and associated corners by using a filter based on the eigenvalues of the second moment matrix. Later improvements to this method added support for scale invariance and rotation invariance.

3.2.2 Descriptors

The second step in image feature recognition is the extraction of descriptors. Descriptors are strings of information identifying a keypoints. The challenge in designing a good feature descriptor is to introduce a high variance between different points to increase the quality of the matching, while maintaining robustness to small variations across view points. Therefore, descriptors need to have a high enough di-

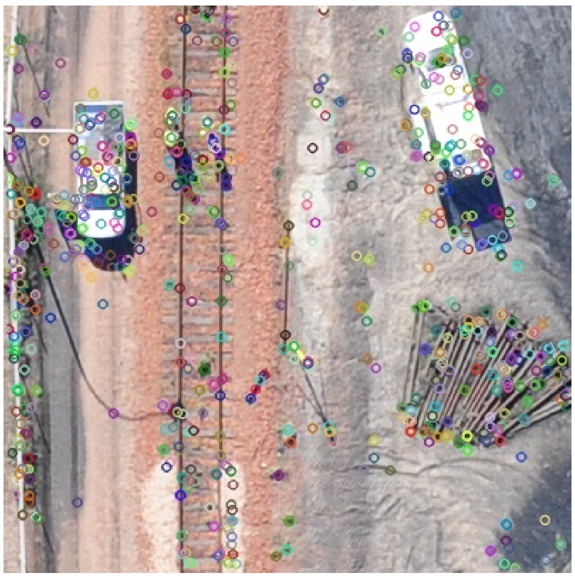


Figure 3.3. Example of keypoints found by the KAZE algorithm

mensionality to discriminate between keypoints descriptors. On the other hand, a too high descriptor dimensionality implies a complex matching step.

3.2.3 Features matching

The last step in features recognition is matching keypoint descriptors. An example of this procedure is shown in figure 3.4. This project will use the FLANN [21] matching algorithm, which is a significant improvement over the brute force method.

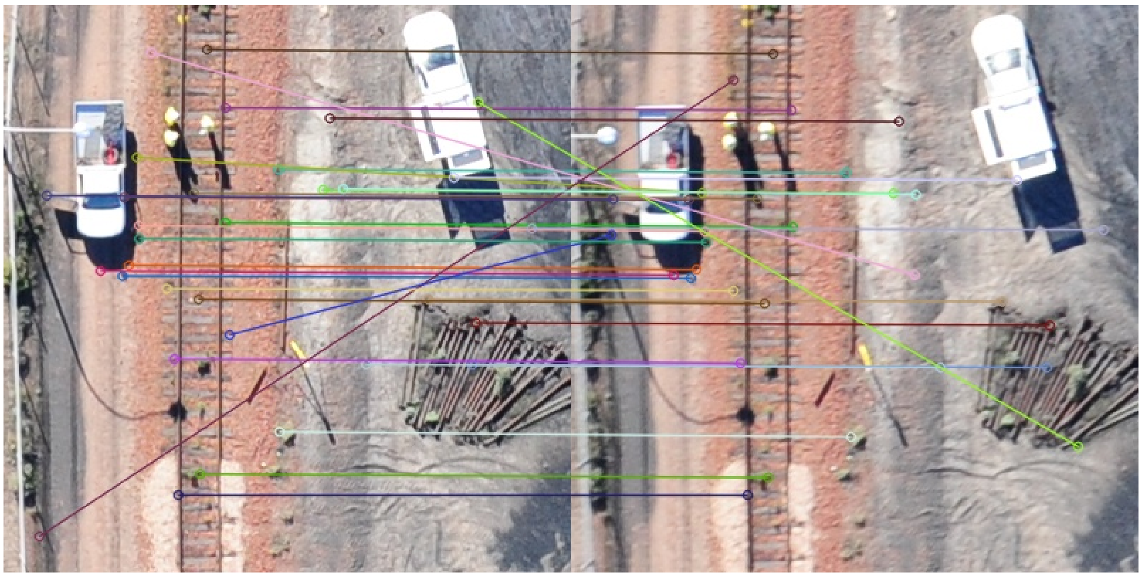


Figure 3.4. Matching 32 features from the BRISK algorithm using the FLANN matcher. Note the presence of many outliers.

3.3 Geometric reconstruction

The geometric reconstruction part of the problem is formulated under the Bayesian maximum aposteriori framework. The parameters to be estimated are those which have maximum probability given the data and their prior distribution:

$$\theta^* = \arg\max_{\theta} P(\theta|\hat{x}_{ij}) \tag{3.4}$$

In a typical aerial photogrammetry situation there can be hundreds of parameters in the θ vector, hence simplifying assumptions need to be made. The parameter vector θ is divided in *parameter blocks* θ_k containing the parameters X_i and C_j relevant to the projection of x_{ij}. Note that both ground points and cameras are involved in more than one image ray, therefore parameters blocks are not a partition of all parameters but rather a set of subsets.

Then, using Bayes' theorem:

$$\theta^* = \arg\max_{\theta} \prod_k P(\hat{x}_{ij}|\theta_k)P(\theta_k) \tag{3.5}$$

Where $P(\theta_k)$ is the prior distribution of the parameters.

In section 2.2 it was assumed that the image features $\hat{x}_{ij}$ follow a bivariate normal distribution around the true projection points x_{ij}. Therefore from equations (2.3) and (2.4):

$$P(\hat{x}_{ij}|\theta_k) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\hat{x}_{ij} - \mathrm{proj}(\theta_k)\right)^T \Sigma^{-1}\left(\hat{x}_{ij} - \mathrm{proj}(\theta_k)\right)\right) \tag{3.6}$$

where T indicates matrix transposition and Σ is the image features covariance matrix. Using the fact that log is a continuous and monotonous function, equation (3.5) becomes

$$\begin{aligned} \theta^* &= \arg\min_{\theta} -\log \prod_k P(\hat{x}_{ij}|\theta_k)P(\theta_k) \\ &= \arg\min_{\theta} \sum_k -\log P(\hat{x}_{ij}|\theta_k) + \sum_k -\log P(\theta_k) \\ &= \arg\min_{\theta} \sum_k \frac{1}{2}\left(\hat{x}_{ij} - \mathrm{proj}(\theta_k)\right)^T \Sigma^{-1}\left(\hat{x}_{ij} - \mathrm{proj}(\theta_k)\right) + \sum_k -\log P(\theta_k) \end{aligned} \tag{3.7}$$

This is the general maximum aposteriori form and is solvable as a non linear optimization problem given an initial guess for θ and an analytical expression for $P(\theta_k)$. Nonetheless further simplifying assumptions can be made by considering some terms as constants and dropping them in the above equation: the prior $P(\theta_k)$ (then considered uniform) and the image features covariance matrix Σ. Equation (3.7) becomes:

$$\theta^* = \arg\min_{\theta} \sum_k ||\hat{x}_{ij} - \mathrm{proj}(\theta_k)||^2 \tag{3.8}$$

which is the more familiar non linear least square. This expression is sometimes referred to as the *reprojection error*. It is non linear to allow for complex projection functions modeling all camera effects such as lens distortion and principal point displacement.

Conveniently, equation (3.7) allows the integration of ground control points (GCP) as visible image points with fixed ground correspondents. This is very useful for increasing accuracy of the procedure and fixing the coordinate system to the real world. Typically a few ground control points per flight are recommended, although it is possible to georeference without it. The results will simply be in some arbitrary local coordinate system.

3.4 Non linear least squares

Numerical optimization is a wide and mature topic. Therefore in this report the inner details of solving equations (3.7) and (3.8) will not be exposed. The general recommendation when working on a problem that uses numerical optimization is to not reinvent the wheel and use a well performing, state of the art algorithm. This is the approach taken in this project with the Ceres-Solver [3] library.

A very useful feature as a user of a non linear solver system, is automatic differentiation. Most optimization methods require the knowledge of the derivatives of the function to be minimized. Automatic differentiation is different from both symbolic and numerical differentiation. It enables the user to only write the main projection function's code, while the Jacobian is computed is evaluated automatically using the algebra of dual numbers. In Ceres-Solver, this is accomplished using a combination of C++ templates and operator overloading.

3.5 Bootstrap stability analysis

The non linear least square framework developed above yields the maximum likelihood parameter vector θ^*. However it gives no information on the distribution of the results, in particular how robust it is relative to changes of the observations. The bootstrap is a statistical method based on resampling It works by replicating virtual data sets from the original observations, from which the stability of the output can be estimated. The main advantage is that the method is generic, and therefore independent of the underlying estimation model.

Knowing the stability of the georeferencing solution is important for several reasons. First, the accuracy of the result needs to be known. This is most evident in commercial situations where a quantitative evaluation of the quality of the results is desired. Secondly, it is necessary when developing and researching new geometric models for doing, for example, systematic comparison of outputs. This is the main reason for using the bootstrap in this project. The results are presented in chapter 5.

Bootstrap samples B are created by resampling the original observations $\hat{x}_{ij}$ with replacement:

$$\begin{aligned} B_1 &: (\hat{x}_1, \hat{x}_{22}, \ldots) \\ B_2 &: (\hat{x}_{19}, \hat{x}_7, \ldots) \\ &\vdots \\ B_n &: (\hat{x}_2, \hat{x}_{11}, \ldots) \end{aligned} \tag{3.9}$$

For each of those new samples, the model is solved by performing the optimization again. The results is a list of M results from which interesting statistics can be computed.

The size of each bootstrap sample is chosen to be the same as the original observation vector, while the number of bootstrap samples M is a trade-off between computation time and statistical significance.

Another method of stability analysis is to look at the result's covariance matrix. The result's covariance matrix can be computed using the bootstrap as described above, or with $J^T(\theta^*)J(\theta^*)$ where J is the Jacobian matrix. This second method is not investigated in this project because of edge cases when the Jacobian is rank deficient, and because the bootstrap method can also estimate the result's covariance.

3.6 Terrain modeling

Terrain modeling is a vast topic in geoscience and geostatistics. There are many different possibilities for representing digital terrain models (DTM): point based, grid based, triangle based, continuous or discontinuous and hybrid methods.

To keep things simple, this project will only use the point cloud representation of a terrain model. That is, a terrain is simply a list of 3D world points.

Typically only a few features per images are needed for good georeferencing accuracy. But to obtain a detailed terrain model, it is necessary to optimize the position of many ground points to produce a so-called high density point cloud. The recommended procedure is to use an additional optimization sequence, with fixed cameras positions and orientation. New image features are then computed with much higher density and lower tolerance. This is possible because the images are already matched, meaning that features can be searched along the epipolar lines, greatly focusing the search.

3.7 Coordinate systems

In the georeferencing problem, different coordinate systems are to be considered. The first are the local world coordinates in which cameras and ground points are represented. This is easier to work with and enables to work with coordinates at the scale of the problem, because the origin is close to the flight path. Then, this local coordinates are converted to a common global coordinate system, such as WGS84 or UTM. This is done using ground control points as described in section 3.3.

However, this local to global conversion is not without issues. At least three GCPs are needed for accurate global referencing, although more are better. When using external maps for GCPs acquisition, errors sources from the global points can accumulate and contaminate the results. Additionally, perspective effects due to feature points not at the nadir, on buildings and other vertical structure can be the source for incorrect ground control points. Furthermore, working with national coordinate systems can introduce even more errors. For example data sources can inadvertently mix coordinate systems which are relatively close to each other and can be confused.

Figure 3.5 shows sensor geometry and its associated coordinate systems. It is important to keep in mind the different references and units when working in sensor geometry because there can be up to three points of reference:

- Units in meters, relative to the optical center. This is used when computing the result of the projection function.
- Image coordinates in pixels, relative to the top left corner, with the x axis horizontal. This is the coordinate system used by OpenCV internally, and most computer vision algorithms.
- Image coordinates in pixels, relative to the top left corner, with the x axis (usually then called i axis) vertical. This is the layout corresponding to the internal data structure used to store images in computer memory.

Conversion between the above three reference systems is straightforward, but forgetting which point is expressed in which system can lead to frustrating errors.

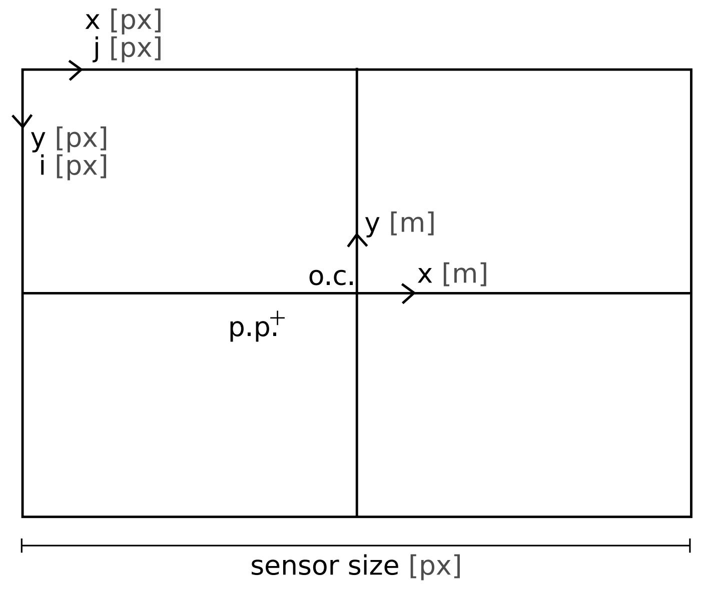


Figure 3.5. Sensor geometry and coordinate systems

Chapter 4

Implementation

A significant part of the work for this project is the software implementation of the georeferencing and terrain modeling method described in the previous section. This section describes the involved data in terms of input and output, the decisions made with respect to the implementation overall architecture, and the details relative to the implementation of non linear optimization, the bootstrap and digital products production.

The focus of this project is the design and implementation of the georeferencing problem. Therefore two parts of the method were not implemented to save time: initial solution via absolute orientation and high density point cloud generation. They are fairly independent problem with good coverage in the literature. The focus is maintained on mathematical modeling of the problem and analysis of the solution via bootstrapping and visual inspection of results.

Implementing a complete georeferencing system is not without difficulties. The problem shows a complex and interconnected internal structure. How to organize the software modules? How can the processing data and intermediate states be saved and reusable? It is also important to be able to serialize processing data to disk storage. This enables two things: the systematic analysis of results and a gradual approach to computing. It is very useful in particular because some processing step are very time consuming. This approach allows each step to be executable independently, or in so-called *batch mode*. It is implemented with the *JSON* format.

Additionally, every step is controlled via scripting to maintain a high degree of control over the research aspect of the project. While the core is written in C++, each feature is controlled via Python. The interface between the two languages is realized either via the JSON files, or via the *Cython* binding system. Different approaches can be efficiently and cleanly compared and composed. Finally, results are reproducible and each step is timed with the Unix *time* function to measure implementation performance.

4.1 Data pipeline

Figure 4.1 is an overview of the different data processing steps. Inputs are of course the set of images acquired from the data collection campaign. The image format can be anything as long as it is readable (this project uses a JPEG compressed data set). This allows the data collection to be independent of the processing step. In some cases, the entity processing the images does not have control over the data acquisition and so it is of prime interest to be able to work with many different sources. Additional inputs can be added but are not required: camera calibration parameters are very useful for high accuracy reconstruction, existing DTM model for improving or increasing precision, altimeter data or LIDAR scans.

It is important to this project that the implementation not a black box. Indeed, each internal step will be interesting for the analysis of results and important to evaluate each decision's impact. To achieve this, each step described in the Theory section is executable independently of the other. Additionally, all output is logged to a log file, as shown in Figure 4.1. Data structures are also serialized to a *JSON* [1] file. This is useful for the hierarchical modeling approach, exporting and importing to and from other photogrammetry software, and managing the project organization on the work station.

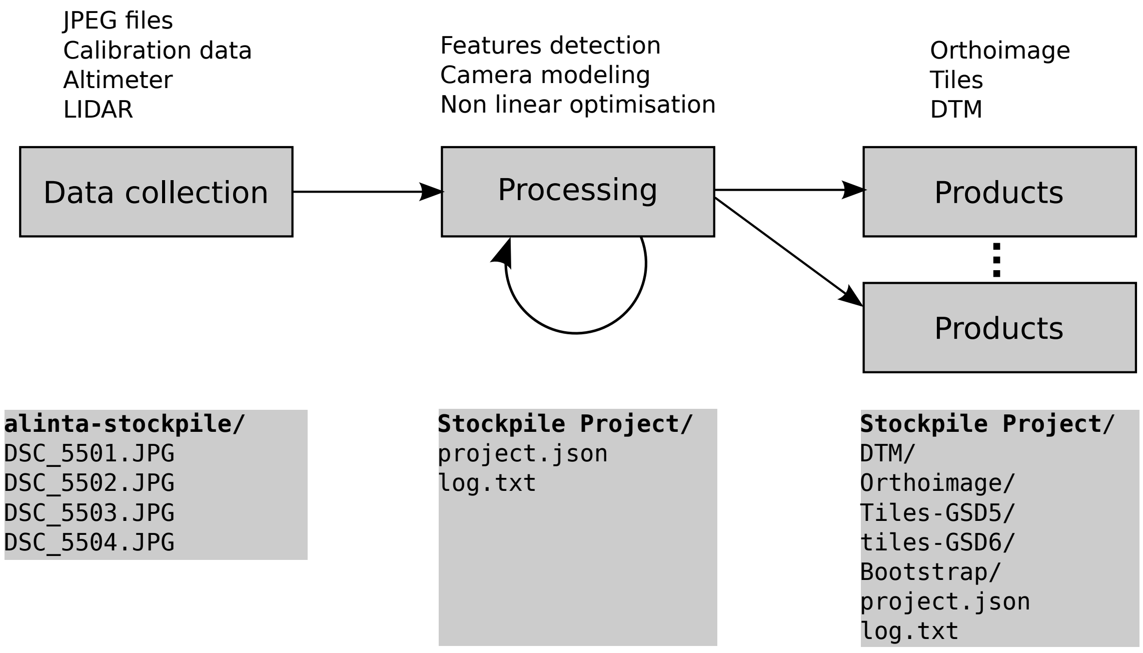


Figure 4.1. Data pipeline. The loop over the processing step represents the hierarchical modeling approach.

4.2 Software architecture

Figure 4.2 represents the principal software components implemented in this project. The key element is the *Model* class. A model essentially represents parameters to be optimized, from an initial approximate solution to a local optimum, minimizing a specific reprojection error. Most importantly, each model has a parent model from which the initial parameters vector for optimization is initialized. This implements so-called *hierarchical modeling*. A parent model can be for example, GPS positions from a flight recorder, geometric solution from absolute orientation, or any other model object. Successive models improve accuracy by increasing the number of parameters available to optimization (degrees of freedom), the internal parameter structure, the projection function, the relative tolerances or the optimization algorithm parameters. They can also change features points algorithm parameters or input image resolution to provide high accuracy positions or high density point clouds.

This approach is necessary to enable working in high dimensional parameters space. Non linear least square can only find local minima. Therefore too many degrees of freedom come with an increased difficulty and can even render the task impossible. Additionally, this hierarchical approach allows for comparability of models. This resonates with the need to manage the complexity of different problems, different projection functions and different parameter structures. The ability to compare and evaluate results is also needed to complete the research part of this work.

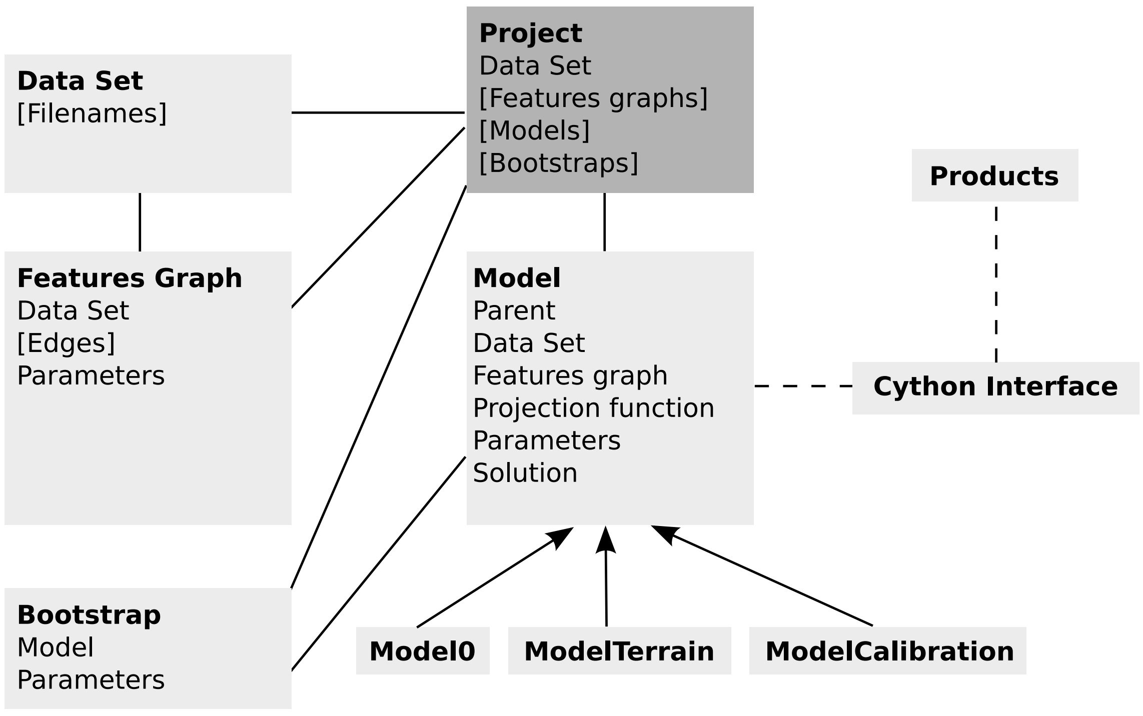


Figure 4.2. Software architecture

Other blocks represented by figure 4.2 are a direct implementation of the data flowchart exposed in the previous section. *Data Set*, *Features Graph* and *Bootstrap* are self describing. *Project* is the top-level object containing all information related to a given georeferencing project, and can be serialized to a *JSON* file. *Model0*, *ModelTerrain* and *ModelCalibration* are three example of implementations of the hierarchical model interface described above.

Finally each class, and in particular projection functions, is exported to Python with a Cython interface. This is useful for two reasons. First, each function only has to be written once. It is exported automatically to the Python language and does not need to be rewritten. Secondly, exporting functions allows to write tests and comparison in the very concise Python language, where performance and compatibility with the optimization library is not critical.

4.3 Bootstrap

Implementing the bootstrap method requires two parameters: the size of each bootstrap sample, and how many of them to generate. In general, it is best to create identically sized samples. Therefore, for a given model to be analysed by the bootstrap method, the size of each sample is chosen to be identical to the original observation vector.

Concerning the other parameter, the number of bootstrap samples, in general more is better. However as the number of observations grows, the run time of each georeferencing increases and completing the entire bootstrap procedure can take a significant amount of time. In the current implementation the number of bootstrap samples is fixed to 100. This can be increased for better statistical significance, at the cost of running time.

After bootstrapping, given the solved model parameters, covariance matrices and histograms are computed and plotted using Python.

4.4 Digital products

The production of orthoimage via orthorectification is implemented using the back-projection method. This procedure is illustrated by figure 4.3. Pixels of the orthoimage are converted to ground points which are then projected to the cameras in which they are visible. Pixel value is obtained by interpolation of the image at the obtained image coordinate. This is made possible by the Cython export mechanism, as explained in the previous section.

The implemented orthorectification is rudimentary and does not contain all the features of a proper digital product system. For example, buildings artefacts due to perspective effect are not taken into account. Additionally, only a flat terrain is assumed to simplify the implementation. Nevertheless, using the reconstructed terrain model during the orthorectification would be a nice addition to this project.

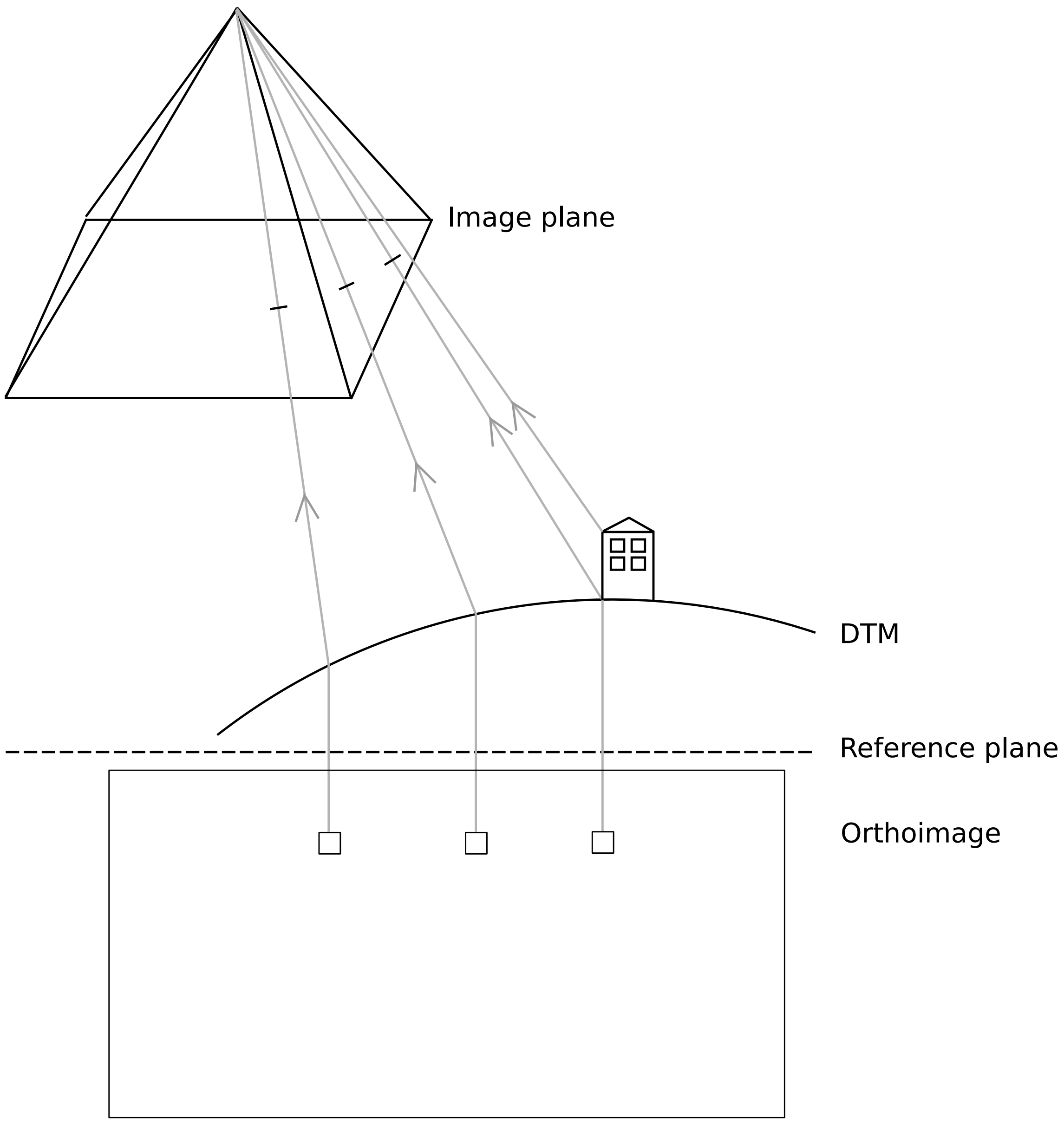


Figure 4.3. Illustration of orthorectification using back-projection, adapted from [20].

Chapter 5

Results

5.1 Feature points

As exposed in section 3.2, there are many feature matching algorithm. This section presents the work performed concerning their analysis. The main purpose is to decide which method is best suited to the task of georeferencing and terrain modeling. To achieve this, the following questions are asked:

- **Accuracy:** How accurate are the keypoints location?
- **Outliers:** How many of the matches are incorrect?
- **Spatial distribtution:** How uniform is the distribution of keypoints over the image?

This reflects the desired properties: high accuracy, high spatial coverage and low proportion of outliers. The following introduces the techniques used to quantify the above three characteristics with the ultimate purpose of selection the best solution.

5.1.1 Method

To analyse feature algorithms' properties the data has be pre-processed into simple image pairs. As illustrated by Figure 5.1, each image in a pair is from a different camera. There is therefore significant perspective change. However translation and rotation have been manually eliminated. Therefore the expected location of features is very similar in each image. This will enable the automatic detection of outliers, something that is not possible in the normal task.

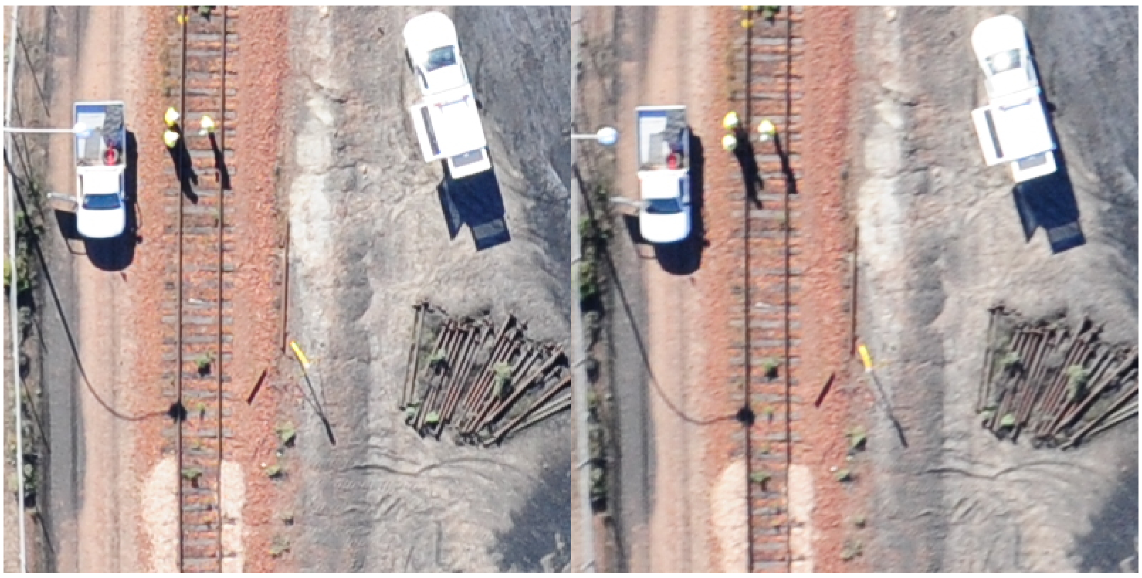


Figure 5.1. Two images of the same ground area, pre-processed for the evaluation of feature algorithms. Notice the parallax effect.

Features are matched by distances in the feature space. Then, they are ordered by distance, from lowest to highest. The general procedure is to keep the N best elements. The result is then a graph of the distance as a function of the *match number*, in other words the rank of the feature match in the distance ordering.

Next comes the question of how to evaluate the spatial distribution of features? In this sense, a good algorithm will cover most of the image. To evaluate the spatial distribution of feature points the following method is used. The input image is divided into a regular grid of c^2 cells. The number of cells is chosen such that, if the keypoints were uniformly spread over the image, there would be one per cell when all keypoints are retained, i.e.

$$c^2 = N$$

Then, a spatial coverage plot can be produced by counting the number of cells containing at least one keypoint as the number of matches increases. This is shown in figure 5.2.

Detecting outliers is difficult in actual processing. But in this constructed case, outliers will have high slope in the line connecting the matches as shown in figure 3.4. Another discriminant could also be used, but this proved sufficient in practice.

This allows generating both the angle plots and outlier plots shown in figure 5.2.

5.1.2 Results

All algorithms from the OpenCV library were integrated into this experiment. Tables 5.1 to 5.6 show the results for each image pair considered. Results include the number of matches below de 5% outlier level, and the spatial coverage as a percentage of the image.

Considering those results, SIFT appears to be the best suited solution for computing image features. Reading the tables, it is apparent that full resolution yields more features, but the extend of this is limited. Therefore it can be useful to reduce the input image size to increase performance without sacrificing features quality.

	5% outliers level	Spatial coverage (%)
SURF	803	33
KAZE	528	28
SIFT	516	16
SURF-o6	491	33
KAZE-ORB	298	24
ORB	136	11
BRISK-SIFT	49	10
AKAZE	38	6
BRISK-ORB	9	3
BRISK	9	3
BRISK-SURF	3	1
BRISK-FREAK	0	0

Table 5.1. Cars

	5% outliers level	Spatial coverage (%)
SIFT	816	33
KAZE	646	33
SURF	607	43
KAZE-ORB	556	28
SURF-o6	410	38
BRISK-SIFT	269	31
AKAZE	133	13
ORB	116	11
BRISK	92	15
BRISK-SURF	45	8
BRISK-ORB	17	3
BRISK-FREAK	0	0

Table 5.2. Desert

	5% outliers level	Spatial coverage (%)
SIFT	1818	23
KAZE	1317	18
BRISK-SIFT	698	21
SURF-o6	673	12
KAZE-ORB	312	8
BRISK	135	7
BRISK-SURF	89	5
BRISK-ORB	56	3
AKAZE	48	1
ORB	48	4
SURF	28	0
BRISK-FREAK	0	0

Table 5.3. Full image - quarter size

	5% outliers level	Spatial coverage (%)
SIFT	527	26
SURF	470	38
KAZE	428	29
SURF-o6	291	37
BRISK-SIFT	154	21
KAZE-ORB	113	12
BRISK	64	13
ORB	53	6
AKAZE	37	5
BRISK-SURF	26	6
BRISK-ORB	7	2
BRISK-FREAK	1	1

Table 5.4. Lake

	5% outliers level	Spatial coverage (%)
KAZE	11 165	14
SIFT	8337	7
BRISK-SIFT	2655	11
AKAZE	528	1
BRISK	266	2
BRISK-SURF	117	1
BRISK-ORB	38	0
ORB	32	3
SURF-o6	4	0
SURF	0	0
KAZE-ORB	0	0
BRISK-FREAK	0	0

Table 5.5. Full image - full size

	5% outliers level	Spatial coverage (%)
SURF	718	29
SIFT	602	26
KAZE	594	23
SURF-o6	556	31
BRISK-SIFT	252	25
KAZE-ORB	117	10
ORB	51	5
AKAZE	47	5
BRISK	33	6
BRISK-SURF	24	5
BRISK-ORB	6	1
BRISK-FREAK	0	0

Table 5.6. Full image - eighth size

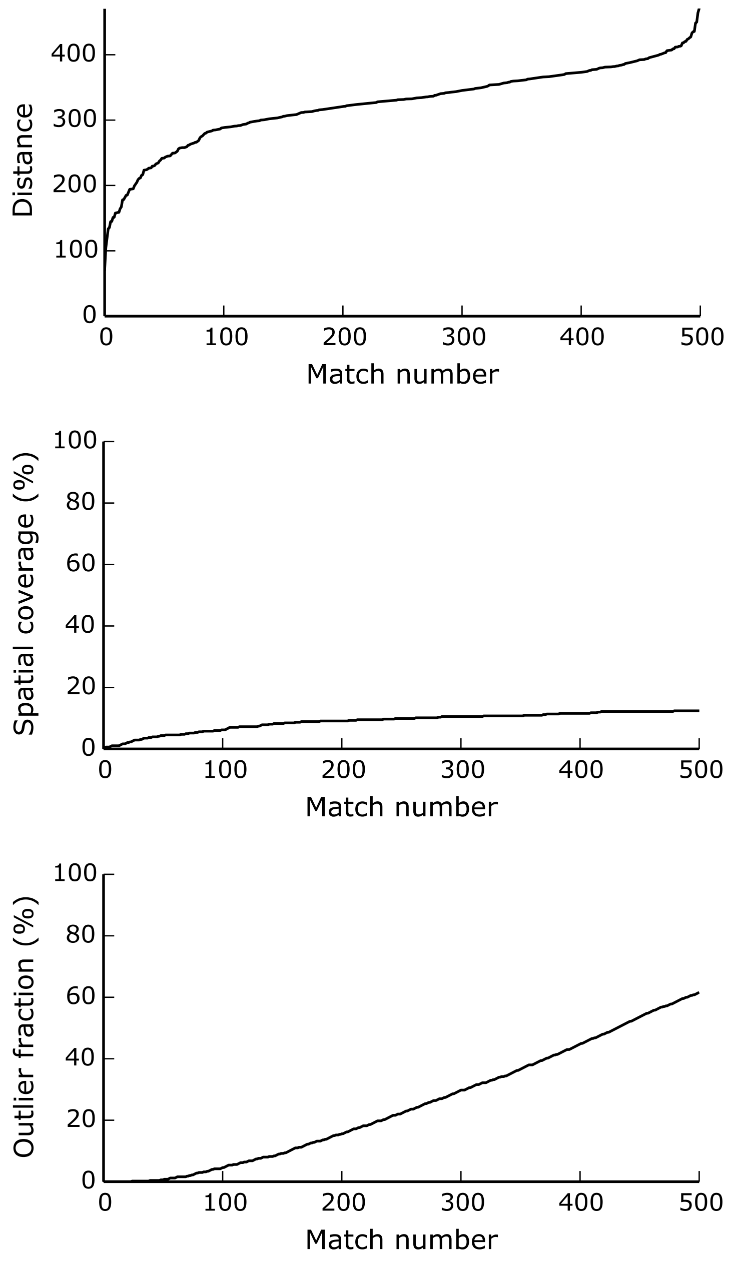


Figure 5.2. ORB – Full image

5.2 Optimization results

The next step after analysis of image features is to look at the non linear least square optimization, or bundle adjustment step. The camera projection function and image features are put together to form the reprojection error, which is then minimized by finding optimum parameters. Once the problem has been set up, this is accomplished by the optimization library Ceres-Solver. This library provides a detailed output during the optimization, as well as a final processing report. An example of this report is shown below.

```
iter      cost      cost_change  |gradient|   |step|    iter_time  total_time
   0  3.696968e-04    0.00e+00    3.01e-03   0.00e+00   6.39e-03    6.57e-03
   1  2.380916e-07    3.69e-04    1.27e-05   1.12e+02   6.65e-03    1.33e-02
   2  1.328014e-07    1.05e-07    1.76e-05   2.04e+01   6.58e-03    1.99e-02
   3  1.109043e-07    2.19e-08    5.33e-06   1.05e+01   6.55e-03    2.64e-02
   4  1.090832e-07    1.82e-09    5.97e-07   3.22e+00   6.53e-03    3.30e-02
   5  1.090626e-07    2.06e-11    7.01e-09   3.52e-01   6.59e-03    3.96e-02

Solver Summary (v 1.10.0-lapack-suitesparse-openmp)

                                     Original                  Reduced
Parameter blocks                           12                       11
Parameters                                 32                       26
Residual blocks                            20                       20
Residual                                   40                       40

Minimizer                        TRUST_REGION

Dense linear algebra library            EIGEN
Trust region strategy     LEVENBERG_MARQUARDT

                                        Given                     Used
Linear solver                     DENSE_SCHUR              DENSE_SCHUR
Threads                                     1                        1
Linear solver threads                       1                        1
Linear solver ordering              AUTOMATIC                    10, 1

Cost:
Initial                          3.696968e-04
Final                            1.090626e-07
Change                           3.695877e-04

Minimizer iterations                        5
Successful steps                            5
Unsuccessful steps                          0
```

```
Time (in seconds):
Preprocessor                          0.0002

  Residual evaluation                 0.0008
  Jacobian evaluation                 0.0382
  Linear solver                       0.0004
Minimizer                             0.0411

Postprocessor                         0.0000
Total                                 0.0413

Termination:                     CONVERGENCE
(Function tolerance reached. |cost_change|/cost: 3.318589e-07 <= 1.000000e-06)
```

The first piece of information is the convergence behavior. In five iteration steps, the solver finds the optimum point and stops. But it is also crucial to note how the cost function value diminishes rapidly, and reaches its minimum in practically two steps. This is very rapid and benefits both speed and reconstruction accuracy.

Next, is the termination condition. The key *convergence* indicates that the optimization stopped because the absolute or relative function cost tolerance has been reached. Termination can also occur when a given tolerance is reached on the parameters value, but it is not the case here.

5.3 Orthoimages

Figures 5.3 and 5.4 show orthorectified images produced using the parameters found by the optimization procedure. The first application of these images is the visual inspecting of the results' quality. Of particular interest is the overlap of the two images into one flat projection. Roads are well connected together and structures are continuously visible. However, radiometric differences are evident at the border of images. This is due to images having been acquired at different angles, therefore different reflection angles cause slightly varying luminance levels. This can be dealt with with a better reprojection system taking into account advanced radiometric corrections.

Image keypoints are shown in red in figures 5.3 and 5.4. A black line segment between two points indicates that they correspond to the same object point. Therefore in the ideal case all keypoints would overlap. This is a simple method to visualize residuals of the optimization process.

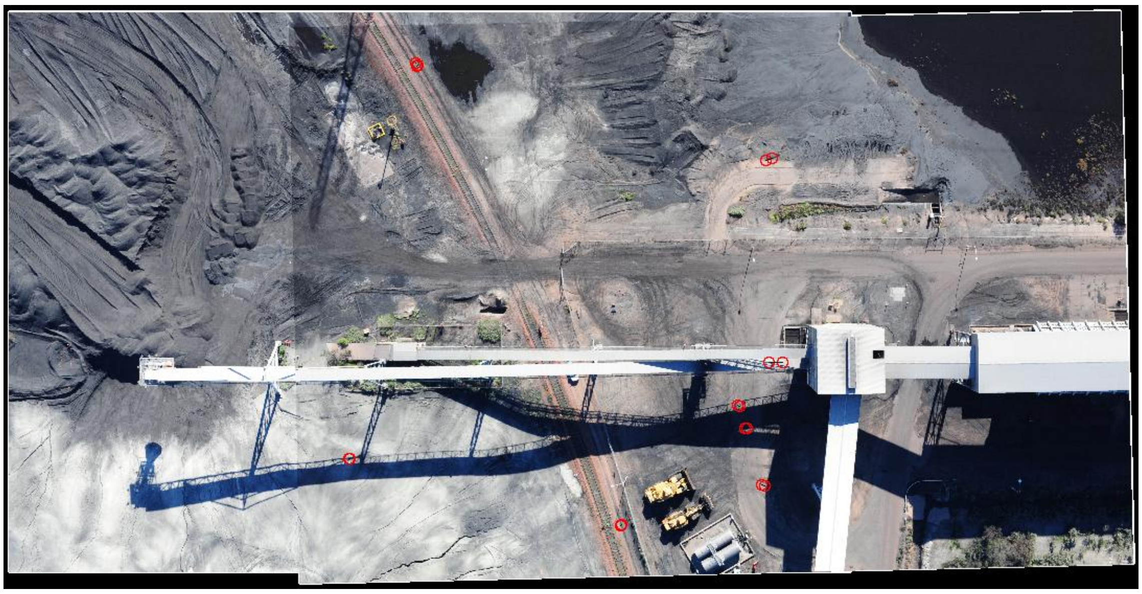


Figure 5.3. Orthoimage with low density features

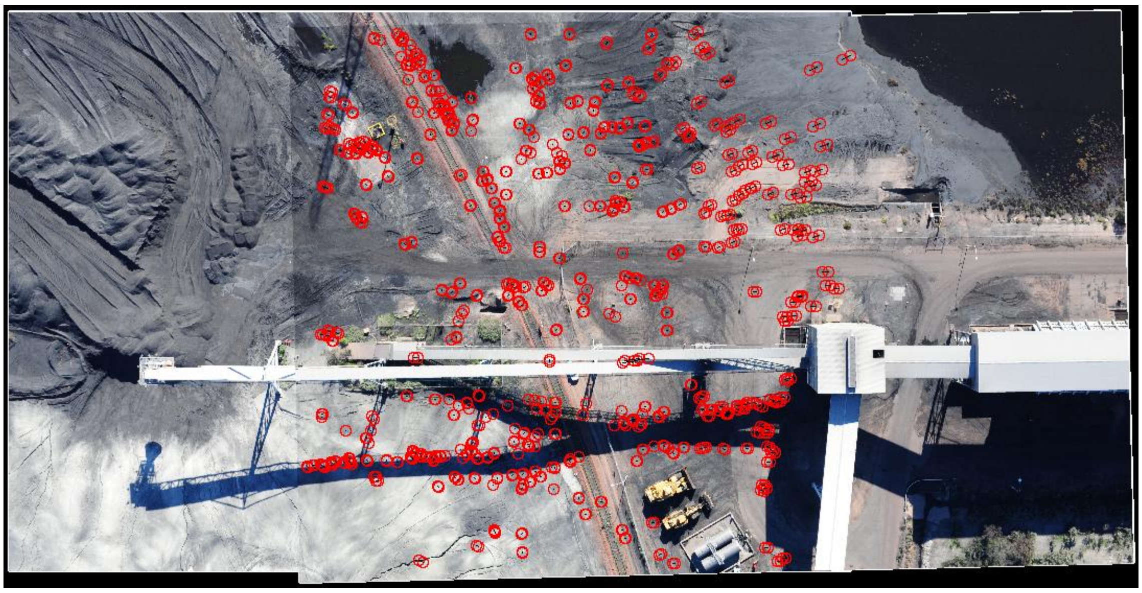


Figure 5.4. Orthoimage with high density features and fixed cameras

In this example there are no outliers. Keypoints are selected by keeping only a fixed number of the best ones. However, even with this method, outliers can be generated and there is a risk of contamination of the final result.

Outliers are caused primarily by incorrect feature matches. This is a huge problem because an outlier will cause the reprojection error to grow very large and be impossible to minimize. If few outliers are present, the reconstruction accuracy will be hurt. But in the worse case, too many outlier will render the optimization impossible and the result will be completely incorrect. Over-parametrization can have a similar effect on the result, where the final parameter values are not at the global optimum.

The difference between figures 5.3 and 5.4 is the density of image features. The first figure is produced by selecting only ten points and optimizing the camera positions. On the other hand, the second image contains two hundred features points, while the camera positions are maintained fixed. This is the most basic method used to obtain a high density terrain model in the form of a point cloud. Even higher density point clouds can be obtained using epipolar geometry constraints or dividing images in select patches to obtain more keypoints.

Physically accurate orthoimages require knowing the terrain model accurately. This is necessary to take into account terrain curvature and shape, as well as perspective effects due to vertical structures, in buildings for examples. Figures 5.3 and 5.4 use orthorectification on a ground plane assumed flat. This is not satisfactory for high quality products but is acceptable for the purpose of this project. Hence, some sources of error in those figures come also from the imperfect orthorectification procedure.

Orthoimage are a powerful tool for qualitative and visual inspection of the results. However the need for a more quantitative mean of analysis becomes apparent as the problem size grows. This will be accomplished using the bootstrap method to provide stability analysis of the reconstructed parameters.

5.4 DTM

From the results of the numerical optimization procedure, precise camera positions and world point coordinates are obtained. The world point correspond to points on the ground that have been identified in images and used for reconstructing the problem geometry. Those points can then be extracted together with the image color values to produce a 3D point cloud representing the terrain. Figure 5.5 shows such a point cloud in the 3D visualization tool *Potree*.

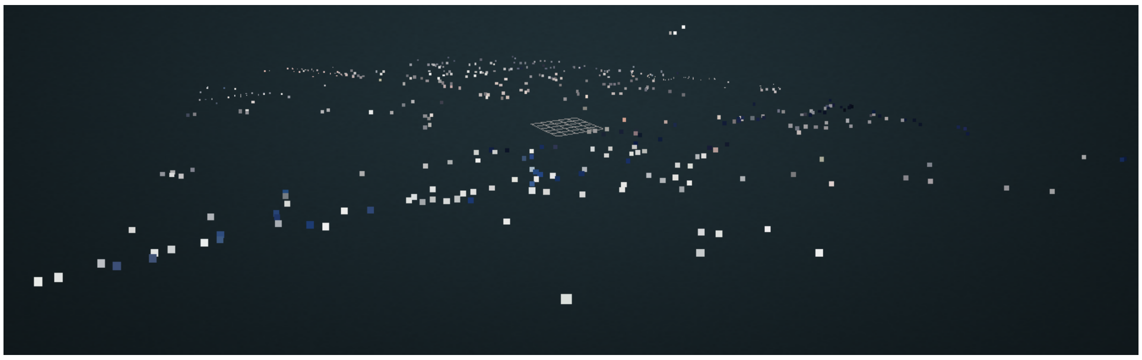


Figure 5.5. Potree

This web based tool allows the display and navigation of large 3D point clouds, together with an associated color value to simulate the environment. The user can move, orbit and zoom interactively while viewing the point cloud. Figure 5.5 shows a low density point cloud with no interpolation. However a typical photogrammetry product will use a very high density point cloud together with interpolation to emulate the aspects of a continuous surface. This section merely demonstrates the implementation of the complete processing chain, from images to a 3D point cloud.

This visualization also comes with the advantage of highlighting errors by visual inspection. Outliers for example become very apparent as artefacts in the product quality. Therefore the need for quality control is made apparent. On the other hand, minimizing user input and user interaction is also a goal to reduce cost and production efficiency. Combining those the goals of efficiency and a high quality product is a serious challenge in designing georeferencing systems. This 3D point cloud rendering tool is very valuable in that respect. It allows the systematic evaluation of different approaches and their respective quality.

5.5 Stability analysis

Analysis of the results' stability is necessary to know the robustness of the method, its sensibility to observations, the precision of the results and correlations of parameters. Stability analysis is also a useful insight into the solution's uncertainty, notably which parameters are more precisely reconstructed along different dimensions.

In this project, stability analysis is implemented with the bootstrap method. The bootstrap procedure re-samples the available observations many times and then georeferences the images based on this new bootstrap sample to produce a result distribution. From this, empirical parameters of the results distribution can be estimated, most notably the covariance matrix and histograms.

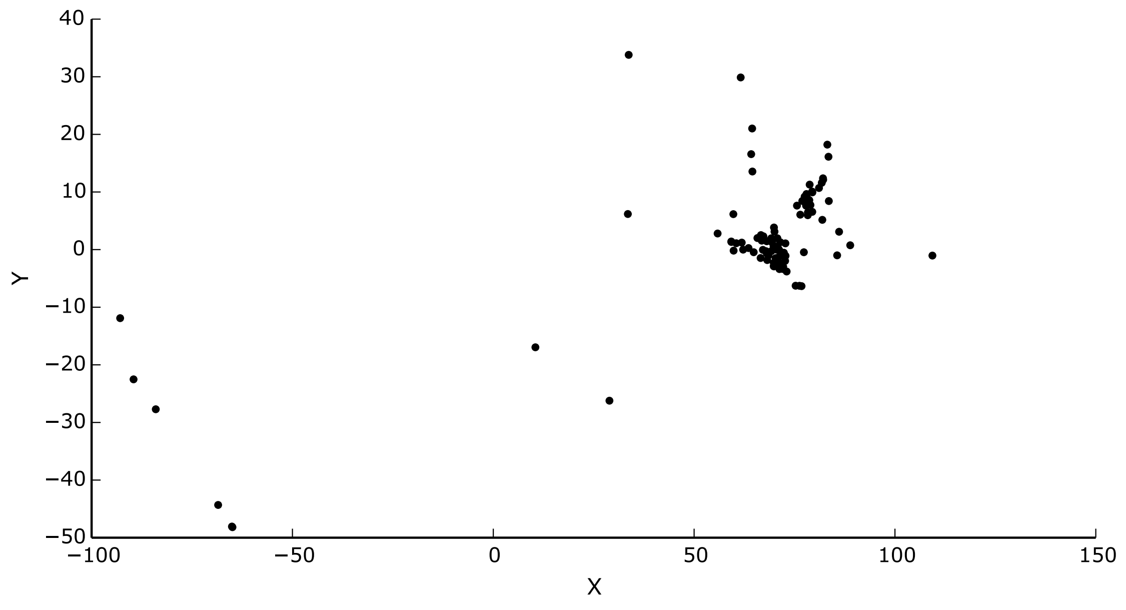


Figure 5.6. 2D density map of camera horizontal position from bootstrap analysis

Figure 5.6 shows the 2D density map of X and Y coordinates of the right camera in the initial georeferencing. The mode is clearly visible around $X = 70m, Y = 0m$, corresponding to figure 5.3. Most georeferencing procedures yield an accurate camera position, however for some bootstrap samples the result is incorrect by up to 100 meters. Such errors are not observed outside of the stability analysis. Therefore it is probably the result of poorly distributed observations resulting from the resampling with replacement inherent to the bootstrap method. This highlights the need for well distributed observations over the image space.

Next, figure 5.7 shows the distribution of the altitude parameter in the reconstructed camera position vector. As expected, depth uncertainty is rather high. This is inherent to the geometry of the problem, because measurement rays are essentially bearing only and altitude information is reconstructed indirectly by triangulation.

Finally, figure 5.8 shows the visualization of the covariance matrices of the results. Correlation coefficients between random variables are shown in a traditional matrix layout, but additionally the squares areas are proportional to the correlation.

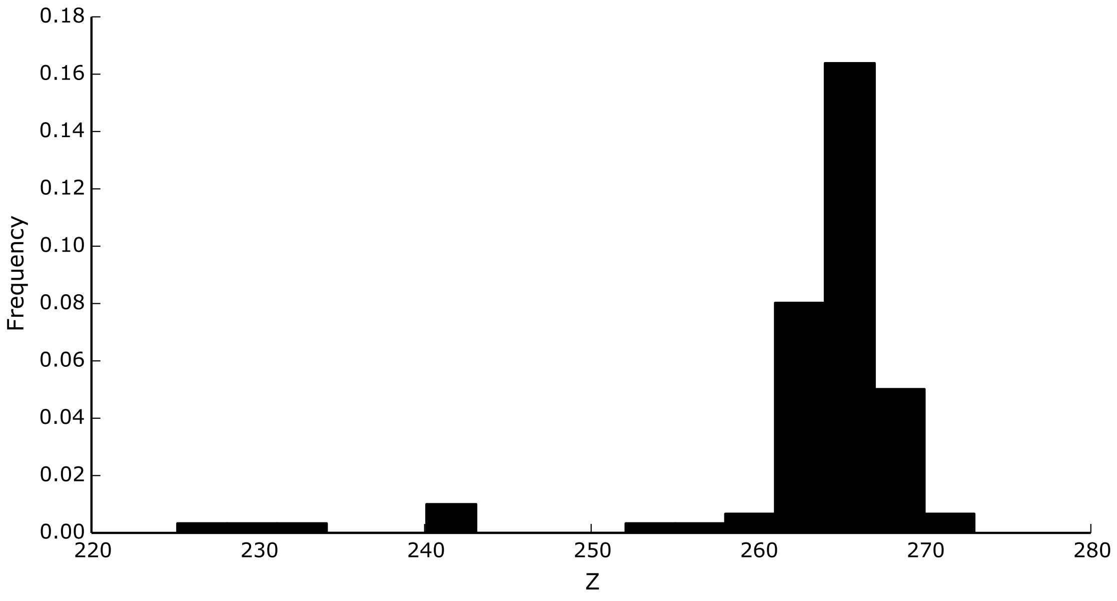
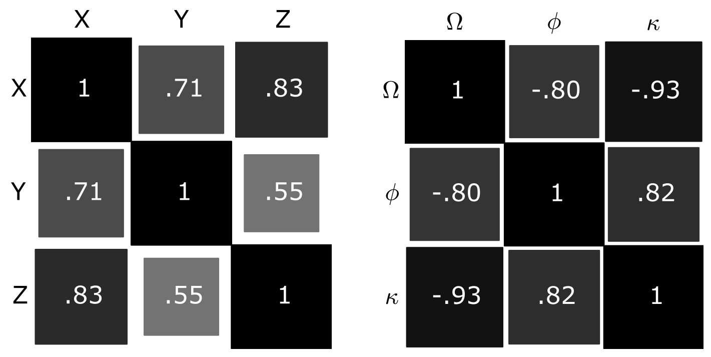


Figure 5.7. Histogram of reconstructed camera altitude from bootstrap analysis



	X	Y	Z
X	1	.71	.83
Y	.71	1	.55
Z	.83	.55	1

	Ω	ϕ	κ
Ω	1	-.80	-.93
ϕ	-.80	1	.82
κ	-.93	.82	1

Figure 5.8. Covariance matrix visualization of reconstructed parameters

5.6 Implementation performance

This section reviews the performance of each step of the processing pipeline. To develop a complete georeferencing system, it is necessary to know how computationally expensive is each step. Secondly, it is also important to understand how timing scales with input size. The experiments in this project are performed with few images, but a real georeferencing system will work with hundreds of images.

Looking at tables 5.7 to 5.10 shows the timing of each step in the processing pipiline for different solving configurations. Computing image features, and orthorectification is the most time consuming processing step. While it depends on the target resolution, orthorectification is expected to take a long time as it involves a complete resampling of the image. Additionally, the implementation of orthorectification is not the focus of this project and therefore no particular effort has been made in that respect.

Computing the image feature points is time consuming mostly due to the fact that the best of them are selected. This involved computing a large amount of points, rating them, and only then selecting the best ones. However, this costly step can effectively be improved by reducing the image input scale. This does not inccur a significant reduction in accuracy but dramatically reduces processing time, as shown by tables 5.9 and 5.10. While a coarse adjustment of the camera positions does not require full resolution images, they might be necessary for generating a high density point cloud DTM. It should also be noted that pairwise image overlap is important to the performance of the features step. Indeed, if no preprocessing is done to detect which images overlap, the number of feature vectors to be computed will be quadratic in the number of images. This is in practice reduced to a linear relationship when images overlap in fixed numbers, which is the case for most flights.

Secondly comes the bundle adjustment step, where the numerical optimization happens. It is in most cases very performant. This is largely due to using a high performance library written in C++. This numerical optimization library uses special code for bundle adjustment types of problems which take advantage of the problem structure. This means that this step is not only fast, but also scales well with the

Step	Time (seconds)
Features	61.51
Bundle adjustment	0.79
Bootstrap	5.85
Orthoimage	38.46
DTM	1.16

Table 5.7. Time of each step in the pipeline, 10 feature points.

Step	Time (seconds)
Features	61.95
Bundle adjustment	1.67
Bootstrap	89.55
Orthoimage	40.05
DTM	1.09

Table 5.8. Time of each step in the pipeline, 200 feature points

Step	Time (seconds)
Features	16.37
Bundle adjustment	0.78
Bootstrap	5.43
Orthoimage	35.91
DTM	0.99

Table 5.9. Time of each step in the pipeline, half resolution input

Step	Time (seconds)
Features	10.34
Bundle adjustment	1.19
Bootstrap	12.74
Orthoimage	52.52
DTM	1.38

Table 5.10. Time of each step in the pipeline, quarter resolution input

number of input images. The challenge is designing a good georeferencing system is then not in a performant optimization step, but rather in the problem structure to avoid errors in the final product.

The timing of the bootstrap analysis step is directly proportional to the bundle adjustment step. This is expected since the bootstrap essentially solves a given model for each bootstrap sample. Controlling the performance of the bootstrap involves therefore a tradeoff between quality of the resulting stability analysis and processing time. However this is only done when analysing a system, not when producing delivarables. Therefore it is a good idea to invest time in this step and obtain good data to interpret.

The last step is the generation of the terrain model point cloud. This is a relatively straightforward operation that involves resampling the images at the location of the adjusted ground points. Therefore the difficulty of dealing with 3D point clouds is usually in the rendering and user interface, not the data processing.

5.7 Discussion

5.7.1 Problem design

A critical issue when building a georeferencing system is that of designing the problem. Designing the problem refers to choosing a parameter system to represent cameras and 3D points, choosing a projection function, a cost function, an robustness method to deal with outliers, priors for the parameters distributions, which parameter blocks are constant or variable, etc. There are many issues explored in this project and they all relate to each other. This section is an attempt at putting it all together and providing an overview or recommendations relating to problem design.

- **Camera parameters.** The number of degrees of freedom must be limited to the necessary, and over parametrization avoided. But it is essential that all possible effects are modeled. Quaternions can be used for computing rotations if singularities occur. Internal camera parameters can be adjusted as part of the optimization procedure, but this comes at a risk of over-parametrization. In most cases, assuming a fixed focal length will provide acceptable results, which will at worse be correct up to a scale factor.
- **Cost function** A multivariate normal distribution transforms the problem into the familiar non linear least squares. This works fine in theory, but can be problematic in the presence of outliers. A robustification function can be added in that case.

- **Priors** Priors can help and are easy to add in the maximum aposteriori framework, but they are not necessary. If the georeferencing does work, they will not be the missing element to the problem.
- **Hierarchical modeling** Hierarchical modeling helps tremendously with both accuracy, solvability and processing speed. It allows getting an answer fast and then refining the result within local bounds. Implementing this approach was a significant part of this project's work. It has proven successful and is highly recommended.
- **Optimization** Numerical optimization is a difficult topic, but thankfully also mature. Using a robust and existing toolkit is highly recommended. Automatic differentiation is very convenient for rapid prototyping and also simplicity. Using automatic differentiation does not incur any performance cost and is therefore also recommended.
- **Calibration vs auto-calibration** If camera calibration information is available, it should be used. The absence of calibration data will reduce the quality of the result but not render it impossible. In that case, some internal parameters can be assumed at a fixed value, while others can be adjusted together with external parameters. This is called auto-calibration. It increases the risk of divergence due to over-parametrization, but is a useful alternative when no calibration information is available.

5.7.2 Recommendations

This project investigated many alternatives and solutions. On top of the earlier results and analysis, here are some general recommendations concerning the design of a complete georeferencing and terrain modeling system.

- Use SIFT features. Compared to all other available image feature algorithms in OpenCV, SIFT seems to be the best performing with respect to performance, proportion of outliers and spatial coverage.
- Adapt the number of image features to the optimization problem. Use few points for adjusting camera positions only, and a higher number for terrain modeling. However a high density terrain model will require specific algorithms on top of a high number of feature points.
- Use GPS information for the initial guess of camera positions. If no prior GPS information is available, the alternative is to use absolute orientation. However this is usually costly for large data sets, and introduces a new source of errors and therefore a new need for time consuming quality control. Additionally, the initial guess needs to be close enough to the optimal solution to guarantee

convergence. If a solution cannot be found, a good solution can be to upgrade the GPS system or reduce the model complexity.

- Ensure the even distribution of image features over the images. Poor spatial coverage will introduce errors in the georeferencing procedure, in the form of some areas of the image being adjusted more precisely than others.

- Ensure the presence of enough features per images, and roughly the same number per image. This was shown by the bootstrap analysis. Some bootstrap samples generate largely incorrect results, while this was not observed in the normal procedure. This is related to the classical overlap parameter in photogrammetry. In general, aiming for enough overlap so that objects are visible in at least three images is a good target.

- Use an existing and robust numerical optimization library. Numerical optimization is a very active area of research and development. There is no reason to develop a custom system when well tested and performing solutions are available. The library used in this project for example, Ceres-Solver, comes with optimizations specific to Bundle Adjustment, which increases performance over a generic non linear least squares.

- Carefully design code handling units, reference frames and coordinate systems. In particular, leveraging good engineering practices and the compiler's type system is recommended.

- A high quality orthoimage requires a high quality DTM for projecting the images onto actual terrain. Otherwise artifacts of incorrect assumptions will be visible in the details.

- A high quality orthoimage also requires radiometric correction. Otherwise differences in exposure and solar angle will be visible at the edges and borders of images.

- Reduce the scale of the input images for the first step of computing feature points. Aerial images are usually high resolution, but reducing their size by a factor of two or four usually has no impact of the quality of feature points. This is particularly recommended for the first pass of optimization where cameras are initially very far off.

- Dealing with the possibility of outliers in image features is a critical part of designing a georeferencing system. Two things are important. First, choose an image features algorithm and its parameters so that the proportion of outliers is minimized. This depends on the type and content of the images and needs to be an appropriate and informed choice. Secondly, carefully model the distribution of the inlier and outlier points. This is explained in details in [27]. Basically, the normal distribution is very sensitive to outliers due to its very small tails. The possible techniques are using another distribution such as the multivariate Cauchy distribution, or using a robustification function over the least square problem. Outliers happen when too many features are requested per images, the image content is to difficult to work with for the algorithm, image pairwise relations are incorrect or image have little to no overlap.

- Finally, the integration of user driven quality control into the pipeline is to be considered. While care can be taken to minimize the need for quality control, large georeferencing problems will eventually require manual corrections and input. Therefore anticipating the need for user interaction is recommended, rather than dealing with it reluctantly.

5.7.3 Future work

This project explored many issues concerning georeferencing and terrain modeling. However time was a limiting factor and much future work on the subject remains.

Concerning research, better feature algorithms are always a possibility. In particular one could imaging deriving a method specially for the use case of aerial images, which would produce keypoints of high quality and could rely on additional assumptions. This is always an active area of research in the computer vision community. Numerical optimization research is also going strong. More knowledge in issue specific to bundle adjustment would be very useful, especially situations where uncertainty and under-determined problems arise. Finally, existing bundle adjustment literature seems difficult to understand for researchers outside the field. Therefore more accessible material in the form of books or library documentation would be useful for wide spread adoption of the methods. Nonetheless the Ceres-Solver library documentation is excellent and deserve a mention as it can serve as a suitable introduction to georeferencing.

Regarding implementation, the only complete and usable georeferencing solutions are commercial solutions. This is understandable considering the large amount of research and development involved in such a piece of software. In this thesis as well, implementation was an integral part of the work. Thankfully mature and open source libraries exist for bundle adjustment type of optimization problems. Computer vision libraries are dominated by OpenCV and a few others. However documentation can be confusing at times, and clear description of algorithms, image features in particular, is not always available. Therefore more robust and well tested libraries in computer vision would have a positive impact, both for feature algorithm and product quality.

Finally, more experiments similar to the ones conducted in this project are needed. Large data set (more than 100 images) were not experimented with due to time constraints. Large data set are useful for exposing time scaling issues, as well as performance bottlenecks. High density terrain modeling is also an integral part of producing digital products. It is usually provided as part of aerial images processing systems but the implementation is a large and independent task. This is also true with respect to orthorectification and projection onto accurate terrain.

In terms of analysis, evaluation of accuracy and stability could be taken further by using other methods. Leave-one-out error for cross validation, the rank of the Jacobian matrix for checking if the problem is under-determined and DTM interpolation are all areas of very interesting future work.

Chapter 6

Conclusion

Georeferencing and terrain modeling are heavy data processing tasks when applied to airborne images. This project effectively presented and investigated key aspects of this pipeline.

A preliminary study of image features was carried out to select SIFT as the most suited image features algorithm for georeferencing. Then, the theory of the method was exposed. This involves derivation of the maximum aposteriori estimator and how it fits within the chosen numerical optimization framework. The bootstrap analysis was also explained with an emphasis on its interpretation.

Then, implementation of a complete georeferencing and terrain modeling system was carried out. It supports hierarchical modeling and different problem parametrisation, which was extremely useful in analysing and comparing different approaches. This was also interesting because the inherent complexity of the system requires important software engineering considerations. This constitutes an original contribution and also gives insight into the necessary work for a complete solution.

In this sense, working with real world data was very valuable. The results and interpretation of the proposed and implemented method is much more robust and justifiable when using real data.

Furthermore, analysis of the results provided by the image features study, the georeferencing procedures and the bootstrap resampling was exposed. This concerns the results of optimization, the production of orthoimages and DTM, the stability of the results using the bootstrap method and the performance of the implementation.

Finally, and perhaps of greater interest, are the discussion section and recommendations. Recommendations come from the experience gained while realizing this thesis. While constrained by time, the implementation is complete and contains all functionality, but it is not usable as a software product. Therefore the discussion section is perhaps the most valuable of this project as guidelines for another future project to produce a more user friendly piece of software.

Bibliography

[1] `http://json.org`.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. "Building rome in a day". In: *Communications of the ACM* 54.10 (2011), pp. 105–112. URL: `http://dl.acm.org/citation.cfm?id=2001293` (visited on 06/04/2014).

[3] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver.* URL: `http://ceres-solver.org`.

[4] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. "Censure: Center surround extremas for realtime feature detection and matching". In: *Computer Vision–ECCV 2008.* Springer, 2008, pp. 102–115.

[5] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. "Freak: Fast retina keypoint". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* Ieee, 2012, pp. 510–517.

[6] Pablo F Alcantarilla and TrueVision Solutions. "Fast explicit diffusion for accelerated features in nonlinear scale spaces". In: *IEEE Trans. Patt. Anal. Mach. Intell* 34.7 (2011), pp. 1281–1298.

[7] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. "KAZE features". In: *Computer Vision–ECCV 2012.* Springer, 2012, pp. 214–227.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "Speeded-Up Robust Features (SURF)". en. In: *Computer Vision and Image Understanding* 110.3 (June 2008), pp. 346–359. ISSN: 10773142. DOI: `10.1016/j.cviu.2007.09.014`. URL: `http://linkinghub.elsevier.com/retrieve/pii/S1077314207001555` (visited on 06/06/2014).

[9] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. "Brief: Binary robust independent elementary features". In: *Computer Vision–ECCV 2010.* Springer, 2010, pp. 778–792.

[10] C BROWN Duane. "Close-range camera calibration". In: *Photogram. Eng. Remote Sens* 37 (1971), pp. 855–866.

[11] Chris Harris and Mike Stephens. "A combined corner and edge detector." In: *Alvey vision conference.* Vol. 15. Manchester, UK, 1988, p. 50.

[12] Johannes Kopf, Michael F Cohen, and Richard Szeliski. "First-person hyperlapse videos". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 78.

[13] Walter G. Kropatsch, Fuensanta Torres, and Geetha Ramachandran. "Constrained Bundle Adjustment for Panoramic Cameras". In: (). URL: `ftp://cmp.felk.cvut.cz/pub/cvl/articles/alblcene/Albl-Pajdla-CVWW-2013.pdf` (visited on 06/07/2014).

[14] Sbastien Leprince, Sylvain Barbot, Franois Ayoub, and Jean-Philippe Avouac. "Automatic and Precise Orthorectification, Coregistration, and Subpixel Correlation of Satellite Images, Application to Ground Deformation Measurements". In: *IEEE Transactions on Geoscience and Remote Sensing* 45.6 (June 2007), pp. 1529–1558. ISSN: 0196-2892. DOI: `10.1109/TGRS.2006.888937`. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4215064` (visited on 06/04/2014).

[15] Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. "BRISK: Binary robust invariant scalable keypoints". In: *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2548–2555.

[16] Manolis Lourakis. "Bundle adjustment gone public". In: (). URL: `http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf` (visited on 05/02/2014).

[17] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[18] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Image and vision computing* 22.10 (2004), pp. 761–767.

[19] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M. Newman, and Ian D. Reid. "A Constant-Time Efficient Stereo SLAM System." In: *BMVC.* 2009, pp. 1–11. URL: `http://www.robots.ox.ac.uk/~cmei/articles/AConstantTimeEfficientSt` `rss_09.pdf` (visited on 05/02/2014).

[20] Edward M Mikhail, James S Bethel, and J Chris McGlone. *Introduction to modern photogrammetry.* Vol. 1. John Wiley & Sons Inc, 2001.

[21] Marius Muja and David G Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." In: *VISAPP (1)* 2 (2009).

[22] Timm Ohlhof, Oliver Montenbruck, and Eberhard Gill. "New approach for combined bundle block adjustment and orbit determination based on Mars-94 three-line scanner imagery and radio-tracking data". In: *Spatial Information from Digital Photogrammetry and Computer Vision: ISPRS Commission III Symposium.* International Society for Optics and Photonics, 1994, pp. 630–639. URL: `http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=982576` (visited on 05/02/2014).

[23] Richard Roberts, Sudipta N. Sinha, Richard Szeliski, and Drew Steedly. “Structure from motion for scenes with large duplicate structures”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3137–3144. URL: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995549` (visited on 06/07/2014).

[24] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[25] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. “ORB: an efficient alternative to SIFT or SURF”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571. URL: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126544` (visited on 03/12/2015).

[26] Thierry Toutin. “Block bundle adjustment of Landsat 7 ETM+ images over mountainous areas”. In: *Photogrammetric engineering and remote sensing* 69.12 (2003), pp. 1341–1350. URL: `http://info.asprs.org/publications/pers/2003journal/december/2003_dec_1341-1349.pdf` (visited on 05/27/2014).

[27] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. “Bundle Adjustment — A Modern Synthesis”. en. In: *Vision Algorithms: Theory and Practice*. Ed. by Bill Triggs, Andrew Zisserman, and Richard Szeliski. Lecture Notes in Computer Science 1883. Springer Berlin Heidelberg, Jan. 2000, pp. 298–372. ISBN: 978-3-540-67973-8, 978-3-540-44480-0. URL: `http://link.springer.com/chapter/10.1007/3-540-44480-7_21` (visited on 05/02/2014).

[28] Wikipedia. `https://en.wikipedia.org/wiki/Pinhole_camera_model`.

[29] Wikipedia. `https://en.wikipedia.org/wiki/Distortion_%28optics%29`.

[30] Changchang Wu. “Towards linear-time incremental structure from motion”. In: *3D Vision-3DV 2013, 2013 International Conference on*. IEEE, 2013, pp. 127–134. URL: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6599068` (visited on 06/08/2015).

[31] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz. “Multicore bundle adjustment”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3057–3064. URL: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995552` (visited on 06/04/2014).

DTU Space
National Space Institute
Technical University of Denmark

Elektrovej, building 327
DK - 2800 Kgs. Lyngby
Tel (+45) 4525 9500
Fax (+45) 4525 9575

www.space.dtu.dk