Graz University of Technology

Institute for Computer Graphics and Vision

Master's Thesis

Automated Georeferencing and Guided Camera Orientation for High-Resolution Aerial Imagery

---

Martin Öttl

Graz, Austria, November 2013

Thesis supervisors

Univ. Prof. DI Dr. Horst Bischof

DI Dr. Matthias Rüther

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Abstract

This work describes the design of a state of the art Aerial Triangulation (AT) pipeline. The distinct stages are implemented with the goal of a fast processing speed by exploiting as many information as available, which is crucial for dealing with high resolution aerial images. Such prior information are camera poses obtained from Global Positioning System (GPS) and Inertial Navigation System (INS) measurements as well as rough scene approximations from Digital Elevation Models (DEMs). These terrain data are freely available and cover nearly the entire earth. At the beginning of the pipeline Scale-Invariant Feature Transform (SIFT) features are extracted, then the view selection stage utilizes the afore mentioned prior information to decide, which images display the same part of the scene and thus should be matched. During the subsequent matching step the same information is applied to predict corresponding feature locations in an image given the location in another image. After the triangulation of the obtained feature correspondences the final bundle adjustment stage refines the 3D structure as well as the camera poses. Evaluation results on two different image sets are presented, where one set contains also oblige images.

Keywords. Aerial Triangulation, Bundle Adjustment, Feature Matching, Georeferencing

Kurzfassung

Diese Arbeit beschreibt das Design einer dem Stand der Technik entsprechenden Pipeline für Aerotriangulation. Die einzelnen Stufen sind mit dem Ziel einer schnellen Abarbeitung unter Ausnutzung aller verfügbaren Informationen entworfen worden, dies ist besonders bei Verwendung von hochauflösenden Luftbildaufnahmen wichtig. Solch verfügbares Vorwissen besteht aus Global Positioning System (`GPS`) und Inertial Navigation System (`INS`) Messungen der Kamera Posen sowie aus einer ungefähren Näherung der Szene mittels Digital Elevation Model (`DEM`). Diese Geländedaten sind frei erhältlich und decken fast die ganze Erde ab. Am Beginn der Pipeline werden Scale-Invariant Feature Transform (`SIFT`) Features extrahiert, anschließend nutzt der View Selection Schritt das zuvor erwähnte Vorwissen um zu entscheiden, welche Bilder den gleichen Teil der Szene zeigen und deshalb im Matching Schritt auf übereinstimmende Features untersucht werden sollen. Während des nun anschließenden Matching Schritts wird die gleiche Information dazu verwendet um für gegebene Feature Positionen in einem Bild korrespondierende Feature Positionen in einem anderen Bild zu schätzen. Nach erfolgter Triangulierung der ermittelten Feature Korrespondenzen verbessert eine abschließende Bündelausgleichung die 3D Struktur und die Kamera Posen. Es werden bei zwei verschiedenen Luftbild Datensätzen die erzielten Ergebnisse aufgeführt, wobei ein Datensatz auch schräg aufgenommene Bilder enthält.

Schlagwörter. Aerotriangulation, Bündelausgleichung, Feature Matching, Georeferenzierung

Acknowledgments

I would like to thank Univ. Prof. DI Dr. Horst Bischof and DI Dr. Matthias Rüther for giving me the opportunity to work within this interesting area and supporting this thesis with a research grant. In particular, I want to express my gratitude to DI Dr. Matthias Rüther for providing the used aerial imagery, his superb support and valuable feedback.

Then I also would like to thank DI Markus Rumpler for a introduction on the already existing Structure from Motion framework, developed at the Institute for Computer Graphics and Vision. Furthermore he helped me in setting up the therefore necessary build environment.

Finally I want to thank my colleague Roman Zeleznik for his tests and feedback on the implemented pipeline. Additionally he supported me by generating test datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A general prerequisite for the generation of many Geographic Information System (GIS) products like orthophotos, surface models of cities, etc. are accurately georeferenced camera poses [1, 2]. Direct georeferencing uses very accurate Global Positioning System (GPS) and Inertial Navigation System (INS) measurements to determine these camera poses [3]. In contrast Aerial Triangulation (AT) can produce a georeferenced output with missing or less accurate measurements by refining the given or estimated camera positions and orientations during a bundle adjustment stage [4]. This work concentrates on the case that imprecise pose measurements are available, which have to be refined for a further processing. For that purpose an AT pipeline is designed. As side product a sparse scene reconstruction is obtained.

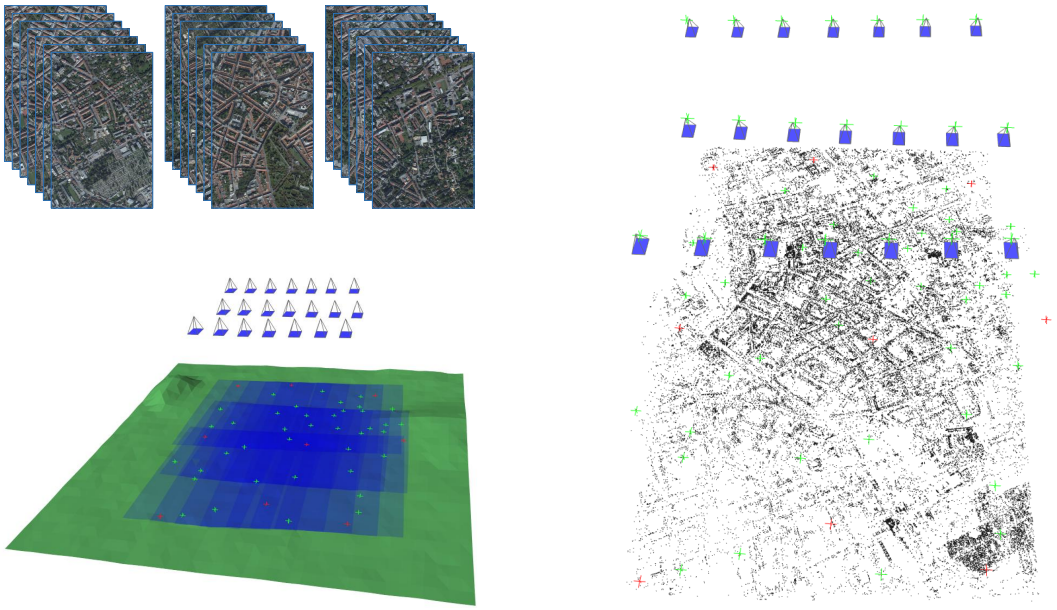
Another application of bundle adjustment is camera calibration. On the one hand the intrinsic camera parameters as well as lens distortion coefficients can be computed [5]. On the other hand the relative offset and orientation between pose measurement device and camera can be determined, which are known as lever arm offset and boresight misalignment rotation respectively [6]. Thus bundle adjustment may be also required for direct georeferencing.

For performing AT, feature points are detected in each aerial image and afterwards matched. Without any further measures one has to match each aerial image with all others. For example let's consider an aerial survey of a quadratic shaped area consisting of 1000 images obtained with 80% forward overlap and 60% side-lap. When each image is matched against all others, then $999 \cdot 1000 \ / \ 2 = 499500$ image pairs are processed in the feature matching stage. However due to the layout of the survey it is known, that a image in the center of the region overlaps with 44 other images. Therefore actually only

$44 \cdot 1000 \ / \ 2 = 22000$ image pairs have to be matched, if the border of the aerial survey is not considered. This results in a saving of more than 95% of the effort compared to the naive approach. As demonstrated, the naive approach has a complexity of $O(n^2)$ in the number of images, while the other strategy has a linear complexity. So the savings will be even larger for larger image sets. The calculation of overlapping images for feature matching is referred to as view selection within this work and essential for a reasonable performance.

Some aerial camera systems produce large scale images, for example the Microsoft Super-Large UltraCam generates images with a size of 20000 pixel $\times$ 12900 pixel [7]. As a result fewer flight strips are required to cover a certain terrain and therefore the flight costs are reduced. Depending on the interest point detector parametrization one obtains usually a huge amount of feature points for these images due to their large format. This is in general a desirable behavior, because then the matching stage generates also many correspondences. That results in a sparse reconstruction consisting of many 3D points. Note that also the accuracy of the refined camera poses increases with the amount of correspondences [8]. Obviously also the matching effort increases. For example consider the matching of two images, each containing 100000 interest points. An exhaustive feature matching strategy would compare each feature of the first image with each feature of the second image, resulting in $100000^2 = 10^{10}$ comparisons and therefore a quadratic complexity in the amount of features. A more efficient approach is to predict the feature location in the second image for a feature in the first image and then comparing only features within a small search region around this prediction [8]. If it is assumed that about 100 interest points are found within each search region then only $100 \cdot 100000 = 10^7$ comparisons are required, which are just 0.1% compared to the exhaustive method.

Within this work methods for view selection and feature location prediction are investigated, which exploit the available prior knowledge. The considered prior information are measured poses, even though they are possibly imprecise and scene approximations like a ground plane or a Digital Elevation Model (`DEM`), see Fig. 1.1(a). Note that the elevation data is freely accessible and available for nearly every place on earth [9]. Furthermore with the obtained feature correspondences a sparse georeferenced scene is reconstructed and the scene as well as the camera poses are refined during a bundle adjustment stage, see Fig. 1.1(b) for an example.



(a) (b)

Figure 1.1: Example input data and corresponding AT pipeline output. In (a) the utilized input data are schematically shown, which are aerial images, imprecise measured camera poses and a DEM. The resulting georeferenced sparse scene reconstruction and the refined camera poses are given in (b).

The next chapter gives a brief overview of recent related work in the area of AT and Structure from Motion (SfM) pipelines. Afterwards the background chapter describes components used in a state of the art pipeline. Then the chosen methodology is delineated, containing alternative realizations for the distinct stages. The experiments chapter gives detailed evaluation results of the different processing steps. A conclusion summarizes the gained findings and states the directions for further work.

Chapter 2

Related Work

Contents

The following work is motivated by recent developments in the field of Visual Simultaneous Localization and Mapping (SLAM), more precisely by the therein contained subtask of efficient feature location prediction, cf. [10–13]. SLAM is the problem of locating a vehicle within an unknown environment and simultaneously building a map of this environment by evaluating some sensor information. In the particular case of Visual SLAM, a camera is used as sensor. Here a new camera pose is estimated from previous observations. This estimated camera pose and the current map are then used to predict feature locations in the current image. Due to pose uncertainties, the errors between predicted and true feature locations are correlated. These correlations can now be exploited to accelerate feature matching. In AT pipelines a similar problem occurs, also here inaccurate camera poses from GPS and Inertial Measurement Unit (IMU) measurements are available, which should be exploited for feature matching. In the following the working principle of commercial AT systems is outlined, afterwards recent developments in the area of AT and SfM pipelines are given.

2.1 Commercial AT pipelines

Various commercial AT systems with included bundle adjustment stage exist in the meantime, for example Inpho MATCH-AT and Inpho inBLOCK from Trimble [14] or LPS with the ORIMA add-on from Intergraph [15]. Although a recent information about the therein

used algorithms is not easily available, some basic image matching methods used within these packages are delineated in the following [4].

In general feature based, area based and structural matching methods are applied. Feature Based Matching (FBM) methods extract points, lines or even more complex objects from images and match them. Area based matching strategies compare the similarities of gray values within a reference window located in the first image with those of a search window in the second image. The location of the search window is modified until the similarity between both patches is a maximum. Structural matching [16] detects points, lines and regions as well as their relations and generates a structural description for each image. The images are then hierarchically matched by comparing their descriptions. The above mentioned matching methods often apply an image pyramid to perform a coarse to fine matching, starting with images of lower resolution.

One way to measure patch similarity for area based matching is to apply Normalized Cross Correlation (NCC), which correlates normalized image gray values. This approach achieves search window position accuracies of one pixel. Another area based matching method is Least Squares Matching (LSM) [17], which generates subpixel accurate locations. In contrast to NCC, the initial search window position for LSM must be already specified with a maximal deviation of two pixel from the correct location. LSM models geometric and radiometric distortions of a reference patch for fitting it to the corresponding location in the second image. This fitting procedure minimizes the gray value differences by adjusting the parameters of the geometric and radiometric transformations. The geometric distortion can be approximated by an affine transformation, while the radiometric distortion may be modeled by a linear relationship. The fitting is performed iteratively by solving least squares optimization problems. LSM is often used to refine the results of a preceding NCC matching step and allows the inclusions of constraints like the epipolar constraint. Also a simultaneous matching of patches in multiple images is possible with LSM, which is known as multiple image matching. Here the common 3D location of the corresponding object point is estimated. The location of the object point projection in each image is used as observation and guides the matching. Due to the inclusion of multiple observations in the matching process more stable results are achieved.

A common feature of commercial AT systems is their ability to determine various parameters for modeling imaging distortions. Additionally they are able to consider GPS shift and drift terms as well as the lever arm offset and the boresight misalignment. Furthermore they allow an extensive analysis of the results, for example by determining

the covariances of the calculated values.

In Fig. 2.1 the workflow of the commercial product Inpho MATCH-AT [18] is given as an example. MATCH-AT applies FBM and LSM within an image pyramid, searches tie points along epipolar lines and supports multi image matching. After each matching of an image pyramid level a robust bundle adjustment run is performed to update the camera poses. The matching stage utilizes either a given DEM or generates a DEM during the hierarchical matching.

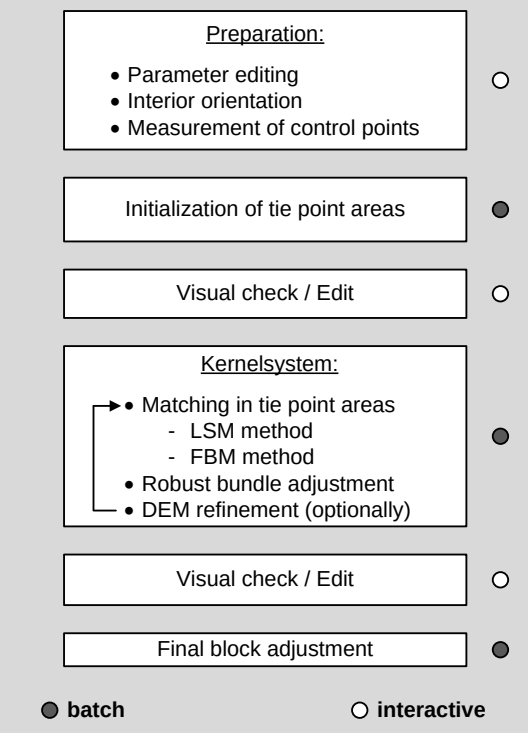


Figure 2.1: Workflow of Inpho MATCH-AT, from [18].

2.2 Recent developments

Some recent approaches for AT and SfM pipelines are proposed in [19–23]. In [22] images are first grouped to overlapping image sets, then these image sets are incrementally reconstructed, whereas in each iteration a bundle adjustment step is performed. Because the sets are reconstructed separately, this can be easily executed in parallel. The obtained results are then merged together by transforming them into a common coordinate system via similarity transform. Finally a global bundle adjustment step is applied. This ap-

proach is suitable for large image sets. There the camera poses are reconstructed solely from feature matches.

In [21] GPS and INS data is exploited to speed up the feature matching stage by considering only view pairs, where the corresponding image rectangles projected onto a scene approximation significantly overlap. They perform a single robust bundle adjustment run, where the camera centers are initialized with GPS measurements. Feature correspondences are used to estimate relative camera orientations, which in turn are registered with the GPS measurements to obtain initial camera orientations for the bundle adjustment step.

The approach in [23] performs at first an exhaustive pairwise feature matching on all images, whereas the computation time has been reduced by using only features of higher scale. These matches are utilized to estimate an affine transformation between image pairs. This transformation is on the one hand applied to calculate the overlap between images and on the other hand to guide the feature matching. The guidance places a search region for a feature point in one image into another image, so that the search region is centered at the transformed feature point location. Only feature points within the search region are considered for matching. Furthermore they discard interest points within a surrounding of low contrast.

Further strategies for efficient feature matching in the field of AT are given in [8, 24, 25]. The matching strategy in [24] triangulates world points using in track matches and predicts search regions for the corresponding tie points in images of neighboring flight strips by projecting the world points into these images. This is possible because camera positions and orientations are measured by GPS and INS. Then multi image matching in image space is performed, using NCC as similarity value. Geometric distortions are partially considered by a correlation window wrapping. Therefore a horizontal plane passing through the corresponding world point is utilized and the correlation window is projected onto that plane and back projected into the other images.

In contrast the matching strategy suggested in [8] back-projects a feature point onto a DEM. The resulting world point is then projected into the other images to obtain initial guesses for the feature matching search regions. Ideally this prediction procedure considers the involved uncertainties, e.g. camera pose inaccuracies, elevation data errors as well as feature extraction imprecisions, and adapts the size of the search region accordingly. Also here GPS and INS measurements are exploited.

A new robust feature matching strategy is proposed in [25]. Here interest points are detected by applying the Difference of Gaussian (DoG) operator on the scale space, similar

to the Scale-Invariant Feature Transform (SIFT) detector. At first a pairwise matching is performed by applying NCC on the image patches of the corresponding scale. Then LSM refines the locations of the interest points. Afterwards the relative pose between both cameras is estimated and refined by a robust bundle adjustment run. The pairwise matches between two images are combined to three view matches and afterwards merged to multi view correspondences. Each linkage step is followed by a multi image LSM to further refine the interest point locations and a robust bundle adjustment run. They perform a verification with the trifocal constraint.

Feature matching algorithms applied in the computer vision domain are given in [10, 12, 13, 26–28]. They are described within the background chapter of this thesis in more detail.

Chapter 3

Background

Contents

The goal of this thesis is to obtain a georeferenced 3D structure as sparse point cloud from high-resolution aerial imagery and simultaneously a refinement of the measured camera poses. This chapter gives first an overview of different geospatial coordinate systems. Then the applied geometric algorithms and data structures are briefly explained, followed by a description of the usually utilized camera model and a suitable pose measurement technique. Finally the processing stages in a typical `AT` pipeline are given, where for the distinct tasks alternative methods are listed. The stages of the considered `AT` pipeline are as following [19–23], a block diagram is given in Fig. 3.1.

- **Feature Extraction**
  The feature extraction stage finds image points, which are repeatable detectable in different images.

- **View Selection**
  To speed up the subsequent feature matching stage, view pairs showing the same scene content are determined.

- **Feature Matching**
  In this stage the previously extracted features are matched across views, so that each pairwise match corresponds to two observations of the same 3D location in the scene.

- **3D Structure Computation**
  The 3D structure consists of triangulated world points and is generated by the usage of the feature correspondences and the known camera parameters.

- **Bundle Adjustment**
  Uncertainties in the camera parameters and in the positions of the extracted features lead to imprecise locations of the triangulated world points. The bundle adjustment stage optimizes the camera parameters and world point locations to obtain a jointly optimal solution.

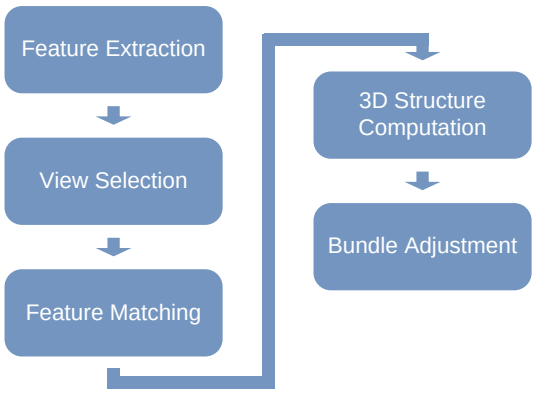


Figure 3.1: General flow chart of an AT pipeline.

3.1 Geospatial Coordinate Systems

As mentioned above the goal of this work is to obtain camera positions and triangulated world points within a global coordinate system. Therefore the input data like the measured camera positions and the elevation data of the processed regions have to be converted into a common coordinate system suitable for processing within the AT pipeline. So this

chapter introduces geospatial coordinates first. Then an overview of different types of map projections is given followed by a discussion of different elevation specifications. The remaining sections list additional information concerning the coordinate systems used within this thesis.

3.1.1 Coordinate Reference Systems

In the past a huge amount of different Coordinate Reference Systems (`CRSs`) emerged, suitable for different purposes. While some of them are used to define positions on the whole Earth others describe the location within a specific country. This section gives an overview of the different methods for defining positions, following mainly [29].

The first way to specify positions is to approximate the Earth surface with an ellipsoid and give positions onto this ellipsoid. Depending on the purpose, different reference surfaces exist, as illustrated in Fig. 3.2. While a globally best fitting ellipsoid is suitable to describe positions over the whole Earth within a single reference system, local best fitting ellipsoid are used in regional reference systems.

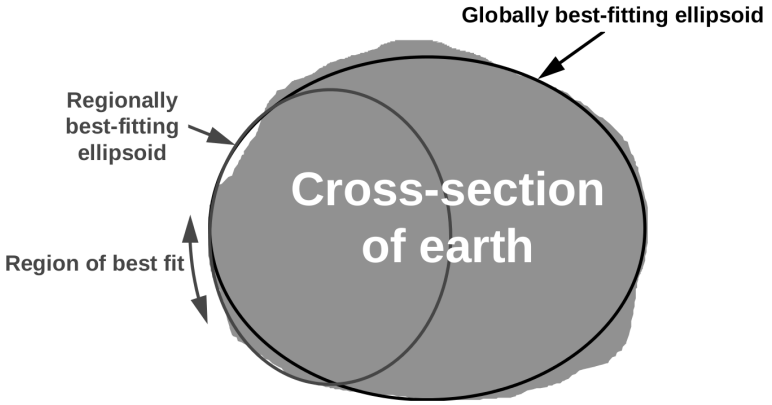


Figure 3.2: Approximation of the earth surface with ellipsoids, from [29]. While the globally best-fitting ellipsoid provides a reference surface for the whole earth, regionally best-fitting ellipsoids are able to give more accurate approximations within the regions they are designed for.

The position of a point P with respect to the ellipsoid is now given in form of two angles called latitude ϕ and longitude λ and additionally the height H above the ellipsoid, see Fig. 3.3. These angles are defined with the aid of a grid of meridians and parallels. A meridian is a north-south line of constant longitude, while a parallel is a east-west line of constant latitude. The prime meridian is that meridian with zero longitude and the equator is the circle of zero latitude. So the longitude value of a point on the ellipsoid is

defined as the angle between prime meridian and meridian through the point and usually given within the ranges $0° - 180°$ west and $0° - 180°$ east depending on which hemisphere the point is located on. Furthermore the latitude value of a point on the ellipsoid is specified as angle between equatorial plane and the line perpendicular to the ellipsoid passing through the point. Latitude values are given within the ranges $0° - 90°$ north and $0° - 90°$ south, also depending on the current hemisphere. Finally the ellipsoid height H is measured along the vector perpendicular to the ellipsoid passing through the point P.

The second way to specify a point on Earth is to use a right handed Cartesian coordinate system with the three orthogonal axes X, Y and Z. Its origin coincides with the center of the ellipsoid and the XY-plane lies within the equatorial plane. The positive X-axis points to the prime meridian and the positive Y-axis points to the 90° east meridian.

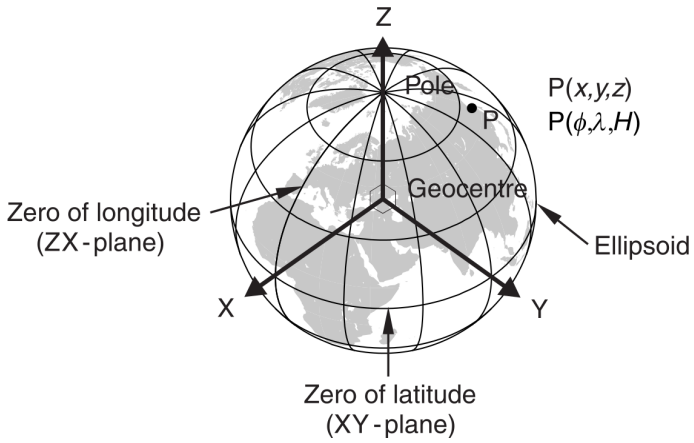


Figure 3.3: Earth coordinate systems overview, from [29]. The location of a point P can be given with the two angles latitude ϕ and longitude λ and additionally the height H above the ellipsoid. Alternatively the location of P can be specified within a Cartesian coordinate system.

For the definition of a reference system one has to specify the ellipsoid, that means center and orientation relative to the Earth, the ellipsoid size and its shape. Fig. 3.3 illustrates one such system. Here the center is located at the Geocentre, which is the centre of Earth's mass. Furthermore it is orientated in a way that the semi-minor axis, the shorter axis of the ellipse used to define the ellipsoid of revolution, coincides with the Earth rotation axis. Because we are interested in positions of objects on the Earth land surface such a definition is called a Terrestrial Reference System (`TRS`) or alternatively a datum. Examples of `TRSs` are the European Terrestrial Reference System 1989 (`ETRS89`), the International Terrestrial Reference System (`ITRS`) and the World Geodetic System

1984 (WGS84). To use a coordinate system in practice, one needs points with known positions with respect to the TRS, these points are called Terrestrial Reference Frame (TRF). For example in the case of GPS the exactly known positions of the tracking stations are used to determine the position of each satellite. The satellites broadcasts now their actual positions, which in turn are utilized to triangulate the position of a GPS receiver. Therefore all tracking stations act as TRF used to implement a realization of WGS84, the corresponding TRS.

3.1.2 Map Projections

As outlined above a point on Earth can be specified either as a pair of angles with a height above a reference ellipsoid or as a triple of Cartesian coordinates. A third possibility to give the position of a point on Earth is to apply map projections [30]. Here a point on the reference ellipsoid is projected onto a map surface, e.g. plane, cylinder or cone. Within the projection surface the location of a point is given in Cartesian coordinates. A height component is not handled and must be specified otherwise. Map projections have the disadvantage of introducing geometric distortions, which depend on shape and size of the mapping surface as well as on its position and alignment relative to the reference ellipsoid. Fig. 3.4 gives an overview of different map projections.

3.1.3 Geoids

As mentioned earlier the height of a point can be given relative to a reference ellipsoid, which is a geometric height. However in every days usage one assumes that between points of equal height no water flows, e.g. the water of a river runs always downhill, starting form a point with a given height towards a point with a lower height. This implicitly means, that points of equal height must have the same potential energy in Earth's gravitational field. So they have to be on the same level surface. Therefore the height must be defined relative to a reference level surface which in principle can be chosen arbitrarily. Note that the direction of gravity is always at right angle to a level surface [29].

Due to irregular mass distribution on Earth's surface e.g. mountains and variations of the Earth's density, the level surfaces are complex shaped and can be only approximately represented by an ellipsoid. One common level surface is the Geoid [29, 31], which best fits the Earth's oceans average water level. The height above the Geoid is also called orthometric height. Ellipsoid height H and orthometric height h are related by the Geoid-ellipsoid separation N according to the simplified formula given in Equation (3.1).

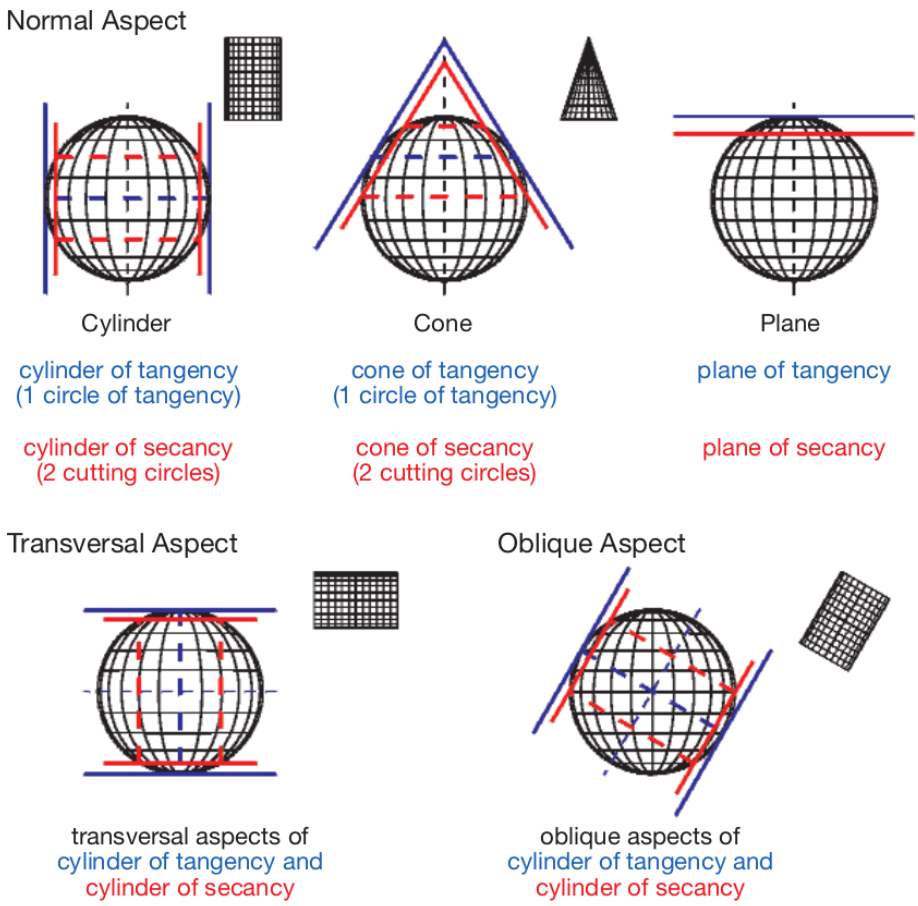


Figure 3.4: Overview of map projections, from [30]. The categorization is performed according to the shape of the mapping surface, its orientation which is also called aspect and its size. Tangential mapping surfaces are given in blue while intersecting one are displayed in red.

$$H = h + N \tag{3.1}$$

A Geoid model can be stored as lookup table delivering the Geoid-ellipsoid separation N for a given latitude longitude pair. Fig. 3.5 shows the Earth Gravitational Model 1996 (`EGM96`) Geoid-ellipsoid separation for the `WGS84` ellipsoid. It can be seen that the `WGS84` ellipsoid approximates the Geoid with an accuracy of about ± 100 m. The newer and improved Earth Gravitational Model 2008 (`EGM08`) Geoid has not been applied within this work, because the utilized height data refers to the older `EGM96` Geoid.

Beside the global Geoid also local geoids [29], optimized for a specific country, exist. They are usually defined via the Mean Sea Level (`MSL`) observed at a single specific coastal

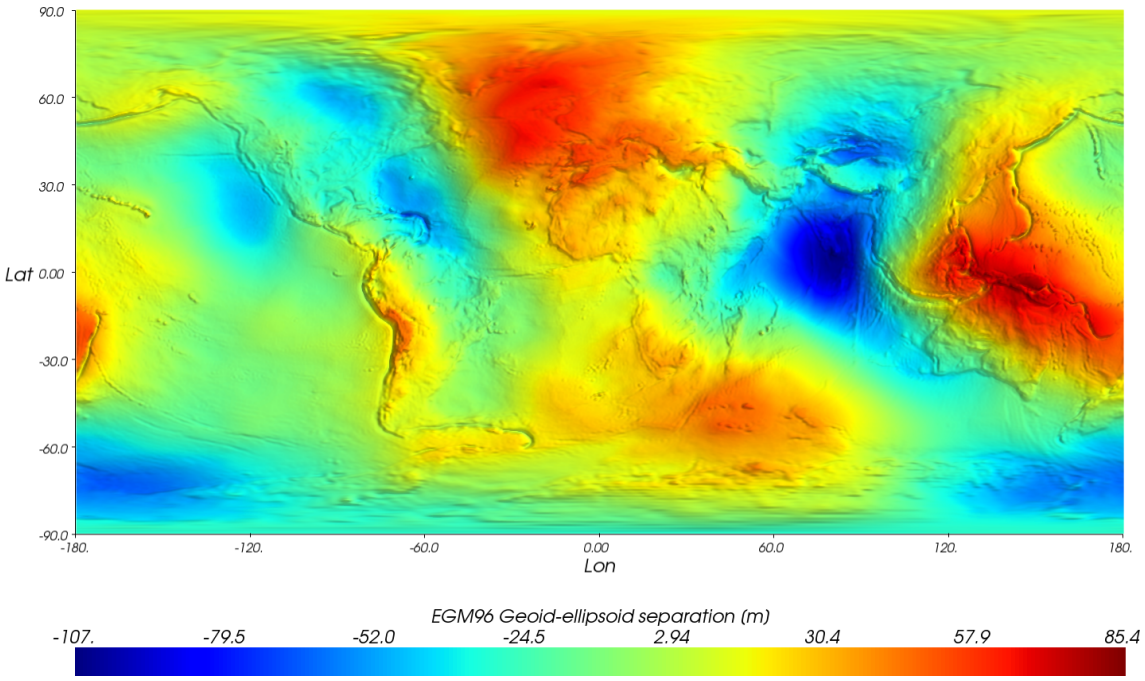


Figure 3.5: Geoid-ellipsoid separation for the EGM96 Geoid and WGS84 ellipsoid, from [31].

place and at a given time, forming a national tide gauge datum. Effects like oceanic currents, tides, wind, variations in water temperature and pressure produce hills and valleys in the water surface, causing that the utilized MSL differs from the average water level of all oceans. That means the heights obtained from a local geoid and the heights related to the global Geoid differ. These deformations of the sea surface are also known as Sea Surface Topography (SST). Fig. 3.6 summarizes the relationship between the different heights.

3.1.4 World Geodetic System 1984

WGS84 [29] is used as TRS for global coordinate systems and applied for example for GPS. The reference ellipsoid and therefore also the associated three dimensional Cartesian coordinate system have its origin at the Geocentre. Furthermore the reference ellipsoid has the same size and shape as the Geodetic Reference System 1980 (GRS80) ellipsoid, which best-fits the Earth's Geoid. Table 3.1 lists semi-major and semi-minor axis of the ellipse utilized to construct the GRS80 ellipsoid by rotating the ellipse around its semi-minor axis. Equator and prime meridian of the reference ellipsoid coincide with the definition from

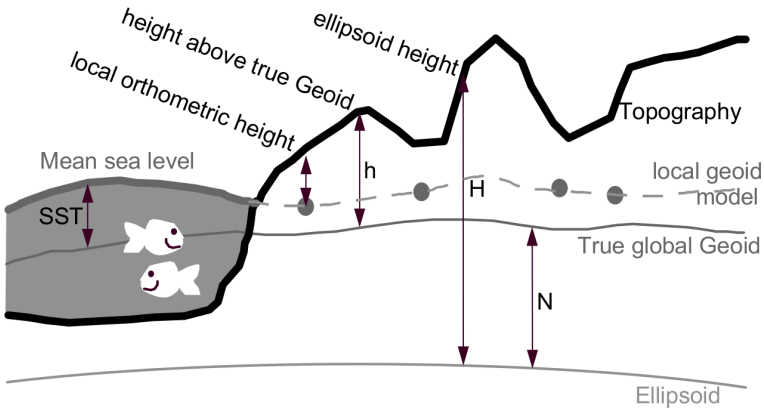


Figure 3.6: Illustration of the relationship between different heights, from [29].

Parameter	Value
Semi-major axis a	6378137.000 m
Semi-minor axis b	6356752.3141 m

Table 3.1: Shape and size of the `GRS80` ellipsoid, from [29].

the Bureau Internationale de l'Heure at epoch 1984.0, which is the beginning of year 1984.

Because `WGS84` is designed for global usage, the relative motion of continents has to be considered. This is accomplished by updating the ellipsoid orientation in a way, that the average motion of all tectonic plates relative to the ellipsoid is zero. This implies that the `WGS84` Cartesian axes are steadily moving relative to a single continent. The International Reference Pole and International Reference Meridian are defined in the same manner and coincide with the Z-axis and prime meridian of the `WGS84` ellipsoid respectively. An alternative for mapping tasks is the `ETRS89` which is fixed to the Eurasian plate.

An Earth-Centered, Earth-Fixed (`ECEF`) [32] coordinate system is a three dimensional Cartesian coordinate system, where the center coincides with the Geocentre of the Earth. Furthermore it is rotating with the Earth. So the `WGS84` Cartesian coordinate system is also an `ECEF` coordinate system. In this work all `ECEF` coordinate values refer to the `WGS84` Cartesian coordinate system.

3.1.5 Universal Transverse Mercator

Universal Transverse Mercator (`UTM`) [29, 30] is a map projection scheme applying cylinders with transversal aspect as projection surfaces. The Earth's longitude range is subdivided into 60 zones, each having a width of 6°. Therefore the central meridians of each zone are separated by 6° longitude. To reduce the maximal scale distortion, each zone utilizes a

projection surface with a radius slightly smaller than the length of the semi-minor ellipsoid axis. Distances along the central meridian are hence by the factor 0.9996 lower than their real value. Due to the orientation of the projection cylinders, the first Cartesian axis within the map surface points in east direction, while the second axis points in north direction. In practice different reference ellipsoids are used in combination with `UTM`, within this work only the `WGS84` ellipsoid is utilized. Furthermore within this thesis the zone is denoted by the zone number followed by the letter N or S indicating the north or south hemisphere, e.g. $33N$ denotes zone 33 north hemisphere.

3.1.6 East North Up Coordinate System

The East North Up (`ENU`) [32] coordinate system consists of the three orthogonal axes east north and up. It is defined relative to an arbitrarily chosen reference point on the reference ellipsoid. At the reference point a plane tangential to the reference ellipsoid is placed. The plane contains the two axes east E and north N, which point into their corresponding directions. The third axis up U is orthogonal to the tangential plane and points away from the Earth's Geocentre. One usually selects the reference point as center of the Region of Interest (`ROI`). Fig. 3.7 shows schematically an `ENU` coordinate system nearby the given point P. An advantage over map projection is, that no distortions are introduced, because the `ENU` system is just a translated and rotated `ECEF` system.

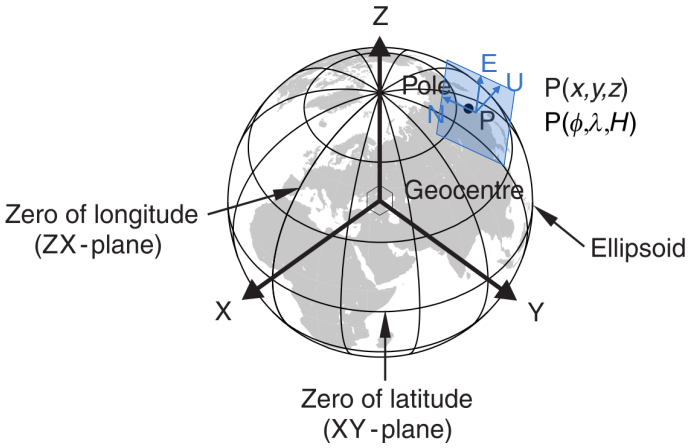


Figure 3.7: `ENU` coordinate system, adapted from [29, 32]. The two axes east E and north N lie in the blue marked tangential plane while the axis up U is perpendicular to the plane.

3.1.7 Example

To illustrate the application of different geospatial coordinate systems, a point near Uhrturm Graz has been selected and its coordinate values have been determined for different CRSs. Fig. 3.8 shows the selected point in an orthophoto and Table 3.2 lists the corresponding coordinates. Note that the UTM and ECEF values are related to the WGS84 ellipsoid.

Figure 3.8: Orthophoto of Uhrturm Graz, from [33]. The marker depicts the horizontal location in WGS84 format.

Parameter	Value
WGS84 latitude	47.073685° north
WGS84 longitude	15.437692° east
UTM 33N east	533230 m
UTM 33N north	5213445 m
Height H above WGS84 ellipsoid	472.3 m
Height h above EGM96 Geoid	424.8 m
Height h above local geoid	425.2 m
ECEF X	4194996.2 m
ECEF Y	1158464.1 m
ECEF Z	4647693.5 m

Table 3.2: Geospatial coordinates of the point shown in Fig. 3.8, from [33].

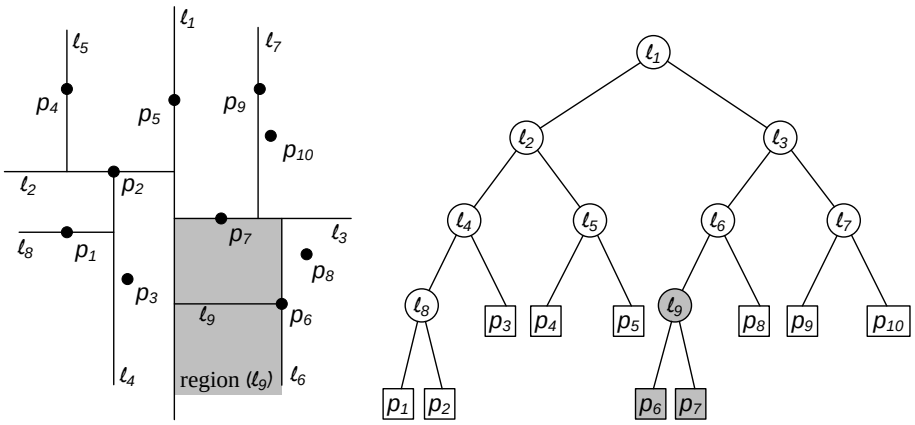
3.2 Geometric Algorithms and Data Structures

Within this work some geometric algorithms and data structures are used to efficiently process the distinct stages in the `AT` pipeline. This section gives a short overview about them.

3.2.1 kD Tree

For performing range queries in a k dimensional Cartesian space on point like data one can apply kD trees [34]. The input data is recursively split into halves by the usage of hyperplanes orthogonal to one coordinate axis. While the internal nodes store the separating hyperplanes the leaf nodes contain the data points. More precisely the tree construction procedure is as following. The root node splits the given data points into two equally sized sets by using a hyperplane orthogonal to the first dimension. Then the data is forwarded to both generated child nodes and the separating plane is stored at the root node. Now each child node splits the forwarded data along the second dimensions into equally sized sets and stores again the corresponding separating plane. The newly generated partitions are forwarded to the next level and split along the next dimension. When the depth k is reached the subdivision starts again with the first dimension. This is repeated until each leaf node contains only a single point. Fig. 3.9 shows an example for two dimensional input data p_i including the separating lines ℓ_i and the corresponding kD tree. It can be seen that each node represents a region within the input space, for node ℓ_9 the region is indicated by the gray shaded area in Fig. 3.9.

A range search is performed by traversing the tree downwards and comparing a nodes region with the search window. Is the region of a node entirely within the search window, all leaf nodes belonging to the investigated node are returned without any further checks. In the opposite case, when a nodes region is completely outside, then the node isn't further analyzed. For all other cases the tree is traversed down until either the nodes region is completely inside or outside the search window or a leaf node is reached. When a leaf node is reached it is tested, whether the corresponding point is within or outside the window. While a nodes region is always rectangular shaped, this is not necessarily the case for the search window.



(a) (b)

Figure 3.9: kD tree built from two dimensional point data, adapted from [34]. The points p_i in (a) are subdivided into partitions by using axis parallel lines ℓ_i, (b) shows the resulting kD tree.

3.2.2 Range Tree

When rectangular axis aligned search windows are used for range queries on point data, one can also use range trees [34]. They have the advantage that the performed queries are asymptotically faster in comparison to kD trees, but have the drawback of an increased memory consumption. A range tree can be seen as extension of a binary search tree to higher dimensions.

Binary search trees store data points in leaf nodes, while internal nodes contain threshold values. Data points with a value below or equal to the threshold stored in an internal node are located on the left subtree of the internal node, while all others can be found on the right subtree. Fig. 3.10 illustrates the result obtained from a range search in the interval $[x, x']$ on a binary search tree $\mathcal{T}$. The search starts at root node $root(\mathcal{T})$ and leads to the leaf nodes μ and μ' for the lower and upper bounds x and x' respectively. The traversals for both bounds take the same path through the tree until the internal node v_{split} is reached, there the search path branches. The search result consists of all subtrees directly connected to one of both search paths having a depth higher than that of v_{split}. They are gray shaded in Fig. 3.10. The leaf nodes of one subtree with root node v are called the canonical subset and denoted with $P(v)$. If such a subtree is discovered during the traversal it can be directly reported without any further checks. Whether the result contains also the nodes μ and μ' must be checked separately.

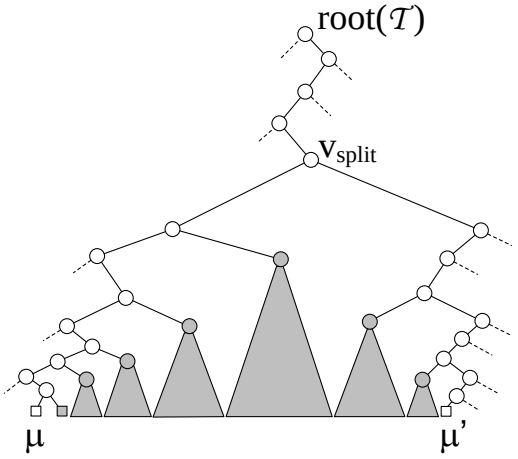


Figure 3.10: Illustration of the result from a one dimensional range query performed on a binary search tree, from [34].

Now let's examine a range tree for performing two dimensional queries with a window $[x, x'] \times [y, y']$. For the search along the first dimension a binary search tree is applied again to obtain all canonical subsets $P(v)$ with points in the range $[x, x']$. Instead of reporting each subset, a range search within the interval $[y, y']$ along the second dimension has to be performed. Therefore each node v in the binary search tree $\mathcal{T}$ of the first dimension points to an associated structure formed by another binary search tree $\mathcal{T}_{assoc}(v)$ operating on the second dimension. The second search tree now reports all points of $P(v)$, where the second dimension is in the range $[y, y']$. Note that for each node v in the binary search tree of the first dimension another binary search tree containing $P(v)$ and operating on the second dimension exists, see Fig. 3.11.

The extension to higher dimensions is straight forward, for example in a three dimensional range tree one node v of the first binary search tree points to a two dimensional range tree handling $P(v)$. Query time improvements are possible by applying a technique called fractional cascading, which replaces the binary search trees of the last dimension by sorted arrays and exploits the fact, that each array is searched for the same range, e.g. $[y, y']$ in the above example.

To handle data other than points within a range query, one can represent the input data as points in a higher dimensional space [34]. For example the problem of searching all triangles entirely within an axis aligned rectangular window in a two dimensional space can be replaced by the problem of searching points in a four dimensional space, where the first two dimensions represent the top left corner of the triangles bounding box and the

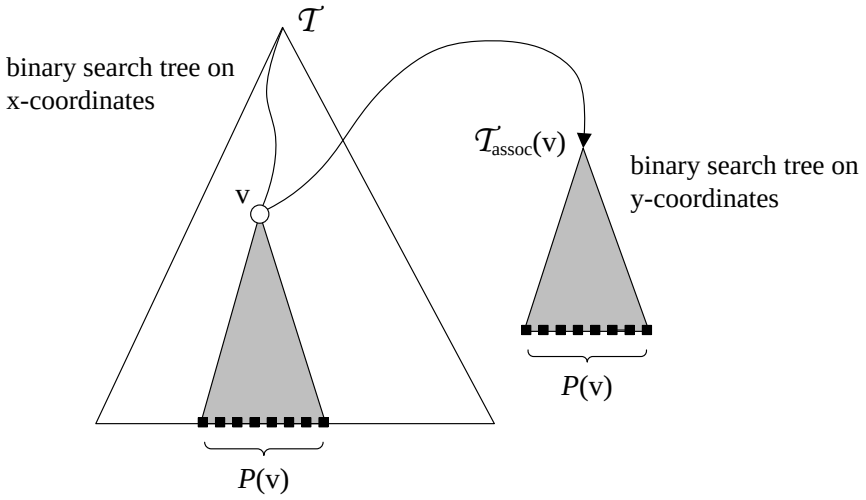


Figure 3.11: Range tree for two dimensional point data, from [34]. The Figure shows the linkage between one node in the binary search tree operating on the first dimension and the associated binary tree operating on the second dimension.

last two dimensions contain the bottom right corner of the triangles bounding box.

3.2.3 dD Segment Tree

Segment trees [34, 35] can be used to perform the following queries on d-dimensional intervals i.e. axis aligned bounding boxes in d-dimensional space:

- **Inverse Range Query** Given a query point, determine all intervals containing that point.
- **Enclosing Query** Given a query window, determine those intervals which enclose the given window.
- **Window Query** Given a query window, determine those intervals which are contained in the given window or overlap with the window.

At first let's consider the one dimensional case. To build a segment tree, the start and end points p_i defining the given intervals are sorted in ascending order. Then elementary intervals between each two consecutive points $[p_i : p_{i+1}]$ are generated. They split the given intervals into smaller parts, for example the segment s_1 in Fig. 3.12 is divided into the two elementary intervals $[p_1 : p_2]$ and $[p_2 : p_3]$. From the elementary intervals a balanced binary search tree is built bottom up, where each node stores the interval range

it represents. Furthermore the leaf and internal nodes store their assigned segments s_i. A segment is always assigned to the top most node, which has an interval range entirely contained within the segment. That means, when a node v contains a segment, all child nodes do not hold the segment. The segments $S(v)$ owned by a node v are called the canonical subset of node v.

An inverse range query is now performed by traversing the segment tree top down, passing all nodes with an interval range containing the query point and reporting their assigned segments. Note that one usually takes an interval tree to perform an inverse range query on one dimensional intervals due to their lower memory consumption. A disadvantage of the interval tree is, that no generalization to higher dimensions exists.

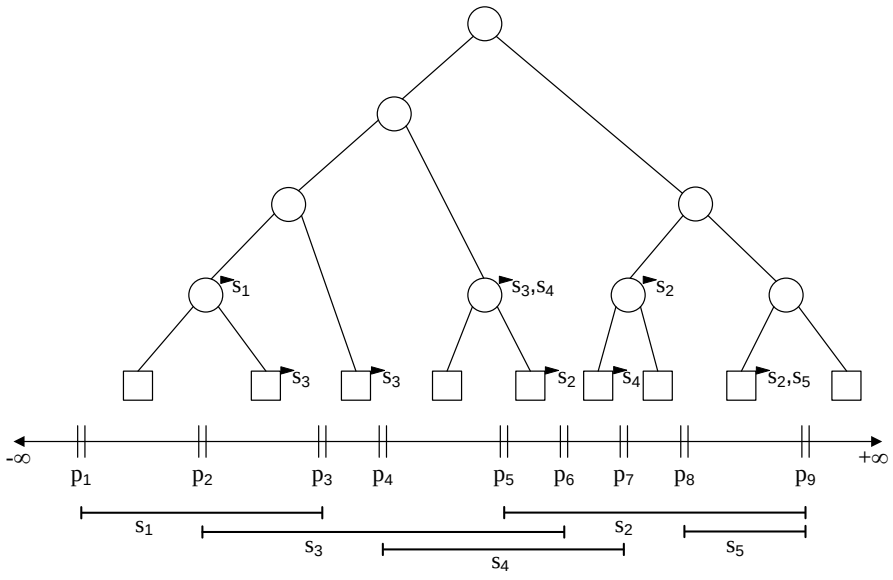


Figure 3.12: Segment tree for one dimensional intervals, adapted from [34, 35]. The given segments s_1, s_2, s_3, s_4, s_5 with their start and end points p_i and the resulting elementary intervals $[p_i : p_{i+1}]$ are shown below the tree. Furthermore the assigned segments are listed beside the nodes.

The same approach used to extend the range tree to multidimensional point data can be applied to extend the segment tree to multidimensional intervals, e.g. axis aligned bounding boxes within a plane. Fig. 3.13 illustrates a segment tree for two dimensional intervals. Here each node v of the segment tree $\mathcal{T}$ operating on the first dimension points to another associated segment tree $\mathcal{T}_{assoc}(v)$ operating on the second dimension of the corresponding canonical subset $S(v)$. Note that it is possible to combine range and segment trees, for example $\mathcal{T}$ might be a segment tree while $\mathcal{T}_{assoc}(v)$ could be a range tree.

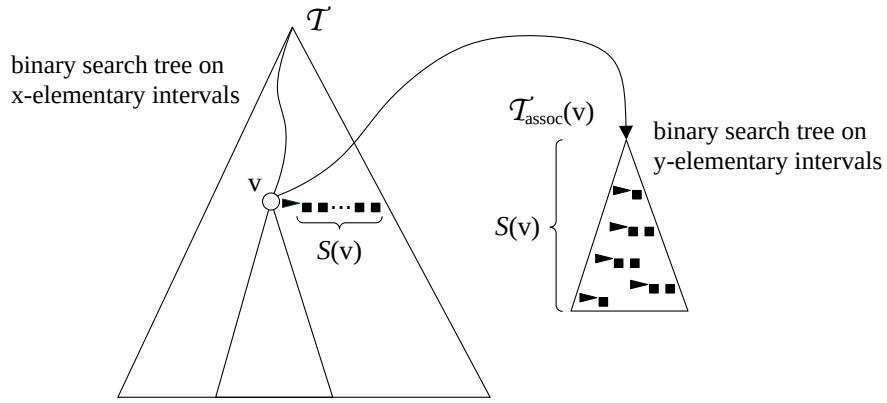


Figure 3.13: Segment tree for two dimensional axis aligned bounding boxes, adapted from [35]. The Figure illustrates the linkage between one node in the segment tree operating on the first dimension and the associated segment tree operating on the second dimension.

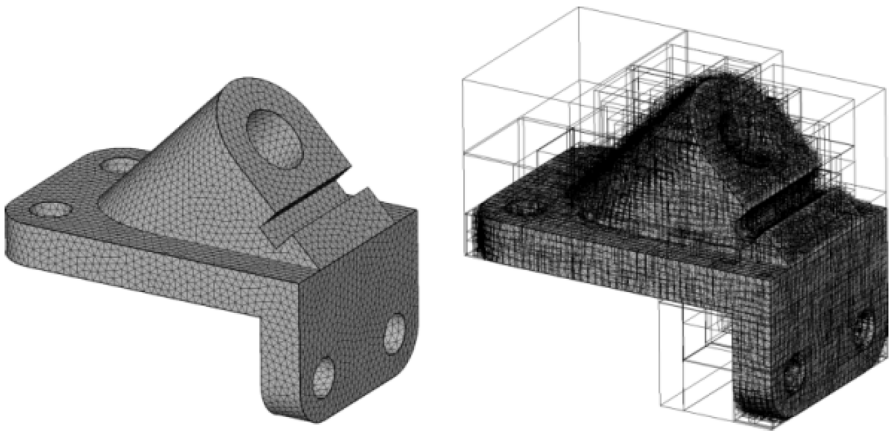
3.2.4 Axis-Aligned Bounding Boxes Tree

Axis Aligned Bounding Box (`AABB`) trees [36, 37] represent models assembled by geometric primitives e.g. triangles or polygons with a hierarchy of minimum sized `AABBs`. Starting from the root node containing the bounding box of the whole model, the tree is generated top down by subdividing the nodes bounding box into two smaller ones. This process is repeated until each leaf node contains only a single primitive, yielding to a binary tree. The subdivision is performed by utilizing a plane orthogonal to the largest box dimension. Primitives are now assigned according to their center location to either one of the two newly generated boxes. One can position the plane in a way that both boxes contain the same amount of primitives leading to a balanced tree. Another common method is to place the plane in the middle of the bounding box to generate equally sized child boxes.

One advantage of hierarchical bounding volume representations is, that intersection computations are performed with a divide and conquer strategy against the bounding volumes. When a bounding box does not intersect with a query object, all child boxes with the corresponding leaf nodes and the therein contained primitives also do not intersect and are therefore not further analyzed. In the opposite case, the tree is traversed downwards to those leaf nodes which intersect. Finally the test is repeated for those primitives contained in the found leaf nodes to verify, whether the intersection really exists. So the amount of costly primitive intersection tests is minimized.

Beside `AABB` trees there exist other hierarchical bounding volume representations, utilizing a different bounding volume representation like Oriented Bounding Boxes (`OBBs`)

or spheres. The example in Fig. 3.14 shows an AABB tree generated from a triangulated surface.



(a) (b)

Figure 3.14: AABB tree representation of a triangulated surface mesh, from [37]. The AABB tree in (b) is built by using the triangles of (a) as input primitives.

The AABB tree implementation of [37] supports different intersection queries, e.g. return all points located on a surface mesh intersecting with a given ray or return all line segments defining the intersection between the model and a plane. Furthermore distance queries are supported, e.g. search the point p on a surface mesh closest to a given query point q. These distance queries are implemented as intersection queries of the AABB tree with a sphere centered at q, where the sphere radius is steadily decreased during the search to the distance of the actual nearest intersection, discarding all bounding boxes outside the sphere. The implementation of [37] builds a kD tree from points located on the models surface to obtain an initial guess of the sphere radius.

3.2.5 Triangulated Surface Mesh Simplification

Triangulated surface meshes obtained from regularly sampled points often consist of a huge amount of triangles making further processing time consuming and memory intensive. By applying surface mesh simplification the number of triangles can be significantly reduced, while the volume, shape and boundary of the original surface is retained as much as possible. Within this thesis the approach implemented in [38] is used, it simplifies the

amount of triangles by performing edge collapses. An edge collapse replaces an edge by a single vertex and therefore removes the two triangles containing the edge. The process is controlled by a placement and a cost function. While the placement function returns the optimal position for the vertex resulting from the collapse, the cost function calculates an error describing the deviation from the surfaces actual volume, shape and boundary when the vertex is positioned according to placement function. In the first step of the algorithm a mutable priority queue is initialized with the costs for each edge collapse. Then the edge with the lowest cost is collapsed and the priority queue is updated. This is repeated until a specified stop criterion is met e.g. the target face count is reached. Note that here the error is measured as the deviation from the actual surface instead of the deviation from the original surface. Beside this method also global error tracking algorithms exist, which measure the deviation from the original surface.

3.2.6 2D Regularized Boolean Set-Operations

The 2D regularized Boolean set-operation implementation of [39] supports the following calculations defined on two point sets P and Q. These points sets are described as combinations of general polygons, which are polygons allowing edges other than straight lines, for example arcs. However within this work polygons are only formed by straight lines.

- Intersection $R = P \cap Q$
- Union $R = P \cup Q$
- Difference $R = P \setminus Q$
- Symmetric Difference $R = P \oplus Q = (P \setminus Q) \cup (Q \setminus P)$
- Complement $R = \bar{P}$
- Intersection predicate $P \cap Q \stackrel{?}{=} \emptyset$

A regularization step removes isolated vertices and antennas by taking the interior of the result obtained from an ordinary binary operation and returning the closure of the interior.

3.3 Camera Model

This section describes the imaging process by first introducing the finite projective camera and then extending it to incorporate common types of distortion. Afterwards the mea-

surement of camera positions and orientations is explained by the combined usage of GPS and strapdown INS.

3.3.1 Projection Equation

Within this work the aerial imaging device is described by a finite projective camera [40], which is based on the pinhole camera model. In the following the imaging process is briefly explained. The image $\mathbf{x}$ of a world point $\mathbf{X}$ is determined by the intersection of a ray with the image plane. As can be observed in Fig. 3.15(a) the ray passes through $\mathbf{X}$ and the projection center $\mathbf{C}$. The projection center $\mathbf{C}$ is placed at the origin of the camera coordinate frame, which has the axes X_{cam}, Y_{cam} and Z_{cam}. Furthermore the image plane is perpendicular to the Z_{cam} axis and placed at a distance f from the projection center, this distance is called focal length. The Z_{cam} axis is also known as principal axis and the intersection of principal axis with image plane is called principal point $\mathbf{p}$. The view vector is a vector pointing in positive Z_{cam} direction. As illustrated in Fig. 3.15(b) the y coordinate of the image point $\mathbf{x}$ can be obtained by the usage of similar triangles, the x coordinate is calculated similarly.

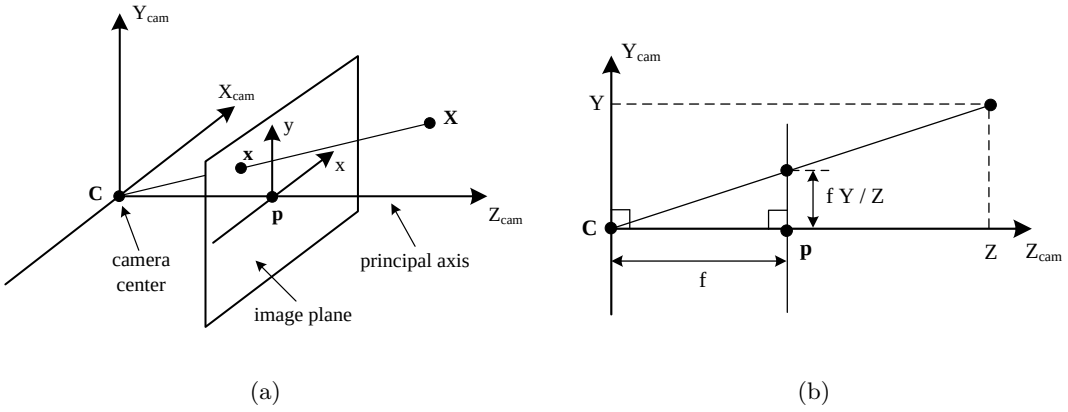


Figure 3.15: Pinhole camera model, from [40]. In (a) the camera frame with the axes X_{cam}, Y_{cam} and Z_{cam} is shown, while (b) illustrates the similar triangles occurring in the projection of one world point onto the image plane.

Fig. 3.16 illustrates the relation between the involved coordinate frames. The world coordinate frame consists of the coordinate axes X_{world}, Y_{world}, Z_{world} and the origin O. It describes point locations by a Cartesian CRS. Camera and world coordinate frames are related by a 3×3 rotation matrix R and a translation vector $\mathbf{t}$, given as 3-vector.

Furthermore the image coordinate frame with the axes x_{img} and y_{img} usually originates at an image corner, so it is shifted against the principal point.

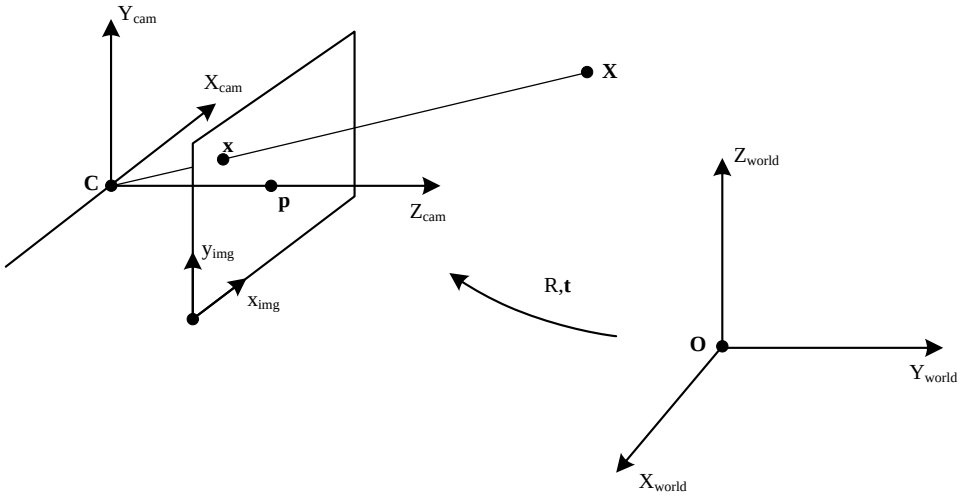


Figure 3.16: Relation between world, camera and image coordinate frame, adapted from [40].

According to [40] the following conventions will be applied. Locations within the world or camera coordinate frames are indicated by variables named with upper case letters in opposite to image frame positions which are denoted with lower case letters. Moreover inhomogeneous positions are denoted with the tilde symbol, e.g. the homogeneous world point $\mathbf{X} = [X, Y, Z, 1]^T$ has the inhomogeneous counterpart $\widetilde{\mathbf{X}} = [X, Y, Z]^T$.

The imaging process is mathematically described by the projection equation, see Equation (3.2). While this is common in the Computer Vision community, photogrammetrists usually apply the collinearity equations [41], which are just a rewriting. The camera orientation and position are described by the above mentioned rotation matrix R and the translation vector $\mathbf{t}$ respectively, the adaption to the interior properties of the camera is achieved by the camera calibration matrix K. It is an upper triangular matrix containing the scaled focal lengths $\alpha_x = m_x\, f$ and $\alpha_y = m_y\, f$, the location of the principal point $\mathbf{p} = [x_0, y_0, 1]^T$ relative to the image coordinate frame and the skew s. The scaling factors m_x and m_y convert metric distances into pixel units. They are different from each other, if the camera has non square pixels. Skew s is usually zero unless the pixels are neither square nor rectangular shaped, which might occur as the result of taking an image from a photo. Now the 3×4 projection matrix P is calculated from the camera calibration matrix K and from the rotation matrix R and translation vector $\mathbf{t}$. While the

calibration matrix contains the intrinsic parameters, the extrinsic parameters are stored in R and $\mathbf{t}$. The translation vector in turn is calculated from the center of projection, given in inhomogeneous coordinates $\widetilde{\mathbf{C}}$.

$$\mathbf{x} = P\mathbf{X} \qquad P = K[R|\mathbf{t}] \qquad K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{t} = -R\widetilde{\mathbf{C}} \tag{3.2}$$

3.3.2 Distortions

In practice the above introduced finite projective camera model is only an approximation to the real imaging process due to distortions, even when the camera is laboratory calibrated. That is because different environmental conditions, like distinct flying heights, cause different distortions. One approach is to modify the camera calibration, so that the distortions occurring at the expected operating conditions are compensated [42]. Another method is to extend the above imaging equations to model the distortions. In [41] three distortion types are considered for aerial images. While radial and decentering distortions occur within the camera, atmospheric effects outside the camera cause a bending of the ray of light, which is known as atmospheric refraction. Also camera manufacturer specific distortion models exist. However, in the following only radial and decentering distortions are inserted into the camera model.

Therefore Equation (3.2) is split into Equation (3.3) and Equation (3.5) similar to [43]. The radial distortion components Δx_{radial}, Δy_{radial} and the decentering distortion components Δx_{decen}, Δy_{decen} are added to $\mathbf{x}_p$ before the conversions to pixel coordinates is performed. Note that also other formalisms for integrating the distortion terms can be found in the literature.

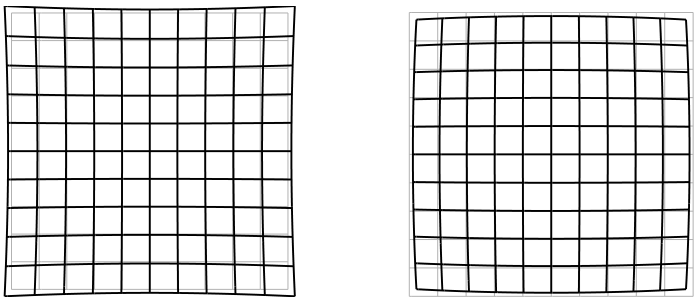
$$\mathbf{x}_p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [R|\mathbf{t}]\ \mathbf{X} \tag{3.3}$$

$$\tilde{x}_p = \frac{x_p}{z_p} \qquad \tilde{y}_p = \frac{y_p}{z_p} \qquad r = \sqrt{\tilde{x}_p^2 + \tilde{y}_p^2} \tag{3.4}$$

$$\mathbf{x} = \begin{bmatrix} m_x & s & x_0 \\ 0 & m_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_p + \Delta x_{radial} + \Delta x_{decen} \\ \tilde{y}_p + \Delta y_{radial} + \Delta y_{decen} \\ 1 \end{bmatrix} \tag{3.5}$$

Radial distortion is modeled by a polynomial depending on the distance r of an image point from the principal point. For the above calculated image point $\mathbf{x}_p = [x_p, y_p, z_p]^T$ the distance r is calculated according to Equation (3.4). The distortion components Δx_{radial} and Δy_{radial} are calculated by Equation (3.6) using the coefficients k_1, k_2 and k_3, where k_3 is often neglected. With the radial terms it is possible to model pincushion and barrel distortions, see Fig. 3.17(a) and Fig. 3.17(b) respectively.

$$\begin{aligned} \Delta x_{radial} &= \tilde{x}_p \left(k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ \Delta y_{radial} &= \tilde{y}_p \left(k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{3.6}$$



(a) (b)

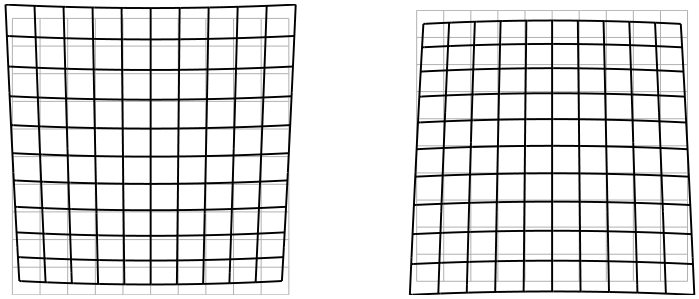
Figure 3.17: Radial distortion, adapted from [44]. In (a) the effect of radial distortion with positive k_1 and $k_2 = k_3 = 0$ is given, while (b) illustrates radial distortion with negative k_1 and $k_2 = k_3 = 0$.

Decentering distortion is modeled by Equation (3.7) and uses the coefficients p_1 and p_2. An illustration is shown in Fig. 3.18.

$$\begin{aligned} \Delta x_{decen} &= p_1 \left(2\tilde{x}_p \tilde{y}_p\right) + p_2 \left(r^2 + 2\tilde{x}_p^2\right) \\ \Delta y_{decen} &= p_1 \left(r^2 + 2\tilde{y}_p^2\right) + p_2 \left(2\tilde{x}_p \tilde{y}_p\right) \end{aligned} \tag{3.7}$$

3.3.3 Epipolar Constraint

The epipolar constraint [40] limits the possible locations of a world point projection in one image, if the corresponding projection into another image is known. On the one hand the epipolar constraint can be used to reduce the search region during feature matching, on



(a) (b)

Figure 3.18: Decentering distortion, adapted from [44]. In (a) the effect of decentering distortion with positive p_1 and $p_2 = 0$ is given, while (b) illustrates decentering distortion with negative p_1 and $p_2 = 0$. For $p_1 = 0$ and positive or negative p_2 the figures are similar, only the horizontal and vertical axes are swapped.

the other hand it enables the detection of false matches.

Let's consider two cameras where the projection centers of camera 1 and 2 are denoted with $\mathbf{C}$ and $\mathbf{C}'$ respectively. The line between both camera centers is called baseline and the projection of one camera center into the other camera is known as epipole. A world point $\mathbf{X}$ and both camera centers $\mathbf{C}$ and $\mathbf{C}'$ define a plane, see Fig. 3.19. Given the camera point $\mathbf{x}$ in camera 1 a ray can be constructed onto which the world point $\mathbf{X}$ must be located. This ray is coplanar with the before mentioned plane and the projection of the ray into camera 2 is called an epipolar line. This epipolar line $\mathbf{l}'$ passes through the epipole $\mathbf{e}'$ and contains the corresponding camera point $\mathbf{x}'$.

The fundamental matrix F contains the whole information for calculating the epipolar line $\mathbf{l}'$ from a camera point $\mathbf{x}$, see Equation (3.8). Note that $\mathbf{x}$ and $\mathbf{x}'$ are 3-vectors representing the camera points in homogeneous coordinates. Also the line $\mathbf{l}'$ is defined as 3-vector, so that only points $\mathbf{x}'$ on the line fulfill $\mathbf{x}'^T\mathbf{l}' = 0$. From that follows directly Equation (3.9).

$$\mathbf{l}' = F\mathbf{x} \tag{3.8}$$

$$\mathbf{x}'^T F\mathbf{x} = 0 \tag{3.9}$$

When the camera calibration matrices K and K' as well as the relative rotation matrix

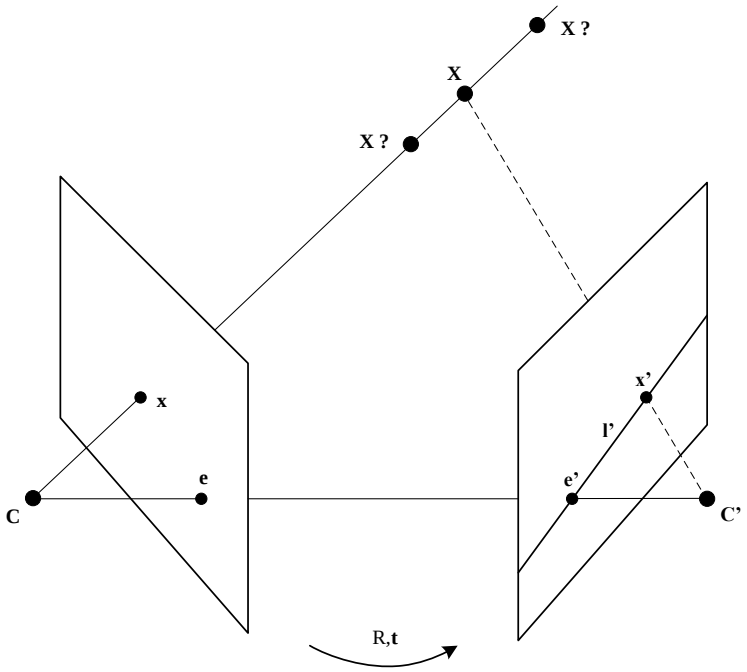


Figure 3.19: Epipolar Geometry, adapted from [40]. The camera centers $\mathbf{C}$, $\mathbf{C}'$ and the world point $\mathbf{X}$ form a plane, whereas the intersection of this plane with the image plane is called epipolar line, e.g. $\mathbf{l}'$. Furthermore the epipoles $\mathbf{e}$ and $\mathbf{e}'$ are shown.

R and the relative translation vector $\mathbf{t} = [t_1, t_2, t_3]^T$ between both cameras are known, then the fundamental matrix can be directly calculated according to Equation (3.10). For example assume the projection matrices $P = K[I|\mathbf{0}]$ and $P' = K'[R|\mathbf{t}]$, then the relative pose is immediately given with R and $\mathbf{t}$.

$$F = K'^{-T}[\mathbf{t}]_\times R K^{-1} \qquad [\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{3.10}$$

Alternatively it is possible to estimate the fundamental matrix from point correspondences. The essential matrix E is similar to the fundamental matrix, whereas here the influence of the calibration matrices is removed, e.g. $K = K' = I$. Equation (3.11) shows the calculation of the essential matrix from the relative pose R and $\mathbf{t}$ between two cameras. For three cameras a concept similar to the fundamental matrix exists, which is called trifocal tensor.

$$E = [\mathbf{t}]_{\times} R \tag{3.11}$$

3.3.4 Exterior Orientation Measurement

The position and orientation of the camera coordinate frame relative to the world coordinate frame is known as exterior orientation. It is usually measured by the combined usage of GPS and INS [6, 41, 45, 46]. Direct georeferencing applications use only these precisely determined exterior orientations, so depending on the accuracy requirements neither Ground Control Points (GCPs) nor a AT are necessary. The target product, e.g. a 3D reconstruction of a city, can be directly computed. The following describes the measurement of camera positions and orientations suitable for direct georeferencing using a strapdown INS, e.g. Applanix POS AV 610.

3.3.4.1 Working Principle

The INS consists of a IMU and a processing unit. While the IMU measures acceleration and orientation changes by accelerometers and gyros, the processing unit integrates the obtained IMU data to achieve position, velocity, and orientation angles relative to the world coordinate frame. A process called initial alignment is used to determine the north and vertical axis of the world frame, so that the integration starts at the correct values. The initial alignment should last about 15 to 20 minutes and requires that the aircraft does not move. It exists also an in-flight alignment, requiring that at least all 10 to 30 minutes a maneuver is carried out by the aircraft. INSs have the advantage of providing pose information at high rates and they are independent from any external signals. Furthermore relative positions and orientations within short time spans are very precisely determinable. A major drawback is, that sensor errors accumulate and therefore the INS frame drifts relative to the world frame. Errors in the INS are unbounded and in principle noise free. In opposite GPS receiver measure positions at lower rates and the errors in position and velocity are bounded but noisy. As can be seen, both systems are complementary and are therefore usually combined.

Fig. 3.20 depicts the relationship between GPS antenna, INS coordinate frame and camera coordinate frame. The vectorial position differences $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$ between the corresponding sensor origins are known as lever arm offsets. Here the subscript indicates the source coordinate frame and the target coordinate frame is given by the superscript. The origin of the GPS antenna is given by its phase center, while the origin of the INS

coordinate frame is determined by the intersection of the three IMU sensitivity axes. As mentioned above the origin of the camera coordinate frame is defined by the center of projection. In practice the INS axes and camera axes are not perfectly aligned, the remaining relative orientation R_{ins}^{cam} between them is known as boresight misalignment. Note that for the rotation matrix the same notation is applied as for the offsets. To achieve a good GPS signal quality, the GPS antenna is placed outside aircraft, preferable above the cameras center of projection. In contrast, the IMU is tightly mounted on the camera housing, at a small distance between IMU origin and center of projection.

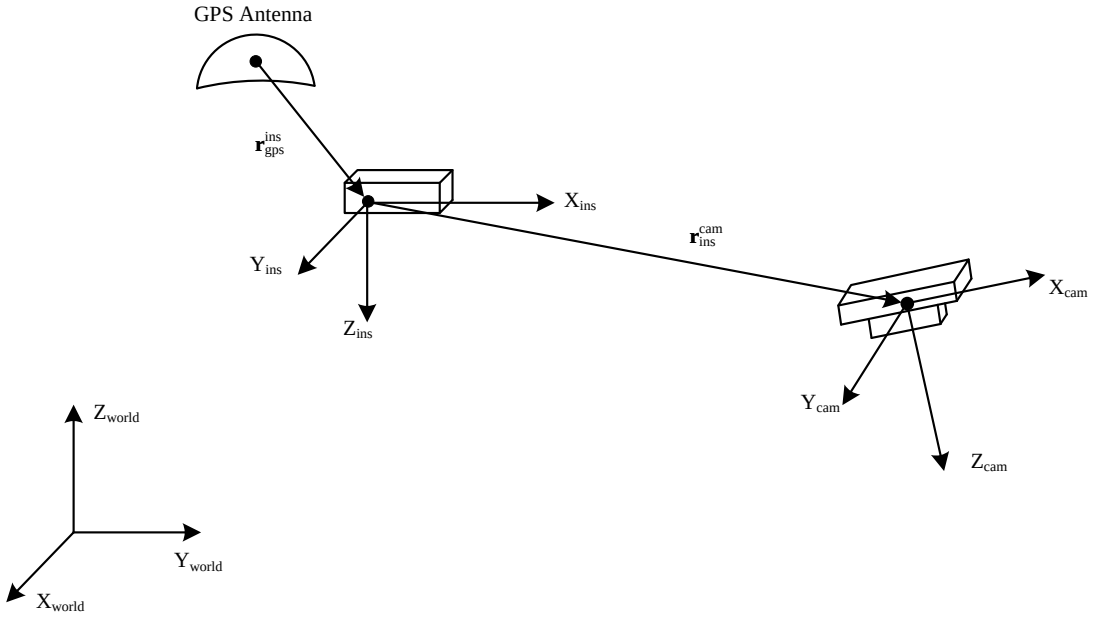


Figure 3.20: Relationship between GPS antenna, INS and camera coordinate frame, adapted from [41]. The lever arm offsets $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$ indicate the shifts in sensor origins.

A Differential Global Positioning System (DGPS) [3] improves the accuracy by using one or more stationary GPS receiver as base stations, whereas their locations have been previously surveyed, so they are known exactly. During a mapping mission each base station evaluates the GPS signal and correction data is transmitted to the aircraft in real time. By using this data in the GPS receiver of the airplane, the position uncertainties can be significantly reduced. Real Time Kinematic (RTK) is an improved version of the standard DGPS [47]. Beside sending the DGPS correction data in real time, it is also possible to store them at each base station and perform the evaluation later in a post processing

step after the flight.

Fig. 3.21 illustrates the processing steps for a combined evaluation of GPS and IMU data. The measured GPS and IMU data are usually forwarded to a Kalman Filter (KF), which estimates IMU errors by incorporating the GPS observations and outputs the corrected exterior orientation. Two different application scenarios are in use, the real time and the post processing scenario. While in real time processing the Kalman filtering is performed only forward in time, in post processing also a backward in time Kalman filtering is applied. The result of the post processing step is the smoothed best estimated flight trajectory [45–47].

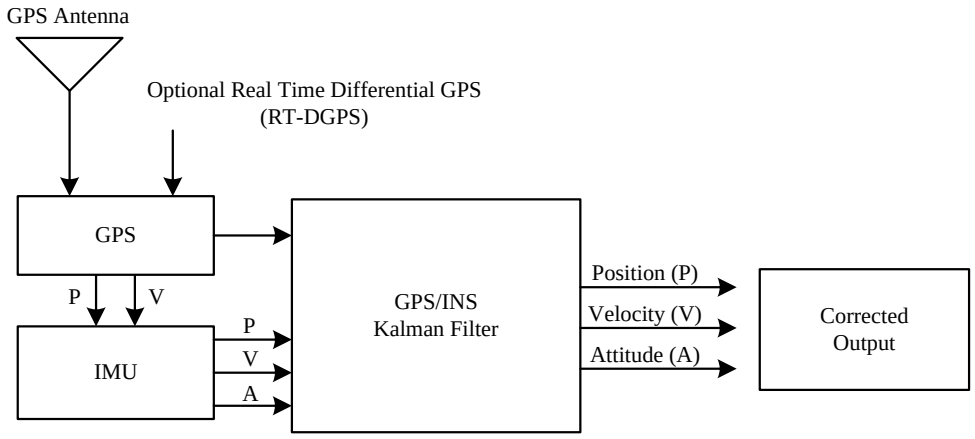


Figure 3.21: Combined GPS and IMU evaluation, from [41].

3.3.4.2 Accuracy

The achievable exterior orientation accuracy depends on the following factors [6, 47]:

- **Lever arm offset calibration**
  The lever arm offsets $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$ can be determined by measuring them, whereas $\mathbf{r}_{ins}^{cam}$ is usually given by the system manufacturer. Alternatively $\mathbf{r}_{gps}^{ins}$ can be estimated in real time or during post processing by the KF, which evaluates the IMU and GPS data. A further possibility is to introduce an additional parameter in the bundle adjustment step, although this approach is not optimal due to correlations with the adjusted camera positions. A wrong calibrated lever arm offset causes flight direction dependent errors.

- **Boresight misalignment calibration**
  The preferred way for calibrating the boresight misalignment is to use an additional

parameter during bundle adjustment.

- **Time synchronization between INS, GPS receiver and imaging device**
  Time synchronization is usually calibrated in lab environments, also a poor time synchronization causes flight direction dependent errors.

- **Initial and in-flight alignment quality**
  As mentioned above, a long INS initial alignment period and periodically performed flight maneuvers improve the obtained INS alignment accuracy.

- **Base station usage**
  The usage of base stations improves drastically the accuracy of the GPS position, see Table 3.3. These base stations can be either user operated or a already existing Continuously Operating Reference Station (CORS) network can be utilized. Note that the position accuracy drops with increasing distance between aircraft and nearest base station. So instead of real base stations one can also use a Virtual Reference Station (VRS) [47], which acts as if it would be closely located to the airplane. This is achieved by a set of sparsely distributed true reference stations and a server calculating the VRS and sending the correction data to the airplane.

- **GPS signal quality**
  Partial or full GPS signal outages can be caused by flying turns with bank angles larger than 40° or 50° respectively. Therefore additional errors can be induced in these cases, which depends on the utilized equipment and the therein implemented algorithms.

Table 3.3 gives the specification values of the Applanix POS AV 610 direct georeferencing system to illustrate the achievable exterior orientation accuracies of a GPS aided INS. Only evaluating the transmitted Coarse/Acquisition code from the satellites leads to large position errors. By the usage of base stations in standard DGPS and in the improved RTK mode the position uncertainties can be significantly reduced. Obviously the best result is obtained after post processing.

3.4 Feature Extraction

The feature extraction stage of the AT pipeline detects salient points in all images. Salient points are for example corners or blob like image regions. They can be repeatable detected

	Coarse/Acquisition Code GPS	DGPS	RTK	Post Processing
Position	4 m - 6 m	0.5 m - 2 m	0.1 m - 0.3 m	0.05 m - 0.3 m
Velocity	0.03 m/s	0.02 m/s	0.01 m/s	0.005 m/s
Roll & Pitch	0.005°	0.005°	0.005°	0.0025°
True Heading	0.03°	0.03°	0.02°	0.005°

Table 3.3: Applanix POS AV 610 absolute accuracy specification (RMS), from [45]. The accuracy values are given depending on the operation mode, whereas the values of the first three columns are achievable in real time.

in images of the same scene obtained from different camera poses. For example an interest point on the rooftop in one aerial image is very likely to be present in another aerial image displaying the same rooftop. Depending on the detector, it is possible to obtain the image positions with subpixel resolution. Some detectors return beside interest point locations also additional information like the image scale on which a point was detected, a orientation or a detector confidence.

A feature descriptor is often applied to ease or rather enable the feature matching between different images. It specifies the surrounding of a feature point by a vector. Note that a simple stacking of intensity values of the pixels within a patch centered at the interest point would lead to a poor matching performance. That is, because neither geometric distortions caused by different camera poses nor different illumination conditions are considered.

There exist many interest point detectors and descriptors e.g. [48–51], however in the following only SIFT features are used. This detector and descriptor combination has already been successfully applied on aerial images, e.g. in [2, 23, 52].

3.4.1 Scale-Invariant Feature Transform Detector

The SIFT detector [49] is covariant to image scale and rotation, meaning that the scale and orientation of the detection result varies with the scale and rotation of the investigated image to produce a consistent result. At the beginning of the detection procedure, the scale space $L(x, y, \sigma)$ of the given image $I(x, y)$ with coordinates x, y is generated for different discrete scales σ. That is achieved by convolving the image with a Gaussian kernel $G(x, y, \sigma)$ according to Equation (3.12). To speed up the scale space generation, the convolution is performed recursively, so that $L(x, y, \sigma)$ is used as input to compute $L(x, y, k\sigma)$.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \qquad G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3.12}$$

Then the convolution of the image with the DoG kernel $[G(x, y, k\sigma) - G(x, y, \sigma)]$ is determined for all considered scales σ, resulting in $D(x, y, \sigma)$. This can be efficiently performed for each scale by calculating the difference between neighboring scale space images $L(x, y, k\sigma)$ and $L(x, y, \sigma)$, see Equation (3.13). Note that the DoG approximates the scale-normalized Laplacian of Gaussian (LoG) $\sigma^2\nabla^2 G$, which is well suited for feature detection.

$$\begin{aligned} D(x, y, \sigma) &= [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{3.13}$$

Now locations $\mathbf{x} = [x, y, \sigma]^T$ with extrema of $D(\mathbf{x})$ are searched over the entire scale space. This search is performed by investigating the neighboring pixels of each possible location. An extremum is found, if all neighboring pixels have lower or higher values than the examined one, see Fig. 3.22.

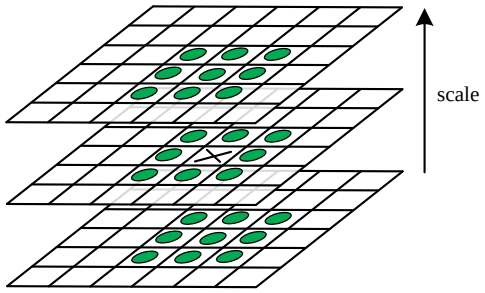


Figure 3.22: Scale space neighbors examined by the SIFT detector, adapted from [49]. The cross depicts the pixel to check for a maximum or minimum, while the circles indicate the 26 neighbors to be compared.

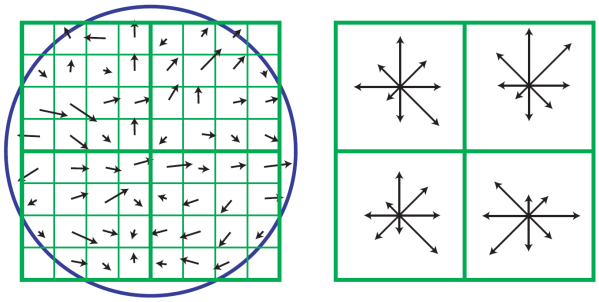
In the next step the positions of the maxima and minima are refined by evaluating a locally interpolated $D(\mathbf{x})$ leading to subpixel accurate locations $\hat{\mathbf{x}}$. Interest points with low $|D(\hat{\mathbf{x}})|$ or those which are likely to be an edge response are discarded. Finally the orientations of the obtained interest points are computed. Therefore an orientation histogram is built for each feature using the gradients of that $L(x, y, \sigma)$ with the scale σ nearest to the scale of the detected interest point. The orientation histogram accumulates

weighted gradient magnitudes according to their orientation, whereas the weight is given by a Gaussian window centered at the detection. Now a single feature point is reported at $\hat{\mathbf{x}}$ with an orientation according to the global maximum of the histogram. When local maxima with magnitudes similar to the global one exist, then further detections are reported for the same location with orientations corresponding to the local maxima.

3.4.2 Scale-Invariant Feature Transform Descriptor

The surrounding of a `SIFT` interest point is usually represented by a 128 dimensional `SIFT` descriptor [49]. For each feature point multiple orientation histograms are generated, then concatenated to a single descriptor vector and finally normalized to reduce the sensitivity to illumination changes. In the following the distinct steps carried out for a single feature point are stated more precisely.

At the beginning a patch is centered at the location of an interest point, which is further subdivided into smaller non overlapping regions. Within each region the weighted gradients are used to build an orientation histogram, whereas the weighting is performed according to a Gaussian kernel centered at the interest point, see Fig. 3.23. This is performed at the detected scale, and the individual histograms are orientated according to the detected feature point orientation.



(a) (b)

Figure 3.23: `SIFT` descriptor, from [49]. In (a) a 8×8 pixels patch with associated gradient vectors is illustrated. Furthermore a Gaussian window is symbolized by a blue circle. The subdivision of the patch into 2×2 orientation histograms is depicted in (b). Each histogram has 8 orientation bins.

Then the individual histograms are concatenated to a single descriptor vector, which is normalized to unit length. Afterwards the descriptor vector elements are limited to

0.2 and the whole vector is renormalized. In contrast to Fig. 3.23, usually 16×16 pixels patches are used to describe the surrounding of an interest point. This results in 4×4 histograms each having 8 orientations leading to a 128 dimensional descriptor vector.

3.4.3 Feature Reduction Strategies

Sometimes it is useful to reduce the amount of features to lower the computational effort in finding feature correspondences between two images. An example where it is sufficient to use only a subset of the generated features is the determination of views showing the same scene content in a set of unordered images. The following list gives an overview of feature reduction methods [19].

- **Image shrinking**
  Instead of using the full resolution images the detector is applied on images of reduced size. As a result only features at a higher scale are obtained and their location accuracy is lowered. An equivalent measure for the `SIFT` detector would be to change the parameterization so that all scale space images $L(x, y, \sigma)$ with a scale σ below a certain value are ignored.

- **Modification of the detector parameterization**
  Also the feature detector parameterization can be changed to obtain fewer features. In the case of the `SIFT` detector the threshold for suppressing weak interest points with low $|D(\hat{\mathbf{x}})|$ can be increased to obtain stronger interest points. On the one hand these feature points are likely to be detected in different images showing the same part of a scene, on the other hand interest points containing valuable information may be suppressed. For example consider aerial images, where cars usually cause strong interest points due to the high contrast to their surrounding, while old road markings raise weak interest points. In opposite to road markings, cars are movable objects and therefore not suited for `AT`.

- **Detector result filtering**
  Additional feature detector output data like the scale of an interest point may be exploited to perform filtering after feature detection. For example instead of changing the detector parameterization to obtain only features of a certain scale range the same effect is achieved, when the scale filtering is performed on the detection result. This approach has the advantage, that the detector must only run once when different feature set sizes of the same image are required.

- **Descriptor filtering**

  The method proposed by [19] reduces the features in query and testing image by restricting the allowed value range for each descriptor dimension, see Fig. 3.24. This approach works only, when the centers of the descriptor vectors in the distinct images are similar.

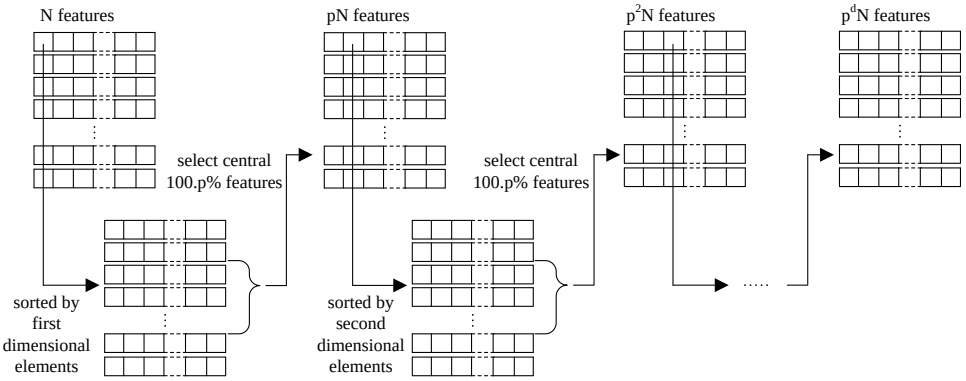


Figure 3.24: Feature reduction by descriptor filtering, adapted from [19]. To reduce the amount of features the descriptor vectors are sorted along one dimension and all features, except the central $100p\%$, are discarded, where p is in the range $[0, 1]$. That is repeated for all dimensions.

- **Bucketing technique**

  A further method is to select randomly a subset of features in the query image which is then matched against a testing image containing all features. For some tasks it is required that the randomly selected features have a certain distance to each other. This can be achieved by applying the bucketing technique proposed in [53]. The query image is overlain by a grid where each grid cell represents a bucket, see Fig. 3.25(a). A feature is selected by first randomly selecting a bucket and then randomly choosing a feature within the bucket. As illustrated in Fig. 3.25(b) the buckets are selected with a probability corresponding to the number of the therein contained features, empty buckets are not considered.

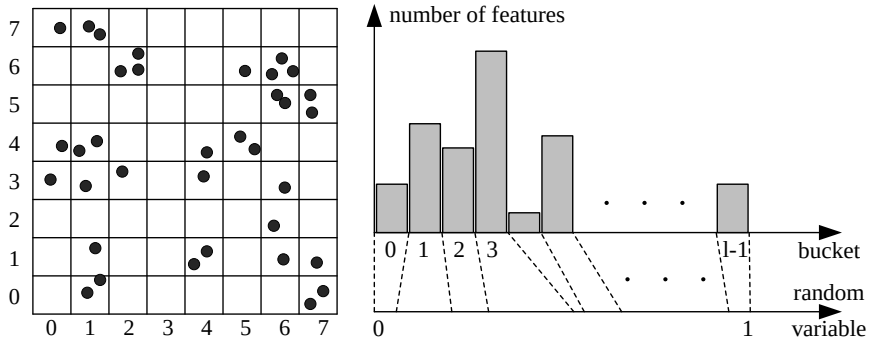


Figure 3.25: Bucketing technique for feature reduction, adapted from [53]. In (a) the grid overlaying a query image is shown. Each grid cell corresponds to a bucket. The feature points are indicated by black dots. For each bucket the number of the contained features are counted and a uniform distributed random variable in the range $[0, 1]$ selects a bucket with a probability according to its feature count, see (b).

3.5 View Selection

The goal of the view selection stage in the `AT` pipeline is to reduce the effort in the subsequent feature matching stage by determining for each image all other images showing the same part of the scene. Then the following feature matching stage has not to match each image with all other images, which results in a significant performance gain. One possible approach for view selection is to compare the image content of one image with all other images, for example by evaluating their features. To achieve an overall performance improvement only a small feature subset is evaluated [19, 22, 23]. Another possibility is to exploit additional information like camera position and orientation from `GPS` and `IMU` measurements [21].

3.5.1 View Similarity Measure

As suggested above one way to determine views showing the same part of the scene is to detect features for each image and perform a pairwise matching of them [19]. From the obtained matches, a view similarity value $S_{im}(\mathcal{V}_i, \mathcal{V}_j)$ between views $\mathcal{V}_i$ and $\mathcal{V}_j$ is calculated according to Equation (3.14). The first term relates the number of matched features $N(\mathcal{V}_i, \mathcal{V}_j)$ between view $\mathcal{V}_i$ and view $\mathcal{V}_j$ to the maximum number of matched features over all image pairs N_{max}. The second term determines the reliability of the matches by

evaluating the mean position difference $\bar{d}(\mathcal{V}_i, \mathcal{V}_j)$ between a feature in view $\mathcal{V}_i$ and the corresponding feature in view $\mathcal{V}_j$, $\bar{d}_{max}$ is the maximum mean position difference over all view pairs, α relates both terms and is in the range $[0, 1]$. The match reliability is high, when the mean feature distance $\bar{d}(\mathcal{V}_i, \mathcal{V}_j)$ between both views is low, which indicates similar images. Furthermore a feature reduction technique can be used to lower the computational effort. Also other measures have been applied for view selection, see for example [22].

$$S_{im}(\mathcal{V}_i, \mathcal{V}_j) = \alpha \frac{N(\mathcal{V}_i, \mathcal{V}_j)}{N_{max}} + (1 - \alpha) \frac{\bar{d}_{max} - \bar{d}(\mathcal{V}_i, \mathcal{V}_j)}{\bar{d}_{max}} \tag{3.14}$$

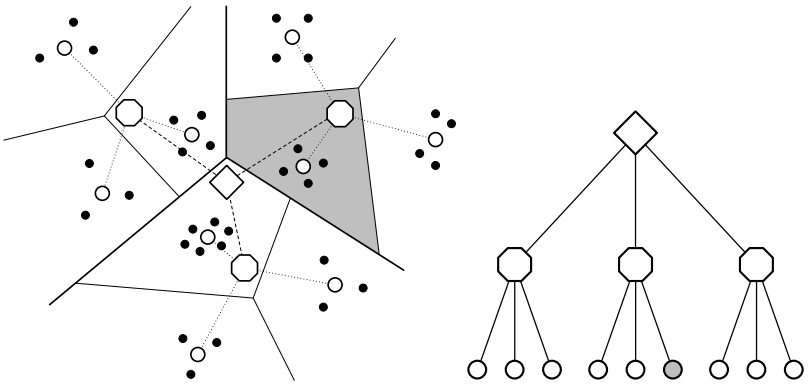
3.5.2 Vocabulary Tree

A vocabulary tree [54] is an efficient realization of the Bag of Words (`BoW`) [55] concept which is used for image classification. It can be applied to compare the content of one image with all others and therefore finds views showing the same part of a scene [21].

At first the `BoW` model is briefly explained. Here a set of training images is given, each containing one object. The task is now to determine the object displayed in a testing image. To achieve this, features are extracted from the training images and similar descriptors are clustered. The clustering process can be performed for example with the k-means algorithm and is also known as vector quantization. Each cluster center represents a so called visual word, so one training image can be described by a histogram counting the occurrences of such words. Also the testing image can be described by such a histogram using the previously learned quantization. Now the histogram of the testing image is compared with the histograms of all training images. The testing image is classified according to the most similar training image, which passes a final geometric verification step. Beside simple histograms, also other representations of visual word collections are possible.

In the case of a vocabulary tree the vector quantization is performed by a hierarchical k-means tree, which is built in the following way. All descriptors of all training images are subdivided into k clusters by applying the k-means algorithm. Then each cluster is recursively split into k smaller ones. Splitting stops when the configured hierarchy level L is reached. Each node in the hierarchical k-means tree represents a cluster center at the corresponding level. Fig. 3.26 illustrates a two dimensional hierarchical k-means tree generated with two levels and a branching factor $k = 3$.

Now the descriptors of each training image are quantized by the learned tree. During this procedure each descriptor takes a certain path through the tree. The sum of the paths



(a) (b)

Figure 3.26: Hierarchical k-means tree built from two dimensional point data, adapted from [54]. The solid lines in (a) illustrate the hierarchical splitting of the two dimensional space into disjoint regions. Each region of each subdivision level corresponds to a single node in the resulting tree, which is given in (b).

obtained from the descriptors of one image is called a population of the tree. To classify a testing image its descriptors are also passed through the tree. Tree populations between testing and training images are compared and the classification is performed according to the most similar population. This comparison can be performed efficiently. A variant of the algorithm compares the population only at leaf nodes, which is essentially the histogram comparison in the BoW model.

3.5.3 View Overlap Criterion

Are the poses of the used cameras and the scene known, then overlapping views can be calculated without evaluating the image content [21]. For example poses can be measured by GPS/INS, while the scene might be approximated by a ground plane or a DEM. In this cases the image rectangle of a view j is back projected onto the scene leading to a region R_j, which is subsequently projected into view i. The overlap between the image rectangle of view i and the projection of R_j into the same view can be used to identify cameras sharing same content.

For unknown scenes one can assume a maximum scene depth S for each camera. Then the scene is approximated for view i by a plane $\mathbf{\Pi}^i$ parallel to the image plane, placed at a distance S in front of camera i, see Fig. 3.27. The image rectangles of camera i and another camera j are projected onto $\mathbf{\Pi}^i$ leading to the regions R_i and R_j respectively.

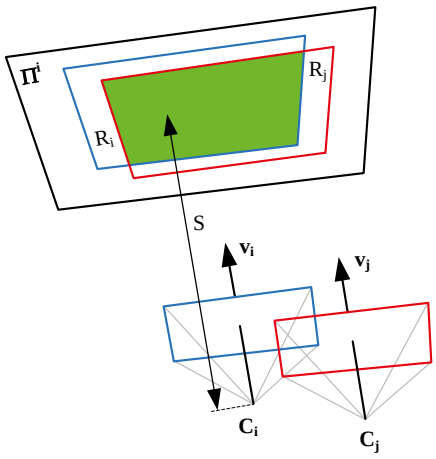


Figure 3.27: View overlap between two cameras, adapted from [21]. The scene is approximated for camera i with a plane $\mathbf{\Pi}^i$ at a distance S parallel to the image plane of the same camera. The measured camera centers are denoted with $\mathbf{C}_i$ and $\mathbf{C}_j$ respectively. Furthermore the corresponding view vectors are given with $\mathbf{v}_i$ and $\mathbf{v}_j$. To determine the view overlap, the projected image rectangles $R_i$and R_j are evaluated.

Now the overlap ratio $\mathcal{O}_j^i$ between camera i and j is calculated according to Equation (3.15), where $a(\cdot)$ denotes the area of a region. So in the feature matching stage only view pairs with an $\mathcal{O}_j^i$ value above a certain threshold are considered. Furthermore it is required that the angle between the view vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ of both cameras is below another threshold, e.g. for `SIFT` $\approx 30°$, otherwise feature matching would fail anyway.

$$\mathcal{O}_j^i = \frac{a(R_i \cap R_j)}{a(R_i \cup R_j)} \tag{3.15}$$

3.6 Feature Matching

The feature matching stage establishes feature correspondences between a query image and a testing image. These correspondences can be used to find neighboring views in an unordered set of views. Furthermore they are required to generate feature tracks across multiple images, which are used to triangulate world points in the 3D structure computation stage of the `AT` or `SfM` pipeline [19, 20].

One strategy is to exhaustively compare the descriptor vectors of the query image with the descriptor vectors of the testing image. Correspondences are generated between those features, which have the most similar descriptors measured by some metric. Afterwards the matches are verified against some expectation to remove outliers. Obviously, for large

feature sets the computational effort for comparing each feature from one image with all features of another image is high. So more efficient methods have been developed. This section first gives an overview of efficient feature matching algorithms and then presents methods to verify the obtained matches.

3.6.1 Approximate Nearest Neighbor Search

A common way to perform exact Nearest Neighbor (NN) search in low dimensional space is to utilize the above introduced kD tree [27], however the applied tree is built slightly different. Instead of performing the split in a fixed order by selecting cyclically one dimension after another, here the dimension along which the feature descriptors have the highest variance is chosen at each level. Recall that the leaf nodes in the kD tree correspond to cells in the k dimensional space and contain exactly one point.

In the first step for obtaining the NN of a query point q the cell containing q is searched by traversing the tree top down. Then the distance between q and the point stored within the cell is computed. Now all neighboring cells within this distance are visited by backtracking to find a possible closer point. This is performed by using a priority search, which examines the nearest cells first. The priority search is realized by a priority queue, which stores the neighboring cells discovered during the search as well as their distances to the query point q. As soon as a new potential NN is found, the search radius is reduced to the distance between q and the new best match, see Fig. 3.28. When all cells within the actual search radius have been visited, the search terminates and the algorithm is sure that the true NN has been found.

The outlined method has the disadvantage that in high dimensional feature spaces the obtained performance degrades to nearly linear in the number of feature points due to the high number of nodes needed to be investigated. To reduce the query time one can apply Approximate Nearest Neighbor (ANN) methods which return the true NN only with a configurable probability. In the case of kD trees one can achieve a performance improvement by terminating the search either if the algorithm is sure that the true NN is found or after a certain maximum amount of nodes has been explored. In [28] different algorithms are compared and they found that the randomized kD trees algorithm and the hierarchical k-means tree method perform best among the tested. Both are briefly introduced in the following sections.

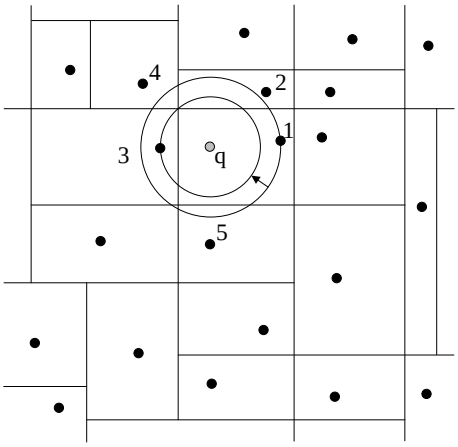


Figure 3.28: Priority search in a two dimensional kD tree, adapted from [27]. The distance between a query point q and the point within the same cell is used to initialize the search region. During backtracking the search region is updated to the actual best match.

3.6.1.1 Randomized kD Trees

The randomized kD trees [27] algorithm uses multiple randomly generated kD trees and aborts the search after the maximum number of examined cells has been reached. Let's consider the following example to reason why multiple randomly generated kD trees improve the search performance compared to one single tree. If one has a feature database of $N = 2^{20} \approx 1000000$ features with 128 dimensional descriptors, then the resulting kD tree has a depth of $\log_2(N) = 20$. That means the starting cell for backtracking is found by comparing only 20 descriptor values, while the remaining dimensions are completely ignored. This can be approximately seen as projecting the 128 dimensional vector into a 20 dimensional space and then searching within this space. The true `NN` is now not necessarily the `NN` in the lower dimensional space, because other points might be projected closer to the query location. So depending on the underlying projection many points may have to be investigated until the true `NN` is found. Now by using multiple trees with different random projections, the probability that a tree is built in which the true `NN` is close to the starting point increases. That is, because the true `NN` is always relative close to the starting point, independently from the applied projection.

Trees with different random projections can be constructed by randomly rotating the feature descriptors in the high dimensional space before constructing the tree. As mentioned above the applied kD-tree splits the data along the dimension at which the descriptor subset has the highest variance. Usually always more than one dimension exist

with high variance. Therefore trees with different random projections can be also built by randomly selecting a split dimension from those with high variance.

For handling multiple trees the search procedure is modified in the following way. At first the cell corresponding to the query point q is determined for each tree. Then the obtained cells with their true distances between the interior point and q are inserted into one common priority queue. The radius of the priority search is initialized with the smallest distance stored in the queue and the search starts at that tree, at which the smallest distance has been discovered. Now backtracking always continues on the tree, to which the cell with the smallest distance to q belongs, found in the queue. Also here the search terminates if either the true `NN` is surely found or the maximum number of explored nodes is reached.

The performance can be further improved by applying a Principal Component Analysis (`PCA`) on the descriptors before building the kD trees. It orientates the coordinate axis with the principal axes. In this case any subsequent random rotation should be performed within the subspace of the principal axis with the largest principal moments.

3.6.1.2 Hierarchical k-Means Tree

A hierarchical k-means tree can be utilized in a similar way for `ANN` searching [28]. Instead of multiple kD-trees a single hierarchical k-means tree is built. For that purpose the feature set is split into k clusters using the k-means algorithm. Now each cluster is recursively subdivided into k smaller clusters until they contain less than k points. From the obtained hierarchical partitioning of the space a tree is built, where each node corresponds to one cluster. Fig. 3.26 illustrates the partitioning on the basis of a two dimensional space for $k = 3$ and shows the resulting tree after two clustering steps. It can be observed that each node in the tree corresponds to a cell within the two dimensional space.

At the beginning of the `ANN` search the cell belonging to the query point q is determined by traversing the tree top down. The discovered cells along the path with their distance between the corresponding cell center and q are added to a priority queue. The initial radius for the following priority search is the smallest distance between one point in the determined leaf node and the query point q. Now the `ANN` is searched by backtracking using always the cell with the center nearest to q from the queue. The search stops after a certain amount of cells are examined, depending on the required approximation accuracy. In contrast to above, multiple randomized k-means trees don't improve the query performance.

3.6.2 Simultaneous Localization and Mapping Methods

SLAM is the problem of localizing a vehicle or robot and at the same time building a map of the environment by the usage of some observations. In Visual SLAM these observations are images of the scene in which a robot or camera operates. A subtask in Visual SLAM [10] is to refine an already uncertain available camera pose by first matching the current observed features with an existing map of features and then updating the camera state, see Fig. 3.29. This feature matching problem is similar to that occurring in the considered AT pipeline, where also the camera poses are uncertainly known. Therefore some of these matching methods are presented in the following.

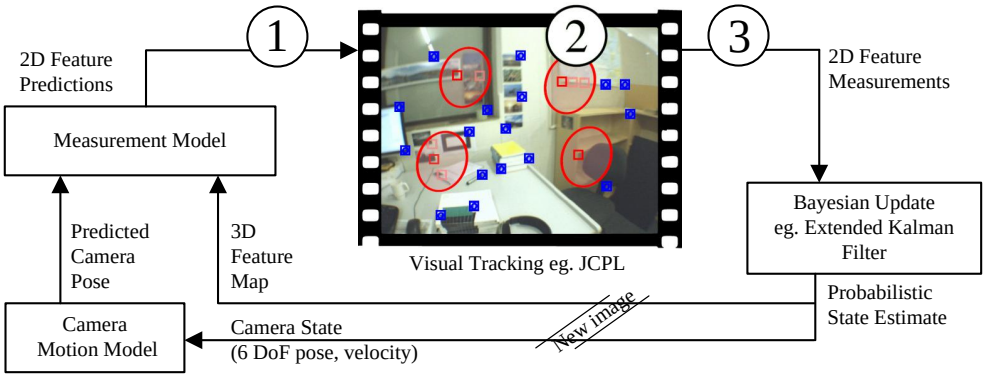


Figure 3.29: Visual SLAM processing diagram, adapted from [10]. The predicted camera pose and the 3D feature map are used to generate 2D feature predictions ①. These predictions are applied in the guided feature matching step ② leading to the matching result ③. A Bayesian update step actualizes the 3D feature map and the camera state for the current image. Now a camera motion model generates a prediction for the next image.

3.6.2.1 Joint Compatibility Branch and Bound

Joint Compatibility Branch and Bound (JCBB) [13] has been developed for that part of the SLAM problem, where m measurements $\{E_1, \ldots, E_m\}$ have to be assigned to n landmarks $\{F_1, \ldots, F_n\}$. It searches those correspondence hypothesis $\mathcal{H}_m = \{j_1, \ldots, j_m\}$ which assigns the highest number of measurements $\{E_1, \ldots, E_m\}$ to landmarks $\{F_{j_1}, \ldots, F_{j_m}\}$. Spurious measurements are assigned to F_0 and not counted. In general it is allowed that multiple measurements are assigned to one landmark resulting in an exponential hypothesis space. This hypothesis space can be represented by an interpretation tree in which each node represents an assignment of one measurement E_i to one landmark F_{j_i}. The

Branch and Bound (BB) search algorithm traverses this tree, concentrating on the most promising hypotheses and early discarding those establishing too few correspondences.

For the traversal of the interpretation tree it is required to generate hypotheses $\mathcal{H}_i = \{j_1, \ldots, j_i\}$ involving only the first i measurements. The assignment problem for one specific hypothesis $\mathcal{H}_i$ is formulated in a probabilistic way using the implicit joint measurement function $\mathbf{f}_{\mathcal{H}_i}(\mathbf{x}, \mathbf{y}) = 0$, which depends on the systems actual state $\mathbf{x}$ and on idealized measurements $\mathbf{y}$. Systems state $\mathbf{x}$ is composed of the true robot pose $\mathbf{x}_R$ and the true landmark positions $\mathbf{x}_{F_1}, \ldots, \mathbf{x}_{F_n}$, the idealized measurement vector $\mathbf{y}$ contains each observations location. Now the implicit joint measurement function states, that the position differences between corresponding idealized observations and true landmark positions must be zero, when the true robot pose is accounted. In reality only a state estimate $\hat{\mathbf{x}}$ with covariance $\Sigma_{\mathbf{x}}$ and a noisy measurement vector $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{u}$ with noise $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{u}})$ and covariance $\Sigma_{\mathbf{y}} = \Sigma_{\mathbf{u}}$ are available, see Equation (3.16).

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_R \\ \hat{\mathbf{x}}_{F_1} \\ \vdots \\ \hat{\mathbf{x}}_{F_n} \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} \hat{\mathbf{y}}_{E_1} \\ \vdots \\ \hat{\mathbf{y}}_{E_m} \end{bmatrix} \tag{3.16}$$

To simplify the evaluation of the joint measurement function a linearization at the current state estimate $\hat{\mathbf{x}}$ and at the current measurement $\hat{\mathbf{y}}$ is performed according to Equation (3.17).

$$\mathbf{f}_{\mathcal{H}_i}(\mathbf{x}, \mathbf{y}) \simeq \mathbf{v}_{\mathcal{H}_i} + F_{\mathcal{H}_i}(\mathbf{x} - \hat{\mathbf{x}}) + G_{\mathcal{H}_i}(\mathbf{y} - \hat{\mathbf{y}}) \tag{3.17}$$

with

$$\mathbf{v}_{\mathcal{H}_i} = \mathbf{f}_{\mathcal{H}_i}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \qquad F_{\mathcal{H}_i} = \left.\frac{\partial \mathbf{f}_{\mathcal{H}_i}}{\partial \mathbf{x}}\right|_{(\hat{\mathbf{x}}, \hat{\mathbf{y}})} \qquad G_{\mathcal{H}_i} = \left.\frac{\partial \mathbf{f}_{\mathcal{H}_i}}{\partial \mathbf{y}}\right|_{(\hat{\mathbf{x}}, \hat{\mathbf{y}})}$$

For checking, whether a hypothesis is possible, the Mahalanobis distance $D^2_{\mathcal{H}_i}$ is calculated by utilizing the inverse joint innovation covariance $\Sigma^{-1}_{\mathbf{v}_{\mathcal{H}_i}}$, which can be built efficiently using the incremental approach given in [13]. The test in Equation (3.18) ensures Joint Compatibility (JC) of the correspondences in $\mathcal{H}_i$, whereas α denotes the confidence level and $d = \dim(\mathbf{f}_{\mathcal{H}_i})$. If it is passed, the number of assignments in the hypothesis is used as quality measure for the BB algorithm, otherwise the hypothesis is discarded.

$$D^2_{\mathcal{H}_i} = \mathbf{v}^T_{\mathcal{H}_i} \Sigma^{-1}_{\mathbf{v}_{\mathcal{H}_i}} \mathbf{v}_{\mathcal{H}_i} < \chi^2_{d,\alpha} \tag{3.18}$$

JCBB does not necessarily find the JC set with the smallest Mahalanobis distance, because the search stops as soon as the largest JC set is found. The proposed extension in [10] continues searching until the best and largest JC hypothesis is found, which is essential in difficult tracking applications.

In opposite to the Individual Compatibility Nearest Neighbor (ICNN) algorithm, which assigns measurements independently of the others to landmarks, JCBB considers the correlation between measurements and therefore produces more robust results in noisy environments. Also the Sequential Compatibility Nearest Neighbor (SCNN) approach accounts for correlations between observations by sequentially establishing correspondences between measurements and landmarks with regard to all previous made assignments. But this strict sequential behavior limits the hypotheses search too much, so that JCBB outperforms SCNN. The main disadvantage of JCBB is the higher computational effort compared to ICNN and SCNN.

3.6.2.2 Joint Compatible Pair Linking

Following [10], the above matching problem can be reformulated in image space by calculating the estimated image location $\hat{\mathbf{z}}_{F_j}$ for landmark F_j with measurement model $\mathbf{h}_{F_j}$ according to Equation (3.19). Instead of the robot pose $\hat{\mathbf{x}}_R$, the camera pose $\hat{\mathbf{x}}_C$ is used here. The resulting location uncertainty $\Sigma_{\mathbf{z}_{F_j}}$ is composed of one term resulting from the state estimate uncertainty $\Sigma_{\mathbf{x}}$ and another term $\Sigma_{\mathbf{r}_{F_j}}$ handling model imprecision by adding White Gaussian Noise (WGN) $\mathcal{N}(0, \Sigma_{\mathbf{r}_{F_j}})$.

$$\hat{\mathbf{z}}_{F_j} = \mathbf{h}_{F_j}(\hat{\mathbf{x}}_C, \hat{\mathbf{x}}_{F_j}) \qquad \Sigma_{\mathbf{z}_{F_j}} = H_{F_j} \Sigma_{\mathbf{x}} H^T_{F_j} + \Sigma_{\mathbf{r}_{F_j}} \qquad H_{F_j} = \left. \frac{\partial \mathbf{h}_{F_j}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}} \tag{3.19}$$

In contrast to the previous section $\hat{\mathbf{y}}_{E_i}$ describes the measurement E_i in image coordinates, resulting in a deviation $\mathbf{v}_{ij} = \hat{\mathbf{z}}_{F_j} - \hat{\mathbf{y}}_{E_i}$ from the expected landmark position F_j. For checking the Individual Compatibility (IC) of assigning observation E_i to landmark F_j the following test is applied, as defined above $\Sigma_{\mathbf{y}_{E_i}}$ denotes the measurement noise covariance.

$$D^2_{ij} = \mathbf{v}^T_{ij} \Sigma^{-1}_{\mathbf{v}_{ij}} \mathbf{v}_{ij} < \chi^2_{2,\alpha} \qquad \Sigma_{\mathbf{v}_{ij}} = \Sigma_{\mathbf{z}_{F_j}} + \Sigma_{\mathbf{y}_{E_i}} \tag{3.20}$$

The vector $\hat{\mathbf{z}}$ containing the estimated image locations of all landmarks $\{F_1, \ldots, F_n\}$ is built by stacking the image location estimates $\hat{\mathbf{z}}_{F_i}$. Then the resulting location uncertainty $\Sigma_{\mathbf{z}}$ can be calculated according to Equation (3.21), where $\Sigma_{\mathbf{r}} = \text{diag}(\Sigma_{\mathbf{r}_{F_1}}, \ldots, \Sigma_{\mathbf{r}_{F_n}})$ considers any model imprecision.

$$\hat{\mathbf{z}} = \mathbf{h}(\hat{\mathbf{x}}) = \begin{bmatrix} \mathbf{h}_{F_1}(\hat{\mathbf{x}}_C, \hat{\mathbf{x}}_{F_1}) \\ \vdots \\ \mathbf{h}_{F_n}(\hat{\mathbf{x}}_C, \hat{\mathbf{x}}_{F_n}) \end{bmatrix} \qquad \Sigma_{\mathbf{z}} = H\Sigma_{\mathbf{x}}H^T + \Sigma_{\mathbf{r}} \qquad H = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}} \tag{3.21}$$

Now a hypothesis $\mathcal{H} = \{i_1, \ldots, i_n\}$ for assigning measurements E_{i_j} to landmarks F_j leads to the subset $\hat{\mathbf{y}}_{\mathcal{H}}$ of measurements with corresponding covariance $\Sigma_{\mathbf{y}_{\mathcal{H}}}$ as given in Equation (3.22).

$$\hat{\mathbf{y}}_{\mathcal{H}} = \begin{bmatrix} \hat{\mathbf{y}}_{E_{i_1}} \\ \vdots \\ \hat{\mathbf{y}}_{E_{i_n}} \end{bmatrix} \qquad \Sigma_{\mathbf{y}_{\mathcal{H}}} = \text{diag}(\Sigma_{\mathbf{y}_{E_{i_1}}}, \ldots, \Sigma_{\mathbf{y}_{E_{i_n}}}) \tag{3.22}$$

To test the JC of a the hypothesis $\mathcal{H}$, the Mahalanobis distance $D^2_{\mathcal{H}}$ is evaluated for the deviation $\mathbf{v}_{\mathcal{H}} = \hat{\mathbf{z}} - \hat{\mathbf{y}}_{\mathcal{H}}$ according to Equation (3.23).

$$D^2_{\mathcal{H}} = \mathbf{v}^T_{\mathcal{H}}\Sigma^{-1}_{\mathbf{v}_{\mathcal{H}}}\mathbf{v}_{\mathcal{H}} < \chi^2_{d,\alpha} \qquad \Sigma_{\mathbf{v}_{\mathcal{H}}} = \Sigma_{\mathbf{z}} + \Sigma_{\mathbf{y}_{\mathcal{H}}} \tag{3.23}$$

Instead of modeling the measurement noise $\Sigma_{\mathbf{y}_{E_i}}$ and model imprecision $\Sigma_{\mathbf{r}_{F_j}}$ separately, these influences are usually summarized in $\Sigma_{\mathbf{r}_{F_j}}$, which is also assumed in the following, e.g. $\Sigma_{\mathbf{y}_{E_i}} = 0$.

For early discarding wrong hypotheses one can take advantage of the monotonicity property of the Mahalanobis distance D^2, which states that D^2 cannot decrease, if the length of a hypothesis is increased [10]. For example $D^2_{\mathcal{H}_a} \leq D^2_{\mathcal{H}_b}$ is always true for the hypotheses $\mathcal{H}_a = \{i_1, i_2\}$ and $\mathcal{H}_b = \{i_1, i_2, i_3\}$.

Joint Compatible Pair Linking (JCPL) [10] is proposed within a three stage tracking algorithm based on Visual SLAM. In the first stage the landmarks are split into two groups, a small primary set and a larger secondary set by applying a heuristic rule. Within the second stage JCPL searches the largest JC hypothesis with the lowest Mahalanobis distance for the small primary set. This hypothesis is determined by first testing JC for all landmark pair combinations of the primary set. Afterwards the final hypothesis is found

by combining the pairwise results and exploiting the monotonicity property of D^2. In the third stage the vector with estimated landmark image coordinates $\hat{\mathbf{z}}$ is sorted into a part $\hat{\mathbf{z}}_p$ containing the expected positions of the primary landmarks and a part $\hat{\mathbf{z}}_s$ with the expected positions of the secondary landmarks. Also the covariance matrix $\Sigma_{\mathbf{z}}$ is rearranged, see Equation (3.24).

$$p(\mathbf{z}) \sim \mathcal{N}(\hat{\mathbf{z}}, \Sigma_{\mathbf{z}}) = \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{z}}_p \\ \hat{\mathbf{z}}_s\end{bmatrix}, \begin{bmatrix}\Sigma_{\mathbf{z}pp} & \Sigma_{\mathbf{z}ps} \\ \Sigma_{\mathbf{z}sp} & \Sigma_{\mathbf{z}ss}\end{bmatrix}\right) \tag{3.24}$$

Now the matching results of the primary features are used to reduce the uncertainty about the secondary landmarks image location by applying Equation (3.25), where vector $\mathbf{z}_p$ contains the measured positions of the primary landmarks. Vector $\hat{\mathbf{z}}_{s|p}$ provides the estimated positions of the secondary landmarks given the matching result of the primary ones, the final uncertainty regions are specified by $\Sigma_{\mathbf{z}\,s|p}$. Finally the best `IC` measurements within the reduced search regions are chosen for each secondary landmark.

$$\begin{aligned} p(\mathbf{z}_s|\mathbf{z}_p) &\sim \mathcal{N}(\hat{\mathbf{z}}_{s|p}, \Sigma_{\mathbf{z}\,s|p}) \\ \hat{\mathbf{z}}_{s|p} &= \hat{\mathbf{z}}_s + \Sigma_{\mathbf{z}sp}\Sigma_{\mathbf{z}pp}^{-1}(\mathbf{z}_p - \hat{\mathbf{z}}_p) \\ \Sigma_{\mathbf{z}\,s|p} &= \Sigma_{\mathbf{z}ss} - \Sigma_{\mathbf{z}sp}\Sigma_{\mathbf{z}pp}^{-1}\Sigma_{\mathbf{z}ps} \end{aligned} \tag{3.25}$$

This algorithm improves the prediction of secondary landmark locations by exploiting correlations between primary and secondary landmark observations. Such correlations exist at least, because they are observed by the same camera, whose pose is not exactly known.

3.6.2.3 Active Matching

Active Matching (`AM`) [12] sequentially searches the landmarks uncertainty regions in image space for establishing correspondences with at most one measurement per landmark. After each search step the uncertainty regions of the remaining landmarks are updated. The search is guided by an information efficiency score, which is designed to minimize the effort for establishing a global consensus set. More precisely the score is built by considering how much the correspondence search on one landmark is expected to reduce the search region sizes of the other landmarks and how many image operations are required for this search.

`AM` maintains multiple hypotheses, each one is represented by a multivariate Gaussian G_k with weight λ_k, all K hypotheses form the Gaussian Mixture Model (`GMM`) given in

Equation (3.26). Initially the GMM consists only of one hypothesis formed by the estimated prior distribution, for example obtained from an Extended Kalman Filter (EKF).

$$p(\mathbf{x}) = \sum_{k=1}^{K} \lambda_k G(\hat{\mathbf{x}}_k, \Sigma_{\mathbf{x}_k}) = \sum_{k=1}^{K} \lambda_k G_k \qquad \mathbf{x} = \begin{bmatrix} \mathbf{x}_C \\ \mathbf{x}_{F_1} \\ \vdots \\ \mathbf{x}_{F_n} \\ \mathbf{z}_{F_1} = \mathbf{h}_{F_1}(\mathbf{x}_C, \mathbf{x}_{F_1}) \\ \vdots \\ \mathbf{z}_{F_n} = \mathbf{h}_{F_n}(\mathbf{x}_C, \mathbf{x}_{F_n}) \end{bmatrix} \tag{3.26}$$

Each measurement $\mathbf{z}_C$ provides a likelihood $p(\mathbf{z}_C|\mathbf{x})$, indicating the probability for a landmark F_i to be located at a specific position given the system state $\mathbf{x}$. This likelihood is also modeled by a mixture of multivariate Gaussians H_m depending on $\mathbf{x}$. Each single Gaussian H_m represents one found match in the search region. In addition two uniform distributions are used, one with height μ_{out} to model the case that the true match is outside the search region and one with height μ_{in} for modeling that the true match is within the search region but not found, see Fig. 3.30 for a 1D illustration.

A new refined system state $p(\mathbf{x}|\mathbf{z}_C)$ is calculated from the measurement $p(\mathbf{z}_C|\mathbf{x})$ and the actual system state $p(\mathbf{x})$ by approximately evaluating Bayes' Rule according Equation (3.27). The term $p(\mathbf{z}_C)$ performs a normalization of the weights λ_k, so that they sum up to one. For M matches within the search region, the resulting GMM contains $K + M$ hypotheses, in a subsequent step weak hypotheses are removed.

$$p(\mathbf{x}|\mathbf{z}_C) = \frac{p(\mathbf{z}_C|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}_C)} \tag{3.27}$$

In the next search operation the refined system state $p(\mathbf{x}|\mathbf{z}_C)$ is used as prior $p(\mathbf{x})$ and the corresponding search region is determined by that landmark Gaussian combination $\{F_i, G_k\}$, which is expected to have the highest information efficiency score. The original algorithm stops when all iteratively refined search regions are investigated.

A beneficial property of AM is, that not all measurements are required in advance, in opposite to JCBB and JCPL. Extensions of AM are Chow Liu Active Matching (CLAM) and Subset Active Matching (SubAM), which reduce the computational effort by approximating $p(\mathbf{x})$ [11]. They are therefore suitable for a higher amount of landmarks.

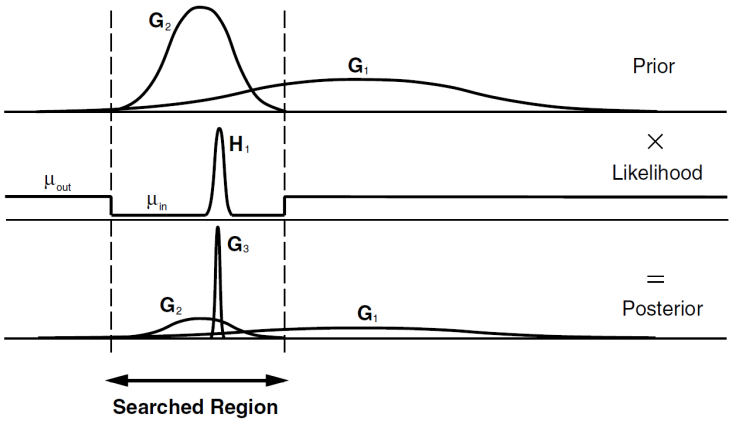


Figure 3.30: 1D illustration of AM update step due to a single match H_1 within search region, from [12].

3.6.3 Restricted Spatial Order Constraint

Restricted Spatial Order Constraints (RSOC) is a graph matching based algorithm designed for aerial image registration [52]. It requires initial one-to-one correspondences between a point set $P = \{p_1, \ldots, p_n\}$ in one image and another point set $Q = \{q_1, \ldots, q_n\}$ in the other image. Based on this initialization, outliers are removed by enforcing a spatial order constraint followed by iteratively solving a cost minimization problem.

For each point p_i in P the K Nearest Neighbors (KNN) $N(p_i) = \left\{p_{N_{p1}}, \ldots, p_{N_{pK}}\right\}$ and the corresponding points $\left\{q_{N_{p1}}, \ldots, q_{N_{pK}}\right\}$ in Q are determined. Fig. 3.31 gives an example for a single point p_i with the angular ordering $O(p_i) = \{N_{p2}, N_{p1}, N_{p6}, N_{p4}, N_{p5}, N_{p3}\}$ in P and the corresponding angular ordering $O'(q_i) = \{N_{p2}, N_{p5}, N_{p6}, N_{p4}, N_{p1}, N_{p3}\}$ in Q. A distance measure between the orderings $O(p_i)$ and $O'(q_i)$ is determined by a cyclic string matching algorithm and denoted as $Dif1(i)$. After determining the ordering distances for all KNN in P the procedure is repeated for all KNN in Q resulting in $Dif2(i)$. In the first step of RSOC, all correspondences with high distance values $Dif1(i)$ and $Dif2(i)$ are removed.

Now the correspondences which decrease an global transformation error E at most are removed iteratively until the error change ΔE is lower equal than a predefined threshold and the total error E reaches a target value. The global transformation error E is calculated according to Equation (3.28) for the remaining correspondences by estimating an affine transformation T from P to Q with parameters θ, Nr denotes the number of the correspondences.

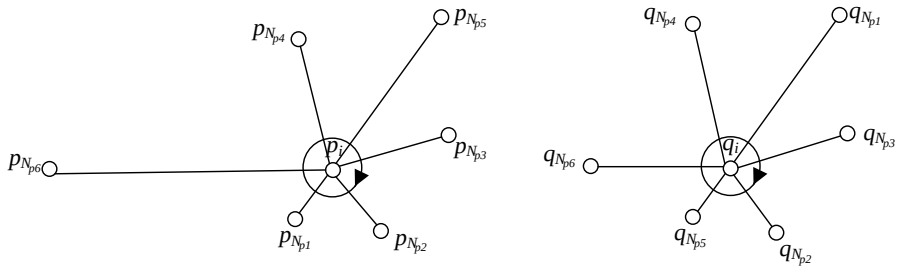


Figure 3.31: Angular spatial order of p_i's KNN in P (left) with corresponding points in Q (right), from [52].

$$E = \sqrt{\frac{\sum_{j=1}^{Nr} \|T(\mathbf{p}_j, \theta) - \mathbf{q}_j\|^2}{Nr}} \tag{3.28}$$

Other feature matching methods for image registration tasks are Graph Transformation Matching (GTM) and Spatial Order Constraints (SOC).

3.6.4 Match Verification

The obtained matches usually contain false matches, e.g. due to repetitive image structure like similar roofs on aerial images. To improve the results of the subsequent processing steps, these matches should be detected and removed as soon as possible. One possibility to identify these matches is to apply the distance ratio test [49]. It calculates the distance ratio between best and second best match, and discards those matches with a ratio above a certain threshold e.g. 0.8 for SIFT. In the following further methods for verifying matches are listed.

3.6.4.1 Epipolar Constraint

Matches between two views can be verified by checking whether they fulfill the epipolar constraint [40]. This can be achieved by estimating the fundamental matrix F from a set of point correspondences $\{(\mathbf{x}_i, \mathbf{x'}_i)\}$ with the Random Sample Consensus (RANSAC) algorithm and simultaneously determining an inlier set. Beside match verification, the obtained fundamental matrix can also be used to perform guided matching. In the following a brief overview of the algorithm is given.

At first a hypothesis for the fundamental matrix F is computed from a few randomly selected correspondences. Then the distances $d_\perp$ of the point correspondences to their epipolar lines are calculated. Now all matches supporting the hypothesis with a $d_\perp$ below

a predefined threshold are determined, they are called the inlier set. That is repeated N times, whereas only the largest inlier set is stored. The value of N depends on the expected amount of outliers and can be adaptively adjusted during execution of the algorithm. Finally the largest set of inliers is used to re-estimate F. Alternatively the fundamental matrix with the largest inlier set can be refined by performing a non-linear optimization step. Further point correspondences can be found by performing guided matching along the epipolar lines calculated from the refined fundamental matrix.

The geometric error $d_\perp(\mathbf{x}_i, \mathbf{x'}_i)$ of two point correspondences $\mathbf{x}_i = [x_i, y_i, 1]^T$ and $\mathbf{x'}_i = [x'_i, y'_i, 1]^T$ from their epipolar lines can be approximately calculated by applying the Sampson error according to Equation (3.29). Here $(F\mathbf{x}_i)_j^2$ denotes the square of the j-th element from the vector $F\mathbf{x}_i$.

$$d_\perp^2(\mathbf{x}_i, \mathbf{x'}_i) = \frac{(\mathbf{x'}_i^T F \mathbf{x}_i)^2}{(F\mathbf{x}_i)_1^2 + (F\mathbf{x}_i)_2^2 + (F^T\mathbf{x'}_i)_1^2 + (F^T\mathbf{x'}_i)_2^2} \tag{3.29}$$

To verify matches across three views the trifocal constraint can be applied, see for example [40]. Also there, a guided matching step can be used to find additional correspondences.

3.6.4.2 Similarity of Neighboring Features

In [19] an outlier detection method is proposed, which assumes that neighboring features have a similar behavior across multiple views. To ease the evaluation a global affine transformation is estimated between two images by using only matches which are verified by the epipolar constraint. Now each epipolar verified correspondence is tested, whether the position difference between the feature location in one image and the transformed feature location of the other image is consistent with the position differences of the neighboring features. A similar test can be performed for the other feature detector output values like scale and orientation.

3.7 3D Structure Computation

Within this work the only reconstructed structure are world points. They are obtained by performing a triangulation using the previously determined feature correspondences and the approximately known projection matrices. Due to the georeferenced projection matrices a metric reconstruction within a `CRS` is possible, leading to georeferenced world

points. The camera parameters and the world points are later refined during a bundle adjustment step. In the following the basic steps for building the 3D structure are given.

3.7.1 Feature Track Generation

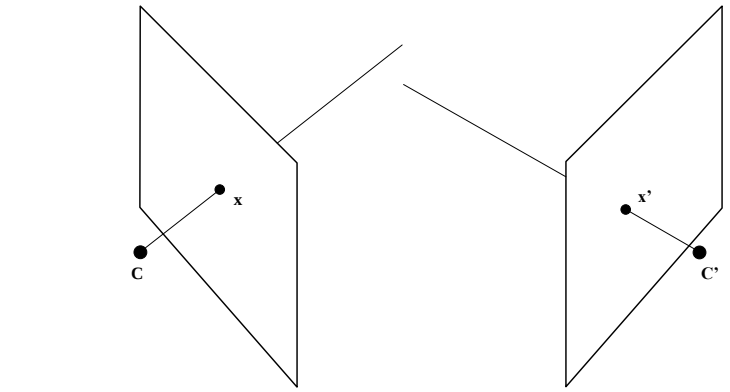
Feature tracks are multi-view correspondences, which are generated by merging the already determined two view matches. They are used during the triangulation stage to obtain world point locations. The usage of feature tracks instead of two view matches avoids redundant world points and improves the triangulation accuracy [56, 57]. To improve the accuracy of the 3D structure further, one can discard those feature tracks in which all rays triangulating a single world point are nearly parallel [20].

For example the feature track generation algorithm in [19] uses previously generated two view feature correspondences to built feature correspondences across three views. These feature triples are tested by their consistency checks, followed by a verification of the trifocal constraint. Finally the feature triples are combined to multi-view matches.

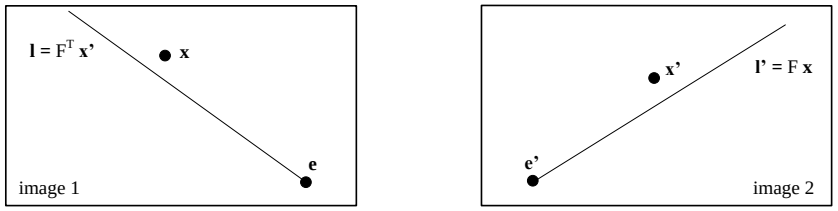
3.7.2 Linear Triangulation

The linear triangulation [40] algorithm can be used to compute world point locations from corresponding feature tracks and known projection matrices. In general it has to be considered that the rays through the individual observations of a single world point do not intersect in practice. That is on the one hand reasoned in measurement errors of the features, causing that the observations are not exactly on the epipolar lines, see Fig. 3.32. On the other hand the projection matrices itself are inexact. To account for that, the presented linear triangulation method computes a least squares solution.

In the following the procedure for triangulating the world point $\mathbf{X}$ from the corresponding feature track, consisting of n observations $\{\mathbf{x}^i\}$, and the associated set of n projection matrices $\{\mathbf{P}^i\}$ is explained. The i-th observation $\mathbf{x}^i$ of the world point $\mathbf{X}$ can be calculated up to scale by applying the projection equation $\mathbf{x}^i = P^i\mathbf{X}$. Then the scale ambiguity is removed by computing the cross product between observation and each side of the projection equation, leading to $0 = \mathbf{x}^i \times (P^i\mathbf{X})$. An expansion leads to Equation (3.30), where two lines of A_i are linear independent, so one line can be discarded. It can be seen that the resulting equation system is linear in the world point location $\mathbf{X}$.



(a)



(b)

Figure 3.32: Triangulation example with two cameras, from [40]. Due to inaccuracies the rays from the camera centers $\mathbf{C}$ and $\mathbf{C}'$ through the observations $\mathbf{x}$ and $\mathbf{x}'$ do not intersect in space, see (a). That is reasoned in the circumstance that the observations are not exactly on the corresponding epipolar lines $\mathbf{l} = F^T\mathbf{x}'$ and $\mathbf{l}' = F\mathbf{x}$, compare (b). The epipoles are denoted with $\mathbf{e}$ and $\mathbf{e}'$.

$$0 = A_i\mathbf{X} = \begin{bmatrix} x^i\mathbf{p}^{i3\,T} - \mathbf{p}^{i1\,T} \\ y^i\mathbf{p}^{i3\,T} - \mathbf{p}^{i2\,T} \\ x^i\mathbf{p}^{i2\,T} - y^i\mathbf{p}^{i1\,T} \end{bmatrix}\mathbf{X} \qquad \mathbf{x}^i = \begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} \qquad P^i = \begin{bmatrix} \mathbf{p}^{i1\,T} \\ \mathbf{p}^{i2\,T} \\ \mathbf{p}^{i3\,T} \end{bmatrix} \tag{3.30}$$

Now the individual A_i from each observation $\mathbf{x}^i$ are stacked to A, leading to the homogeneous equation system $A\mathbf{X} = 0$. Due to the above mentioned inaccuracies this is solved in a least squares manner by minimizing $\|A\mathbf{X}\|$ w.r.t. $\|\mathbf{X}\| = 1$. A solution can be obtained by applying a Singular Value Decomposition (`SVD`). Note that a normalization of A is strongly suggested to improve numerical stability. Alternatively a rewriting as inhomogeneous equation system is possible by representing $\mathbf{X}$ as $[X, Y, Z, 1]^T = [\widetilde{\mathbf{X}}^T, 1]^T$,

resulting in an equation of type $B\widetilde{\mathbf{X}} = b$. A least squares solution is obtained by minimizing $\left\| B\widetilde{\mathbf{X}} - b \right\|$.

3.8 Bundle Adjustment

Following the definition from [5] bundle adjustment is a method in visual reconstruction to refine 3D structure and camera parameters so that a jointly optimal solution is obtained. The following sections define the optimality criteria and show how such a optimization problem is solved.

3.8.1 Problem Formulation

In general bundle adjustment can be used to refine any 3D structure like points, edges and curves in space. Within this work only the location of triangulated points as well as the exterior and interior camera parameters are adjusted. Remember that previous pipeline stages have already extracted feature points from all images, matched them and the resulting correspondences were used to triangulate world points. Due to errors in the camera parameters and imperfections in the feature extraction and matching step, the triangulation is performed in a least squares manner.

Here the reprojection error [40] is evaluated to measure how well image observations, camera parameters and world points fit together. The reprojection error d_{re} is defined as the geometric distance between an observed camera point $\mathbf{x}$ and the location of the corresponding world point projection $\hat{\mathbf{x}} = \hat{P}\hat{\mathbf{X}}$ into the same image, see Equation (3.31). The projection is calculated using the estimated projection matrix $\hat{P}$ and the estimated world point location $\hat{\mathbf{X}}$. These estimates have to be properly initialized and are refined during the bundle adjustment step. For example initial camera poses can be obtained from `GPS` and `INS` measurements, the interior camera parameters from a calibration report and the world point locations from the mentioned triangulation.

$$d_{re} = d\left(\hat{P}\hat{\mathbf{X}}, \mathbf{x}\right) \tag{3.31}$$

When camera positions are measured, then the geometric distances $d_C = d\left(\hat{\mathbf{C}}, \mathbf{C}\right)$ between estimated center of projection $\hat{\mathbf{C}}$ and measured center of projection $\mathbf{C}$ can also be added to the bundle adjustment problem [2]. This has the benefit of achieving a georeferenced metric reconstruction, implicitly resolving the inherent scale ambiguity. Otherwise the scale of the 3D structure may be fixed by choosing a suitable parametrization for the

involved camera translation vectors $\mathbf{t}$, see [40]. Fig. 3.33 illustrates the reprojection errors caused by one world point in two views. Furthermore the deviations from the measured camera positions are shown.

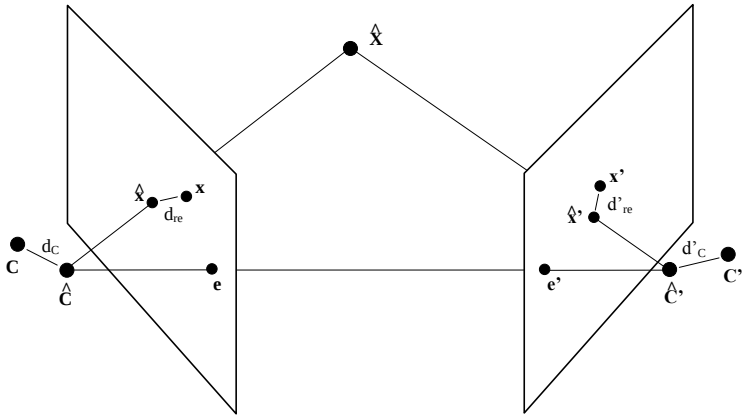


Figure 3.33: Basic bundle adjustment problem, adapted from [40]. The reprojection errors d_{re} and d'_{re} of one world point $\hat{\mathbf{X}}$ within the two views are illustrated. Additionally the distances d_C and d'_C between estimated and observed projection centers are given. The epipoles are denoted with $\mathbf{e}$ and $\mathbf{e}'$.

In summary the goal of the considered bundle adjustment step is to optimize the world point locations as well as the interior and exterior camera parameters, so that the reprojection errors d_{re} and the deviations from the measured camera positions d_C are minimized. The process of adjusting the interior camera parameters is also known as self calibration and is not always performed [5].

3.8.2 Probabilistic View

One view onto the bundle adjustment problem is to regard it as a non-linear least squares optimization task, where the sum of the squared residuals of each observation is minimized, e.g. d_{re}^2 and d_C^2. Note that the non-linearity is introduced by the camera projection equation. Another view onto the bundle adjustment problem is to treat it probabilistically as Maximum Likelihood Estimation (`MLE`) or Maximum-A-Posteriori (`MAP`) problem, which is detailed within this section [5].

From now on the state vector containing all parameters is denoted as $\mathbf{x}$. It contains the world point locations and the camera parameters. Each feature point location and each camera position measurement is represented by a single observation vector $\underline{\mathbf{z}}_i$. Furthermore the parameters $\mathbf{x}$ are used to calculate predictions $\mathbf{z}_i(\mathbf{x})$ for the measurements. The

difference between a single measurement and its prediction is given by the residual prediction error $\mathbf{\Delta z}_i(\mathbf{x}) = \underline{\mathbf{z}}_i - \mathbf{z}_i(\mathbf{x})$. Now assume n independent noisy measurements, each with a distribution $p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})$. Then the probability $p_{model}(\underline{\mathbf{z}}|\mathbf{x})$ of occurring a specific set of measurements $\underline{\mathbf{z}}$ given the parameters $\mathbf{x}$ follows according to Equation (3.32).

$$p_{model}(\underline{\mathbf{z}}|\mathbf{x}) = \prod_{i=1}^{n} p_{model}(\underline{\mathbf{z}}_i|\mathbf{x}) \tag{3.32}$$

The MLE is found by choosing that parameter vector $\mathbf{x}$ with the highest probability, whereas no prior knowledge about $\mathbf{x}$ is considered, see Equation (3.33). When also a prior distribution $p_{prior}(\mathbf{x})$ over the parameters $\mathbf{x}$ is known, then the MAP estimate can be determined by Equation (3.34).

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \; p_{model}(\underline{\mathbf{z}}|\mathbf{x}) \tag{3.33}$$

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \; p_{model}(\underline{\mathbf{z}}|\mathbf{x})\; p_{prior}(\mathbf{x}) \tag{3.34}$$

In the first step lets determine the MLE and assume that the measurement process can be modeled by additive N dimensional Gaussian noise with covariance Σ_i and zero mean deviation from the prediction $\mathbf{z}_i(\mathbf{x})$ according to Equation (3.35).

$$p_{model}(\underline{\mathbf{z}}_i|\mathbf{x}) = (2\pi)^{-N/2}\det\left(\Sigma_i^{-1}\right)^{1/2} \exp\left(-\frac{\mathbf{\Delta z}_i(\mathbf{x})^T \Sigma_i^{-1} \mathbf{\Delta z}_i(\mathbf{x})}{2}\right) \tag{3.35}$$

To simplify subsequent calculations the log-likelihood $\mathcal{L}(\mathbf{x})$ is determined from $p_{model}(\underline{\mathbf{z}}|\mathbf{x})$ by applying the logarithm.

$$\begin{aligned} \mathcal{L}(\mathbf{x}) &= \ln\left(p_{model}(\underline{\mathbf{z}}|\mathbf{x})\right) = \sum_{i=1}^{n} \ln\left(p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})\right) \\ &= \sum_{i=1}^{n} \ln\left((2\pi)^{-N/2}\det\left(\Sigma_i^{-1}\right)^{1/2}\right) - \frac{1}{2}\sum_{i=1}^{n} \mathbf{\Delta z}_i(\mathbf{x})^T \Sigma_i^{-1} \mathbf{\Delta z}_i(\mathbf{x}) \end{aligned} \tag{3.36}$$

In the above log-likelihood formulation only the second term depends on $\mathbf{x}$. Therefore the first part can be omitted during the minimization of $-\mathcal{L}(\mathbf{x})$, which is equivalent to maximizing $p_{model}(\underline{\mathbf{z}}|\mathbf{x})$. The resulting cost function $f(\mathbf{x})$ is given in Equation (3.37). It can be observed that in the case of Gaussian noise the MLE solution is found by solving a weighted non-linear least squares optimization problem with weight $W_i = \Sigma_i^{-1}$.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \; -\mathcal{L}(\mathbf{x}) = \arg\min_{\mathbf{x}} \; f(\mathbf{x}) \qquad f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} \boldsymbol{\Delta}\mathbf{z}_i(\mathbf{x})^T \Sigma_i^{-1} \boldsymbol{\Delta}\mathbf{z}_i(\mathbf{x}) \tag{3.37}$$

Similarly the MAP solution can be obtained by extending $f(\mathbf{x})$ with the corresponding prior cost term. This term is additive and in the case of a Gaussian prior the resulting optimization problem is still a weighted non-linear least squares problem.

3.8.3 Levenberg-Marquardt

The Levenberg-Marquardt (LM) [5, 40] algorithm is a common method to solve bundle adjustment problems and is based on a modified Newton iteration. It provides a faster convergence than the Newton iteration and can solve over parameterized problems. A optimization problem is called over parameterized, when the state vector $\mathbf{x}$ has more elements than the problem has Degrees of Freedom (DoF). In general it is not guaranteed that the global cost minimum is found.

At first the Newton iteration is described, which calculates small state updates $\boldsymbol{\delta}\mathbf{x}$, trying to minimize the cost function $f(\mathbf{x})$ iteratively until convergence. For calculating $\boldsymbol{\delta}\mathbf{x}$ the cost function is approximated by the following Taylor expansion

$$f(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{g}^T \boldsymbol{\delta}\mathbf{x} + \frac{1}{2}\boldsymbol{\delta}\mathbf{x}^T H \boldsymbol{\delta}\mathbf{x} \tag{3.38}$$

at the current state $\mathbf{x}$ with gradient vector $\mathbf{g}$ and Hessian matrix H

$$\mathbf{g} = \nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_i}\right]_i \qquad H = \nabla^2 f(\mathbf{x}) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{i,j}$$

The Newton step is now determined by setting the first derivative of Equation (3.38) to zero and solving for $\boldsymbol{\delta}\mathbf{x}$.

$$\frac{\partial f(\mathbf{x} + \boldsymbol{\delta}\mathbf{x})}{\partial \boldsymbol{\delta}\mathbf{x}} \approx H\boldsymbol{\delta}\mathbf{x} + \mathbf{g} \stackrel{!}{=} 0 \qquad H\,\boldsymbol{\delta}\mathbf{x} = -\mathbf{g} \tag{3.39}$$

In the case of the LM algorithm a Damped Newton step is calculated according Equation (3.40). Here the regularization term λI with the scalar value λ and the identity matrix I has been added compared to Equation (3.39).

$$(H + \lambda I)\,\boldsymbol{\delta}\mathbf{x} = -\mathbf{g} \tag{3.40}$$

For small values of λ the regularization has no influence and the Damped Newton step is approximately equal to the Newton step. In opposite for large λ values the influence of the Hessian matrix H can be neglected and one gets the gradient descent step with $\lambda\,\boldsymbol{\delta}\mathbf{x} = -\mathbf{g}$.

Similar to Newton iteration, also the `LM` algorithm applies iteratively state updates until a local minimum is found, but this is now done by adapting λ for each step. At the beginning λ is typically initialized with 10^{-3} times the average diagonal element of H. Now a state update $\boldsymbol{\delta}\mathbf{x}$ is calculated and the cost function is evaluated at $\mathbf{x} + \boldsymbol{\delta}\mathbf{x}$. When the cost has decreased, then the state vector is updated $\mathbf{x} \rightarrow \mathbf{x} + \boldsymbol{\delta}\mathbf{x}$, λ is divided by a factor e.g. 10 and a new iteration starts. Otherwise λ is multiplied by the same factor, a new step $\boldsymbol{\delta}\mathbf{x}$ is computed and $f(\mathbf{x} + \boldsymbol{\delta}\mathbf{x})$ is compared with $f(\mathbf{x})$. That repeats until a $\boldsymbol{\delta}\mathbf{x}$ is found which decreases the cost.

As mentioned above the problem might be accidentally over parameterized, which causes that H becomes singular. Usually the update step $\boldsymbol{\delta}\mathbf{x}$ can still be calculated due to the regularization term λI. There exist also other `LM` implementations using a different augmentation of the Newton step.

Now let's consider the weighted non-linear least squares cost function from Equation (3.37). By using the weights $W_i = \Sigma_i^{-1}$ and assembling them to the block diagonal matrix W, one can rewrite $f(\mathbf{x})$ according Equation (3.41).

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\Delta}\mathbf{z}(\mathbf{x})^T W \boldsymbol{\Delta}\mathbf{z}(\mathbf{x}) \tag{3.41}$$

Then the cost gradient vector $\mathbf{g}$ and cost Hessian matrix H can be formulated by the usage of the Jacobian matrix J.

$$\mathbf{g} = J^T W \boldsymbol{\Delta}\mathbf{z} \qquad J = \left[\frac{\partial \Delta z_i}{\partial x_j}\right]_{i,j} \tag{3.42}$$

$$H = J^T W J + \sum_i (\boldsymbol{\Delta}\mathbf{z}^T W)_i \left[\frac{\partial^2 \Delta z_i}{\partial x_k \partial x_l}\right]_{k,l} \tag{3.43}$$

A common simplification of the Hessian is the Gauss-Newton approximation $H \approx J^T W J$, which neglects the second term in Equation (3.43). The Jacobian J and the Hessian H are usually sparse, which is illustrated in Fig. 3.34 for a small bundle adjustment problem with five world points and four images made by two different cameras. To see how the sparseness arises consider for example world point A projected into image 1 leading to the camera point $A1$. Here the residual for camera point $A1$ depends solely on the location

of world point A, the camera pose at the exposure moment for image 1 and the camera calibration matrix K_1. As one line of the Jacobian contains the partial derivatives of one observation residual Δz_i w.r.t. the state vector elements x_j, only the former mentioned parameters cause non zero entries in the Jacobian. The sparseness of H results from applying the Gauss-Newton approximation and exploiting the block diagonal structure of W. Due to the sparse structure of H the update step $\boldsymbol{\delta}\mathbf{x}$ in Equation (3.40) is not calculated by inverting $(H + \lambda I)$, because this would result in a dense matrix. Instead methods exploiting the sparseness are used, for example symmetric matrix factorization. More details can be found in [5, 40].

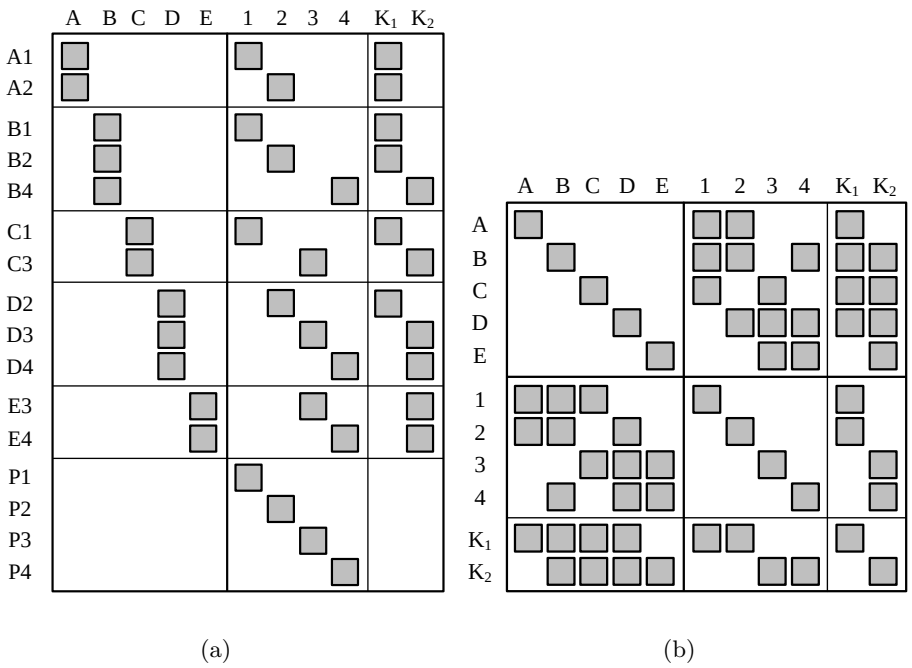


Figure 3.34: Illustration of sparse Jacobian and Hessian matrices for a small bundle adjustment problem, adapted from [5]. The given problem consists of five world points $A-E$ and four camera images $1-4$ obtained from two different cameras with calibration matrices K_1, K_2. The observation error of a camera point is denoted by the combination of world point letter and image number, e.g. $C3$ means world point C in image 3. The difference between observed and predicted center of projection are indicated by $P1 - P4$.

3.8.4 Robust Loss

To see the influence when the observations $\underline{\mathbf{z}}_i$ are drawn from another distribution $p_0(\underline{\mathbf{z}}_i)$ than the designed one, assume that the observations are modeled by n independent identical distributions $p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})$. Then the expected log-likelihood for a given parameter $\mathbf{x}$ is as given in the following equation.

$$\mathbb{E}\left\{\mathcal{L}(\mathbf{x})\right\} = \mathbb{E}\left\{\sum_{i=1}^{n} \ln\left(p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})\right)\right\} = n \int p_0(\underline{\mathbf{z}}_i) \ln\left(p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})\right) d\underline{\mathbf{z}}_i \tag{3.44}$$

For interpreting the result consider the relative entropy, also known as Kullback-Leibler divergence $d(p_0, p_{model})$, between both distributions as given in Equation (3.45).

$$\begin{aligned} d(p_0, p_{model}) &= \int p_0(\underline{\mathbf{z}}_i) \ln \frac{p_0(\underline{\mathbf{z}}_i)}{p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})}\, d\underline{\mathbf{z}}_i \\ &= \int p_0(\underline{\mathbf{z}}_i) \ln p_0(\underline{\mathbf{z}}_i)\, d\underline{\mathbf{z}}_i - \int p_0(\underline{\mathbf{z}}_i) \ln p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})\, d\underline{\mathbf{z}}_i \end{aligned} \tag{3.45}$$

It can be observed that the first term is independent from the chosen model distribution, while the second term is $-1/n$ times the expected log-likelihood. As mentioned above, `MLE` selects the parameter $\mathbf{x}$, corresponding to a distribution from the family $p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})$, with the maximum log-likelihood. This is equivalently in choosing the distribution with the smallest Kullback-Leibler divergence $d(p_0, p_{model})$ w.r.t. p_0. Furthermore it can be seen that the result is in general sensitive to not modeled outliers, because regions with high p_0 and low p_{model} have a huge impact on $d(p_0, p_{model})$. The opposite case influences $d(p_0, p_{model})$ only slightly. Therefore a distribution $p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})$ which accounts for outliers should be chosen [5].

When a radial distribution is selected for $p_{model}(\underline{\mathbf{z}}_i|\mathbf{x})$, one obtains a cost function according to Equation (3.46), which uses loss functions $\rho_i\left(s_i\right)$ for each observation error $\mathbf{\Delta z}_i(\mathbf{x})$. The loss function depends on $s_i = \mathbf{\Delta z}_i(\mathbf{x})^T W_i \mathbf{\Delta z}_i(\mathbf{x})$ and accounts for the shape of the underlying distribution [5].

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{n} \rho_i\left(\mathbf{\Delta z}_i(\mathbf{x})^T W_i \mathbf{\Delta z}_i(\mathbf{x})\right) \tag{3.46}$$

In the following some commonly used loss functions are listed and Fig. 3.35 depicts their shape and the basic shape of the underlying distribution [40].

- **Squared Error**
  It is assumed that measurement errors are Gaussian distributed and this loss function should be only used, when outliers have been removed in advance.

$$\rho(s) = s \tag{3.47}$$

- **Corrupted Gaussian**
  Inliers and outliers are modeled by a mixture of two Gaussians, where each Gaussian represents one group. Here α gives the probability of an inlier and ω is the multiplier for the standard deviation of an outlier.

$$\rho(s) = -\ln\left(\alpha \exp(-s) + (1-\alpha)\exp(-s/\omega^2)/\omega\right) \tag{3.48}$$

- **Blake-Zisserman**
  Inliers are assumed to be Gaussian distributed, while outliers are expected to be uniform distributed. The inlier threshold is $s = -\ln(\epsilon)$.

$$\rho(s) = -\ln\left(\exp(-s) + \epsilon\right) \tag{3.49}$$

- **Cauchy**
  The Cauchy loss approximates the squared loss for small s/b^2 and down weights the influence of higher s. It is based on the Cauchy distribution.

$$\rho(s) = b^2 \ln\left(1 + s/b^2\right) \tag{3.50}$$

- **Huber**
  The Huber loss has an asymptotically linear characteristics, while the squared loss is approached for $\sqrt{s} < b$. Parameter b gives the outlier threshold and only the first derivative of $\rho(s)$ is continuous.

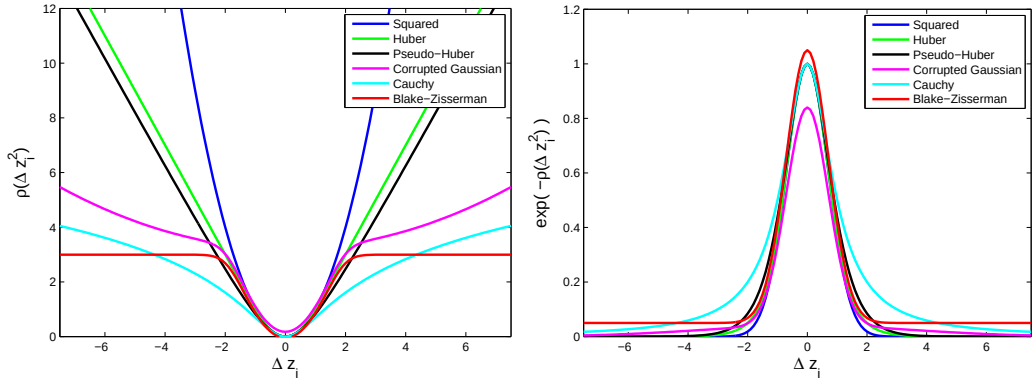
$$\rho(s) = \begin{cases} s & \sqrt{s} < b \\ 2b\sqrt{s} - b^2 & \text{otherwise} \end{cases} \tag{3.51}$$

- **Pseudo-Huber**
  This loss is similar to the Huber loss, but here all derivatives are continuous.

$$\rho(s) = 2b^2 \left(\sqrt{1 + s/b^2} - 1 \right) \tag{3.52}$$

The losses Squared Error, Huber and Pseudo-Huber are convex, while the others are not. When a convex $\rho(s)$ is used in Equation (3.46), also the resulting cost is convex in $\Delta\mathbf{z}$. But be beware of that it is not convex in $\mathbf{x}$, because the predictions $\mathbf{z}_i(\mathbf{x})$ are non-linear in the state vector $\mathbf{x}$. Nevertheless convex loss functions should be preferred to avoid local minima.

Asymptotically linear loss functions achieve their robustness by approximating the L1 norm, instead of modeling the inlier and outlier distributions. The L1 norm loss function has the property that the resulting cost function has its minimum at the median value of the data points, e.g. for a set of scalar data points $\{a_i\}$ the simple cost function $\sum_i |x - a_i|$ has its minimum at $x = \text{median}(\{a_i\})$ [40].



(a) (b)

Figure 3.35: Comparison of different loss functions, adapted from [40]. In (a) the loss functions $\rho(s)$, which are listed in the above text, are given for scalar observations $s = \Delta z_i^2$, $b = 1$, $\alpha = 0.8$, $\epsilon = 0.05$ and $\omega = 5$. The basic shape of the underlying distributions are shown in (b) by plotting $\exp(-\rho(s))$. Note that the area under the curves does not sum up to 1, therefore they are actually not Probability Density Functions (PDFs).

There exist different strategies to take account for the above loss functions within the LM algorithm. One way is to use the robustified Gauss-Newton approximation [5], which uses the first and second derivatives $\rho_i'(s_i)$ and $\rho_i''(s_i)$ of $\rho_i(s_i)$, see Equation (3.53) and Equation (3.54). To shorten the equations, the dependence of $\rho_i'(s_i)$ and $\rho_i''(s_i)$ on $s_i = \mathbf{\Delta z}_i(\mathbf{x})^T W_i \mathbf{\Delta z}_i(\mathbf{x})$ is omitted.

$$\mathbf{g} = \sum_{i=1}^{n} \mathbf{g}_i \qquad \mathbf{g}_i = \rho_i' J_i^T W_i \boldsymbol{\Delta}\mathbf{z}_i \tag{3.53}$$

$$H = \sum_{i=1}^{n} H_i \qquad H_i \approx J_i^T \left(\rho_i' W_i + 2\rho_i''(W_i \boldsymbol{\Delta}\mathbf{z}_i)(W_i \boldsymbol{\Delta}\mathbf{z}_i)^T\right) J_i \tag{3.54}$$

3.8.5 Parameterization

In this context the representation of 3D structure, camera poses and camera calibrations by the state vector $\mathbf{x}$ is known as parameterization [5, 40]. A good parameterization is a one-to-one mapping, meaning one physical state should be represented by exactly one value of the state vector $\mathbf{x}$. Furthermore $\mathbf{z}_i(\mathbf{x})$ should be locally continuous, differentiable and as linear as possible.

As mentioned above, the only 3D structure utilized in this work are points in three dimensional Euclidean space $\mathbb{R}^3$. A homogeneous parameterization represents the points with 4-vectors and allows the specification of locations at infinity with finite coordinate values e.g. $\mathbf{X} = [X, Y, Z, W]^T$ with $W = 0$. As a consequence very distant positions can be reached within few `LM` update steps. This representation has four `DoF` although only three `DoF` are necessary to state any location in three dimensional space, which results in a scale ambiguity. To resolve this ambiguity, a spherical normalization e.g. $X^2+Y^2+Z^2+W^2 = 1$ is required after each state update. When all world points have relatively small coordinate values a representation by 3-vectors is also sufficient, making a normalization unnecessary.

To ease computations, rotations in $\mathbb{R}^3$ are often represented by 3×3 matrices having nine `DoF`, although only three `DoF` would be necessary. Alternatives avoiding over parameterization are the angle-axis representation and unit quaternions. The angle-axis representation uses a 3-vector $\mathbf{r} = [r_1, r_2, r_3]^T$ to specify a rotation around an axis $\mathbf{r}$ by an angle $\|\mathbf{r}\|$. A given angle-axis rotation $\mathbf{r}$ can be converted to a rotation matrix R by applying Equation (3.55), where I denotes the identity matrix.

$$R = I + \frac{\sin\|\mathbf{r}\|}{\|\mathbf{r}\|}[\mathbf{r}]_\times + \frac{1-\cos\|\mathbf{r}\|}{\|\mathbf{r}\|^2}[\mathbf{r}]_\times^2 \qquad [\mathbf{r}]_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \tag{3.55}$$

For $\|\mathbf{r}\| = 2\pi$ the resulting rotation matrix is the identity matrix, independently of the orientation of $\mathbf{r}$, causing a many-to-one mapping. Therefore the following normalization

is suggested after each parameter update, when $\|\mathbf{r}\| > \pi$.

$$\mathbf{r} = (\|\mathbf{r}\| - 2\pi)\,\frac{\mathbf{r}}{\|\mathbf{r}\|}$$

A unit quaternion $\mathbf{q}$ is 4-vector with $\|\mathbf{q}\| = 1$ and can be calculated from an angle-axis representation $\mathbf{r}$ according to Equation (3.56). It is a two-to-one mapping, because $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation. It is required to perform a normalization to unit length after each parameter update.

$$\mathbf{q} = \left[\frac{\sin(\|\mathbf{r}\|/2)}{\|\mathbf{r}\|}\mathbf{r}^T \quad \cos(\|\mathbf{r}\|/2)\right]^T \tag{3.56}$$

Up to now state updates in the `LM` algorithm are always performed by simple vector additions $\mathbf{x} \rightarrow \mathbf{x} + \boldsymbol{\delta}\mathbf{x}$. Alternatively one can reformulate the state update procedure by a generalized vector addition $\boxplus(\mathbf{x}, \boldsymbol{\delta}\mathbf{x})$, where $\boxplus(\mathbf{x}, \mathbf{0}) = \mathbf{x}$ is valid for all $\mathbf{x}$, leading to an update step $\mathbf{x} \rightarrow \boxplus(\mathbf{x}, \boldsymbol{\delta}\mathbf{x})$ [58]. With this modification, it is allowed that $\boldsymbol{\delta}\mathbf{x}$ is of lower dimension than $\mathbf{x}$. For example in a subset parameterization some of the elements in $\mathbf{x}$ are forced at constant values, regardless of the current state, e.g.

$$\boxplus(\mathbf{x}, \boldsymbol{\delta}\mathbf{x}) = \mathbf{x} + \begin{bmatrix}\boldsymbol{\delta}\mathbf{x} \\ \mathbf{0}\end{bmatrix}$$

In contrast local parameterizations consider the current state. For example they can perform a linearization at the current state or state updates $\boldsymbol{\delta}\mathbf{x}$ can act in a lower dimensional tangent space placed at the current state $\mathbf{x}$. Furthermore any normalization can be directly included into $\boxplus(\mathbf{x}, \boldsymbol{\delta}\mathbf{x})$.

Chapter 4

Methodology

Contents

This chapter delineates the chosen design of the AT pipeline. Special attention is paid on a fast processing of the distinct stages by exploiting as many information as available. More precisely the considered prior knowledge is obtained from measured camera poses and from scene approximations like a DEM or a ground plane. A flow chart is given in Fig. 4.1. The purpose of the distinct pipeline stages is given in the following:

- **Configuration**
  A graphical configuration utility assists the user in preparing the input data for the AT pipeline. Additionally a visualization of the camera arrangement with corresponding back-projected image borders onto a specified ground plane is included, see Fig. 4.3.

- **Feature Extraction**
  Many different feature detectors and descriptors are available, e.g. [48–51]. Within this work only SIFT is applied for feature extraction, which has already been successfully applied in AT pipelines, for example in [2, 23, 52]. More precisely the SiftGPU

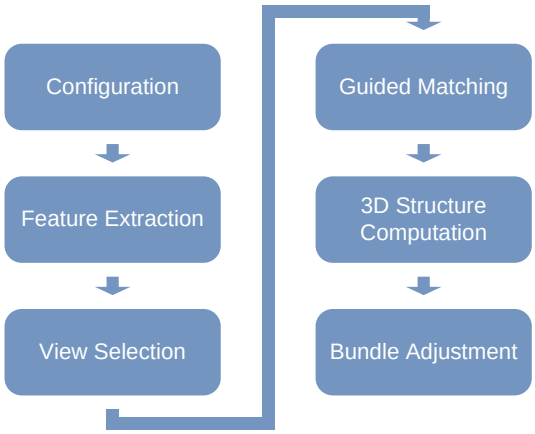


Figure 4.1: Flow chart of the AT pipeline.

[59] implementation is chosen, which performs feature extraction on a Graphics Processing Unit (GPU).

- **View Selection**
  View selection is used to determine images showing the same scene content, so that the subsequent guided matching stage has not to exhaustively match each image with each other. A common method is to perform the exhaustive matching only on a small feature subset and then to decide which images show the same part of a scene [19]. Here a different approach is applied, where the measured camera poses and a given DEM are evaluated to identify these views [21].

- **Guided Matching**
  Instead of comparing each feature in one image with each feature in another image a more efficient method is utilized. The measured camera poses and a specified DEM are used to predict the location of a feature point in one image given the corresponding location in another image. Then the feature correspondence is searched in a small window around the prediction [8].

- **3D Structure Computation**
  The only 3D structure computed within this work are sparse point clouds. To triangulate these world points from image observations, the above introduced linear triangulation algorithm is utilized. The same algorithm has been previously used for the same task, e.g. in [2].

- **Bundle Adjustment**
  Depending on the precision of the camera pose measurement the obtained 3D structure may be very inaccurate. Therefore a bundle adjustment stage is applied to refine the locations of the world points as well as the camera poses [5, 40].

The subsequent section gives an overview of the configuration stage. Then the `DEM` processing is described. Afterwards the view selection, guided matching and bundle adjustment stages are detailed.

4.1 Configuration

Within this work Check Points (`CPs`) are used to evaluate the accuracy of the reconstruction [4, 42, 60]. That are world points with known coordinates, each observed by at least two cameras. The user configures the world point coordinates and the corresponding observations in the images. Then the camera points are handled as normal feature correspondences from which the pipeline triangulates a world point location, which is refined during the bundle adjustment step. Finally the computed world point coordinates are compared with the configured ones to assess the accuracy. Furthermore it is possible to specify `GCPs`, which are known world points each associated with one or more image measurements. In opposite to `CPs`, `GCPs` are not triangulated. But they also add cost terms to the optimization problem and introduce ground truth information into the bundle adjustment stage. Therefore `GCPs` influence how camera parameters are changed.

To configure the `CP` and `GCP` observations, the user has to precisely locate them within the aerial images. These images are possibly large, for example the Microsoft UltraCamX generates images with a size of 14430 pixel $\times$ 9420 pixel. Here it would be very time consuming and error-prone to search the world point observations in each image without any guidance. Therefore a configuration utility has been designed, which suggests the projection of the previously configured world point as observation. However due to inaccuracies in the camera poses, the initial guess has to be refined by the user. Fig. 4.2 shows the user interface for adjusting camera points.

Depending on the aerial survey size, the processing of the remaining pipeline stages may last many hours. Therefore as many checks as possible should be performed in the configuration stage to avoid repeated executions of the whole pipeline. So on the one hand the utility has the ability to perform tests on the configuration and to generate an error report. Such a test is for example the check, whether each `CP` is observed by at least

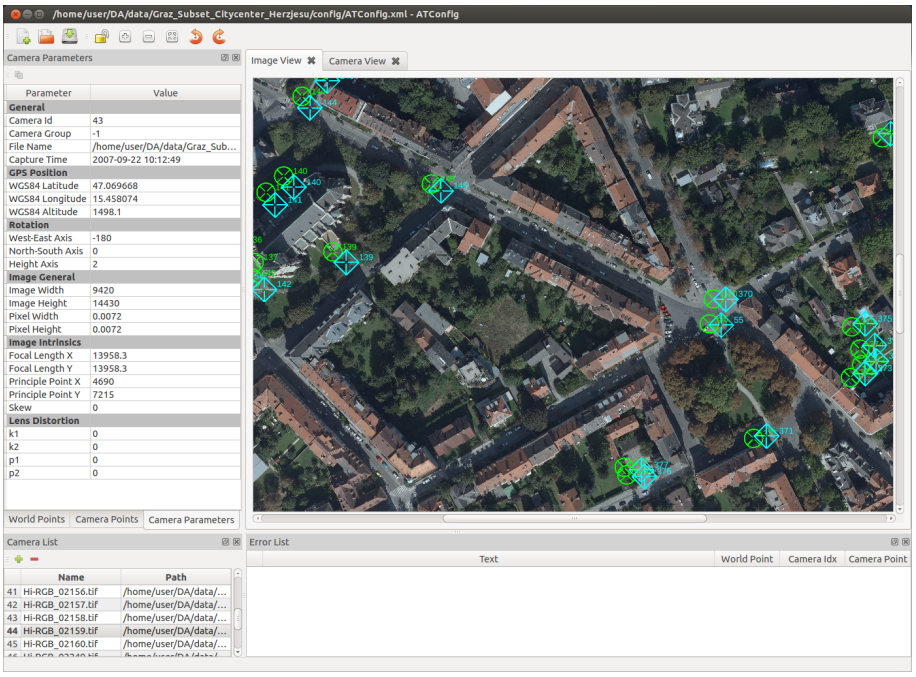


Figure 4.2: Image view of the configuration utility. Green markers indicate projections of the configured world points, while cyan markers depict configured camera points.

two cameras. This particular test is necessary, because otherwise it would not be possible to triangulate a world point location. On the other hand a visualization of the camera poses, image border back-projections onto a ground plane as well as CP and GCP locations is included, see Fig. 4.3. Therefore the user can immediately verify, whether the correct aerial survey will be processed and if the CPs and GCPs are well distributed within the surveyed region.

The input data used for evaluating the implemented AT pipeline is given in different coordinate systems. While the horizontal and vertical camera positions are supplied in WGS84 coordinates, the Shuttle Radar Topography Mission (SRTM) data from [9] for generating the DEM uses horizontal WGS84 coordinates and heights above the EGM96 Geoid. In contrast the applied SfM procedure requires a Cartesian coordinate system, so the horizontal WGS84 coordinates have to be converted anyway. Possible choices for the world coordinate system of the pipeline are amongst others the UTM and the ENU system. The aerial images within this work cover only a small region and therefore the distortions introduced by applying the UTM map projection can be neglected. In addition UTM is widely supported in different GIS products, e.g. [33]. So it has been decided that the AT pipeline

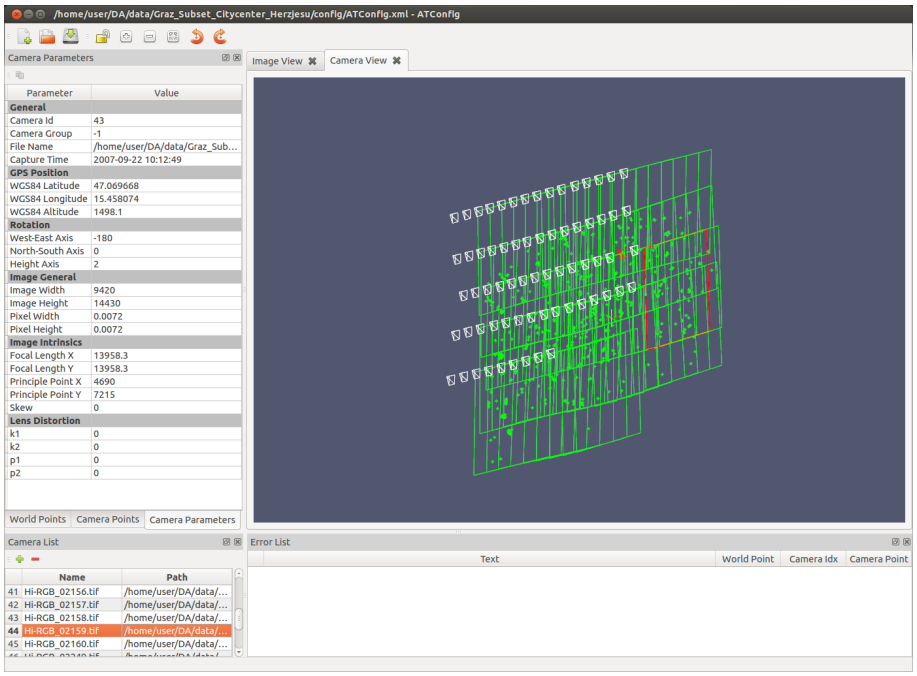


Figure 4.3: Camera view of the configuration utility. Crosses indicate world points, pyramids show camera poses and polygons depict image border back-projections onto a configured ground plane.

operates directly on `UTM` coordinates. Furthermore `WGS84` heights are used, because they are geometrically defined and not gravity based.

4.2 Digital Elevation Model

The pipeline uses a scene approximation in the following stages:

- **View Selection**
  The measured camera poses and the scene approximation are utilized to determine the overlap between images of different cameras within the view selection stage.

- **Guided Matching**
  In the guided feature matching step the world point location of a single given camera point is estimated by back-projecting the camera point onto the scene approximation.

- **Bundle Adjustment**
  During the bundle adjustment stage world points likely to be outliers are removed by evaluating their distance to the scene approximation.

Within this work two different scene approximations are evaluated, the first is a ground plane and the other is a DEM. While the ground plane assumption allows fast and easy computations, the DEM leads to more precise results in terrains with significant height variations. Note that nearly for the entire earth elevation data is available and freely accessible [9]. The next sections describe the chosen procedures for applying a DEM as scene approximation in detail.

4.2.1 Representation

The elevation data in this work is obtained from [9], they publish a refined version of the SRTM data grid with a 3-arc seconds resolution, corresponding to an approximately 90 m grid spacing at the equator. An example tile is shown in Fig. 4.4. The horizontal datum is given in WGS84 format, while the height information is stored in meters above the EGM96 Geoid.

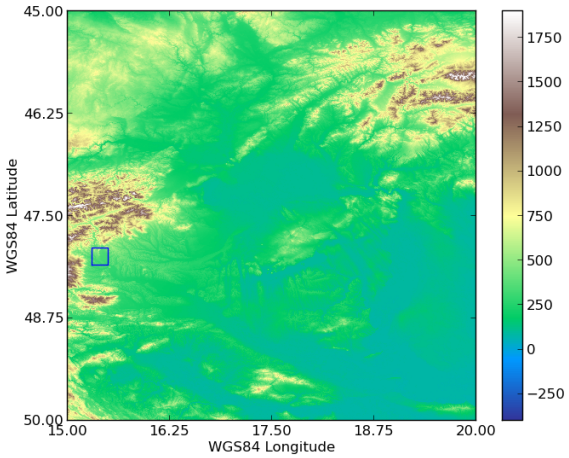


Figure 4.4: SRTM elevation data tile given in WGS84 horizontal datum and EGM96 vertical datum [9].

In the following the elevation surface is represented as triangle mesh, like in [34]. Therefore points are extracted from those grid cells lying within a previously defined ROI. Then these points are converted into horizontal UTM and vertical WGS84 coordinates and triangulated. Finally the mesh simplification algorithm given in [38] and also introduced in Section 3.2.5 is applied. Fig. 4.5 shows the resulting surface from the blue marked region in Fig. 4.4.

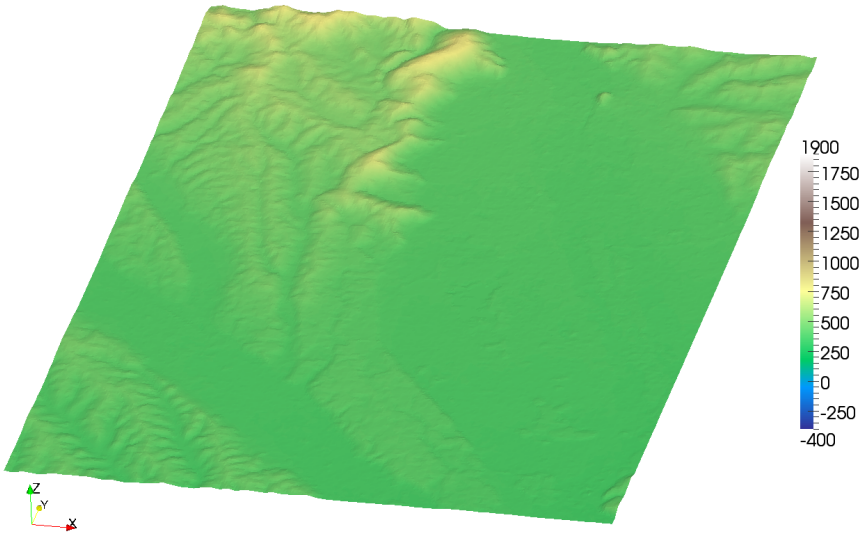


Figure 4.5: Triangulated DEM from the region marked with the blue box in Fig. 4.4. The surface is given in UTM $33N$ horizontal and WGS84 vertical coordinates.

To enable an efficient access to the terrain triangles, a 2D segment tree is built once for the whole DEM, containing the XY-bounding boxes of all triangles. Furthermore a bottom and top plane parallel to the XY-plane are calculated. This planes touch the triangulated surface at the points with the lowest and highest Z-coordinate values respectively.

4.2.2 Visible DEM Region

The above mentioned distance computations between a given world point and the triangulated surface are performed by utilizing the AABB tree implementation of [37]. While this distance computations operate on the whole surface, the camera point back-projection and image overlap calculations are performed only on those triangles visible in a given camera. In the following it is described how these triangles are determined.

At first the view frustum of a given camera is approximated by a minimal sized axis aligned bounding box. When the camera center is above the top touching plane of the DEM, then the bounding box contains the corner points of the image border back-projections onto the top and bottom touching planes, see Fig. 4.6 for an illustration. Otherwise the bounding box contains only the camera center and the corners of the image border back-projected onto the bottom touching plane. Fig. 4.7 shows the determined bounding boxes for two different camera locations. Additionally the projections of the image borders are displayed.

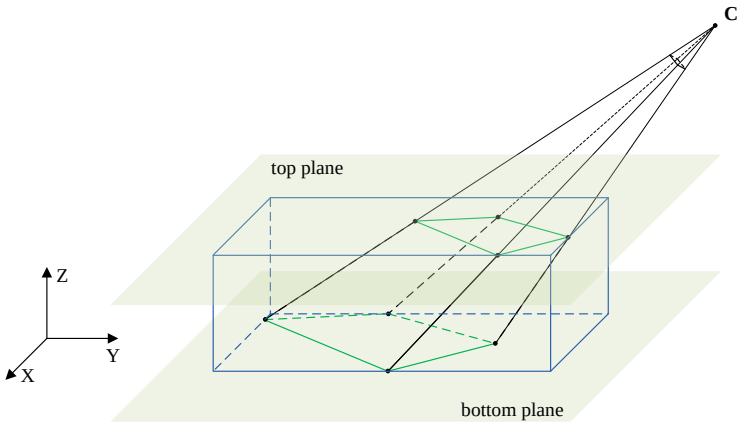
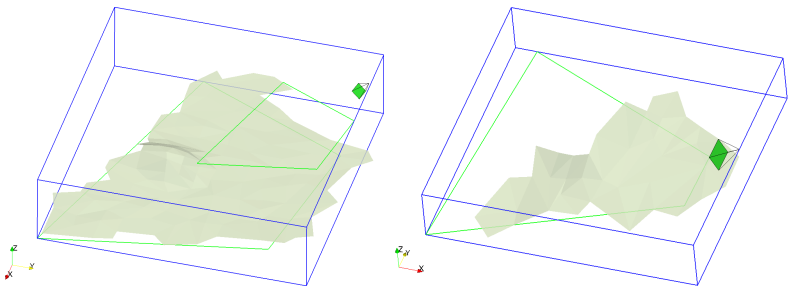


Figure 4.6: Worst case approximation of the space occupied by the camera view frustum. The green polygons indicate the image borders back-projected on the bottom and top plane respectively, while the blue cuboid depicts the axis aligned bounding box built from both polygons.



(a) Camera 1 (b) Camera 2

Figure 4.7: Examples of axis aligned bounding boxes for approximating the view frustum. In (a) the camera is above the top plane, while in (b) the camera is below it.

To determine the visible triangles, the 2D segment tree associated with the whole `DEM` is queried for triangles overlapping with the XY-range of the view frustums bounding box. The returned triangles are clipped with the cameras near clipping plane to remove vertices behind the camera. In the next step the resulting polygons are projected into the camera image. Finally polygons with all points outside the image are discarded, if these points are all located on one side of the image border, for example when all polygon points are on the left of the image, see Fig. 4.8.

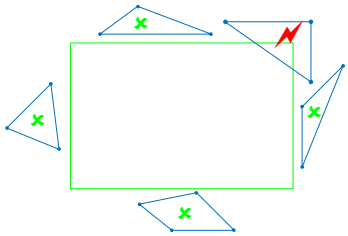
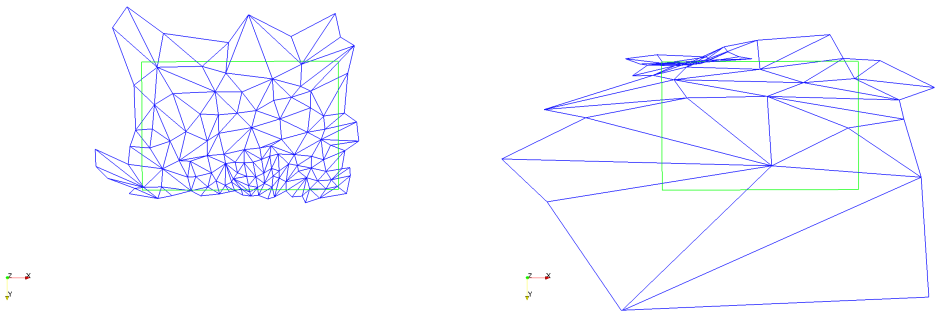


Figure 4.8: Illustration of triangle removal based on the Cohen-Sutherland line clipping algorithm. Removable polygons are marked with a green cross.

Fig. 4.9 shows the images of the remaining polygons for the same cameras as defined above. Such an image is called a view onto the DEM in the following sections. Additionally Fig. 4.7 displays the back-projected polygons in world frame coordinates. Finally a segment tree containing the bounding boxes of the projected polygons is built, to speed up further view related processing.



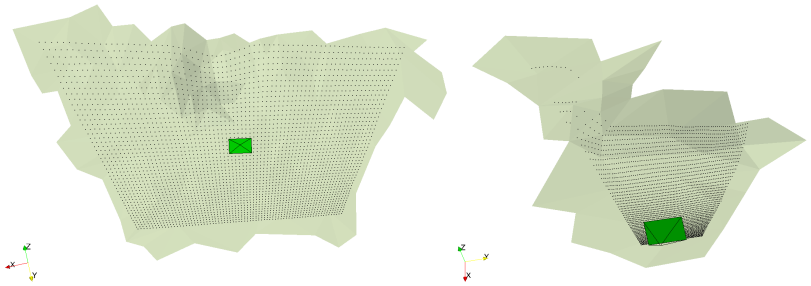
(a) Camera 1 (b) Camera 2

Figure 4.9: Images of potential visible polygons for the same cameras as in Fig. 4.7.

4.2.3 Camera Point Back-Projection

The feature matching stage in the AT pipeline requires that camera points from one view can be back-projected onto the DEM. The obtained world points are then projected into another view to determine initial guesses for feature locations. To back-project a camera point, the segment tree of the related view is queried for all polygons potentially containing this point. For each returned polygon it is checked, whether the point is within the polygon. Then the point is back-projected onto all found polygons. The obtained world

frame location nearest to the camera center is the resulting world point location. Fig. 4.10 shows the determined world point locations of camera points arranged in a grid. An alternative approach would be to use the ray intersection calculation capability of the AABB tree implementation in [37].



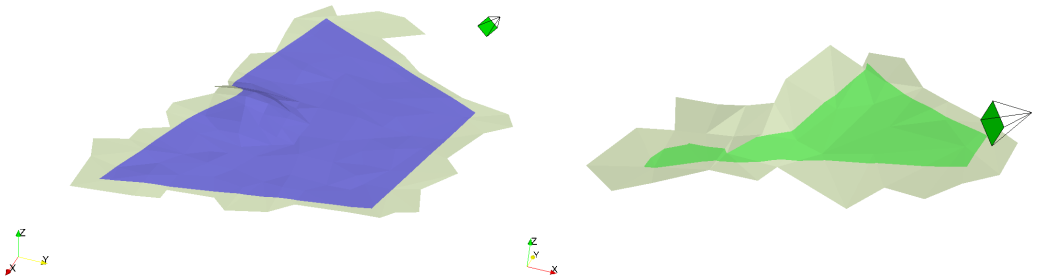
(a) Camera 1 (b) Camera 2

Figure 4.10: Camera points back projected onto a DEM for the same cameras as in Fig. 4.7.

4.2.4 Image Overlap Computation

The goal of the image overlap computation is to determine the region within one image, visible in another image by exploiting the DEM. To achieve this the computation is divided into two parts. The first part determines the portion of the DEM visible in the first view, while the second part calculates the visible area of this portion in the second view. The polygon operations in this section are performed with the 2D Regularized Boolean Set-Operations [39].

More precisely the following procedure is applied in part one. A view onto the DEM is generated for the first camera by using the approach given in Section 4.2.2. Due to the possibility of oblique views onto the DEM the polygons projected into an image may overlap. Therefore for each polygon projected into the first camera the segment tree of this view is queried for potentially overlapping polygons. The regions which are hidden by other polygons in front of the investigated one are removed by applying the Boolean set difference operation. Now the investigated polygon is clipped by the image rectangle. Finally all resulting polygons are back-projected onto the DEM by using the planes of the original polygons. Fig. 4.11 displays the obtained polygons for the same cameras as defined above.



(a) Camera 1 (b) Camera 2

Figure 4.11: Visible DEM polygons in blue for camera 1 and in green for camera 2. The cameras are defined as in Fig. 4.7.

In part two the previously obtained 3D polygons visible in the first camera and existing in the view of the second camera are clipped with the near clipping plane of the second camera. Then these polygons are projected into the second camera. Now the segment tree of the second view is queried for each projected polygon to obtain potentially overlapping polygons. In the next step the visible portion of one projected polygon is determined by removing regions, hidden from polygons in front of it by applying the Boolean set difference operation. The resulting polygon is finally clipped with the image rectangle. Fig. 4.12 shows the polygons visible in both above defined cameras back-projected onto the DEM.

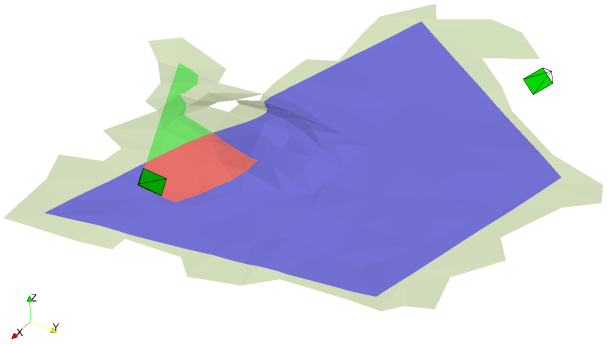
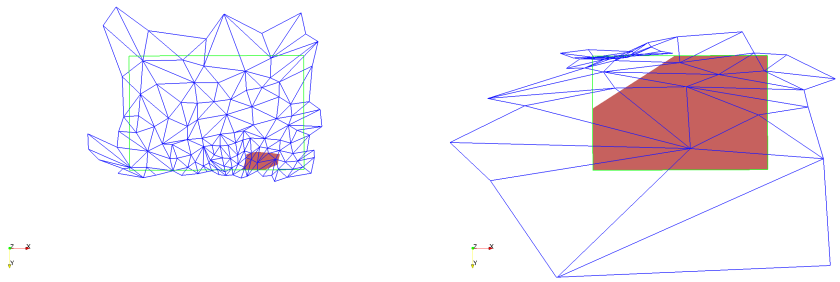


Figure 4.12: Camera 1 and 2 with their visible DEM regions. The region seen in both cameras is shown in red.

In addition Fig. 4.13 displays the overlapping region within the corresponding images. While the overlap occupies a large area in camera 2 the opposite situation occurs for camera 1. Therefore both images will share only few feature correspondences, this circumstance must be considered in the view selection stage by calculating a bad view similarity value.



(a) Camera 1 (b) Camera 2

Figure 4.13: Image of the overlap region for the above defined cameras given in red.

4.3 View Selection

The view selection stage determines images showing at least partly the same scene content, therefore only these images must be matched during the feature matching stage. This results in a huge performance gain compared to matching each image with all others, as already illustrated in the introduction. One way to perform view selection is to exhaustively compare the image content of all images. This can be achieved by using a subset of the detected features for matching each image against all others and calculating a view similarity value [19]. Alternatively view selection can also be performed by applying a vocabulary tree on the detected features [21, 54]. A disadvantage of the image based view selection approach is, that it generates erroneous results, when visually similar images display different parts of a scene [21].

To avoid any confusion between images, the applied view selection stage determines pairwise cameras i and j displaying the same part of a scene by exploiting `GPS` and `INS` data based on the approach given in [21]. More precisely a view overlap $\mathcal{O}_{ij}$ is calculated according Equation (4.1), where R_j^i denotes the scene region seen in camera j projected into camera i and clipped with the image rectangle of camera i. Furthermore A_i gives the area of the image rectangle of camera i and the function $a(\cdot)$ calculates the area of a

given image region. The same notation applies for the other camera, for an illustration see Fig. 4.14. This overlap criterion is image based and symmetric, e.g. $\mathcal{O}_{ij} = \mathcal{O}_{ji}$. Note that in general the regions R_j^i and R_i^j have different areas meaning $a(R_j^i) \neq a(R_i^j)$. Also the angle α_{ij} between the view vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ of both cameras is evaluated, so that the images from cameras with very distinct viewing directions are not considered for matching. So for the subsequent guided matching step, α_{ij} must be below a given threshold and $\mathcal{O}_{ij}$ must be above another threshold.

$$\mathcal{O}_{ij} = \min\left(\frac{a(R_j^i)}{A_i}, \frac{a(R_i^j)}{A_j}\right) \tag{4.1}$$

Two different strategies for computing the regions R_j^i and R_i^j are evaluated within this work. The first approximates the scene by a ground plane, while the second uses a `DEM` as scene representation. Both methods are briefly explained in the following.

4.3.1 Ground Plane Based View Selection

A fast way to perform view selection is to approximate the scene with a single ground plane. For computing the region R_j^i the image rectangle of camera j is back-projected onto a ground plane and then projected into camera i. Afterwards it is clipped by the image rectangle of camera i resulting in R_j^i, see Fig. 4.14.

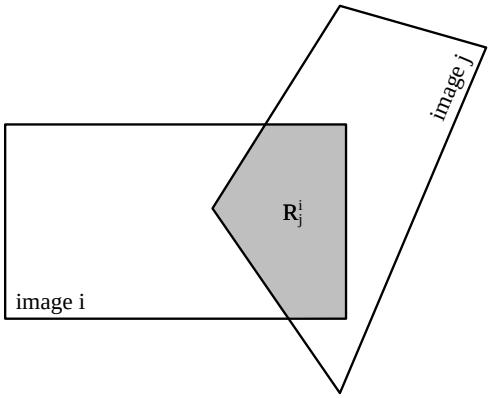


Figure 4.14: Ground plane based view selection. The gray shaded region R_j^i indicates that part of the ground plane which is seen from the cameras i and j.

For oblique imagery one has to consider two particular cases. The first case is when the horizon of the ground plane is visible in the image. Then instead of back-projecting the whole image rectangle onto the ground plane only a clipped rectangle is used, where

the clipping operation removes all parts above the horizon. The second problematic case exists when a part or the whole back-projected image border is behind camera i. This can be handled by clipping the back-projected border with the near clipping plane of camera i.

4.3.2 Digital Elevation Model Based View Selection

Another more involved view selection approach is to use a DEM as scene approximation, yielding to more accurate values for the regions R_j^i and R_i^j. At first a fast check is performed to determine whether it is possible that cameras i and j are displaying the same region of the DEM. For this test the view frustum of each camera is approximated by a minimum sized axis aligned bounding box, as specified in Section 4.2.2. If these bounding boxes are not overlapping, then it is impossible that both cameras are displaying the same part of the scene and the corresponding regions R_j^i and R_i^j are set to empty. Otherwise the image overlaps R_j^i and R_i^j are determined according to Section 4.2.4, see Fig. 4.13 for an example.

4.4 Guided Matching

ANN feature matching is widely used in SfM pipelines, e.g. in [19, 20, 22]. While it reduces the computational effort compared to exhaustive feature matching, it completely ignores any prior knowledge. Neither measured camera poses nor any scene knowledge are exploited. As consequence the time consumption for matching large scale images becomes quite large, which will be demonstrated in the experiments chapter. Recall also the numerical example given in the introduction, it illustrates the enormous performance advantage of predicting feature locations.

Within this work three different feature matching approaches are evaluated, which use the mentioned prior information to reduce the feature correspondence search region in one image for a given feature in the other image. After determining the search region an exact NN comparison selects the best matching feature within the region. Then the feature matches are checked by the distance ratio test and verified by the epipolar constraint. The query for features within a two dimensional area is efficiently performed by applying a kD tree containing all feature positions of the image. On the one hand the processing performance is improved, because a feature in one image is only compared with a small subset in the other image. On the other hand the probability for false matches due to

repetitive image structure is reduced, because any similar image region outside the search region is ignored. For example in aerial images of cities, rooftops and streets can cause many similar feature descriptors, which may confuse a feature matching procedure.

As stated in [8] the search region size depends on the uncertainty of the camera poses and on the accuracy of the scene approximation. These pose uncertainties cause correlations between feature locations in an image. Visual SLAM based methods exploit these correlations during feature matching to achieve a performance gain, cf. JCPL in Section 3.6.2.2.

To improve the above outlined procedure a matching with a small primary set is performed at first. With the obtained matches camera pose uncertainties are reduced, resulting in lower feature location prediction errors. Then matching with tighter search bounds for a large secondary set is performed, whereas the secondary set contains all extracted features. In the following the three evaluated matching strategies are given in more detail.

4.4.1 Homography Based Matcher

This approach is similar to that given in [23]. In their work they use only a few initial SIFT features of higher scale for an initial pairwise feature matching step between two images i and j. From the obtained feature correspondences an affine transformation between the two images i and j is estimated and this transformation is used to predict search regions for one image given feature positions in the other image.

In contrast to their work, here a homography is calculated between two images by using only the measured camera poses and a ground plane assumption. This homography is then used to guide the feature matching of the primary set. More precisely the following steps are applied for matching the features of image i with the features of image j. At first the region in image i, which is overlapping with image j is determined. This is performed by back-projecting the image j border onto the ground plane and then projecting the result into image i. The projected image j border is finally clipped by the image i border. Note that also here the two problematic cases mentioned in Section 4.3.1 must be considered to handle oblique imagery.

Then from the features within the overlapping region in image i a primary set is selected. For that purpose the bucketing technique for feature reduction proposed in [53] has been adapted, cf. Section 3.4.3. The bounding box of the overlapping region is overlain with a rectangular grid, where each cell corresponds to a bucket. Now randomly a non

empty bucket is selected and from that bucket randomly a feature point is chosen. That is repeated until the required amount of features for the primary set is reached, whereas each feature is selected at most once. For each primary set feature of image i a search region is calculated in image j by applying the previously calculated homography H, see Fig. 4.15. As mentioned above, the exact `NN` within the search region is determined and selected as match, if it fulfills the distance ratio test. When the distance ratio test fails, then no match can be determined for that feature in image i. Here a constant sized search region is applied for each feature.

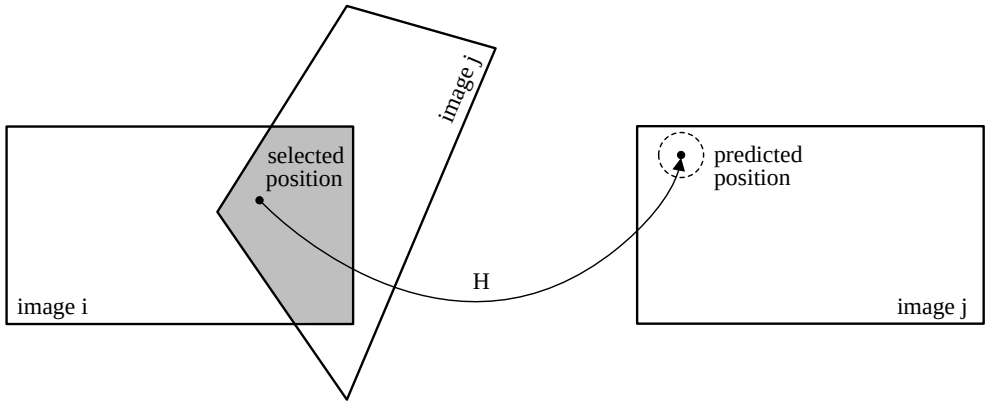


Figure 4.15: Homography based feature matching. The border of image j is back-projected onto an assumed ground plane and then the back-projected border is projected into image i. The features within the gray shaded region are used for matching. For each selected feature point in image i a homography H is applied to calculate a corresponding position in image j.

With the obtained primary set matches a homography and a fundamental matrix are robustly estimated by applying the `RANSAC` algorithm. Because the primary set is small, all features in the overlapping region of image i are used as secondary set for the subsequent matching, discarding the already existing correspondences. The estimated homography reduces the feature location prediction error, which allows to choose a smaller search radius. The search region area is further reduced by evaluating the estimated fundamental matrix. This is done by excluding features from matching, if their distance to the corresponding epipolar line is too large. Finally all matches are verified by the epipolar constraint.

4.4.2 Digital Elevation Model Based Matcher

The method introduced in the previous section uses a ground plane as scene approximation, which is reasonable for flat areas but not for mountainous regions. For the method described in this section the accuracy of the scene approximation is improved by applying a DEM as suggested in [8], see Fig. 4.16. So a feature point in image i is back-projected onto the DEM and then the resulting world point is projected into image j to determine the corresponding search region center for feature matching.

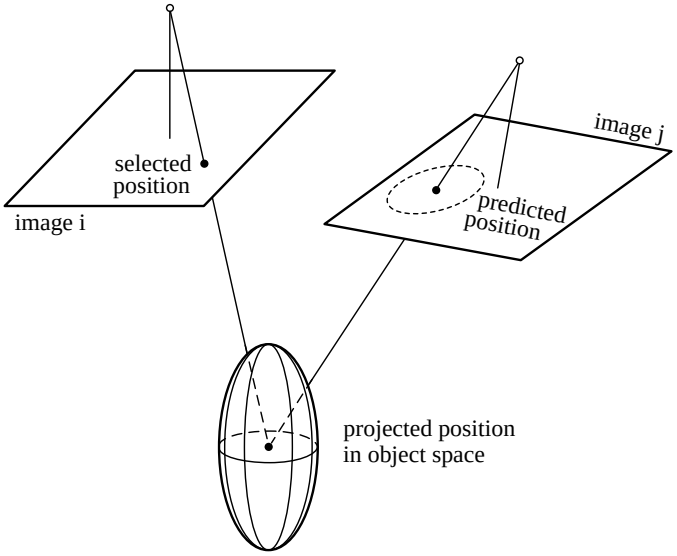


Figure 4.16: DEM based feature matching, from [8]. A selected feature point in image i is back-projected onto a DEM and afterwards projected into image j to obtain a predicted location. The sphere illustrates the uncertainty of the world point location.

Also here a primary set is used to reduce feature location prediction errors. Therefore the adapted bucketing technique of the previous section is applied to those image i features, which are projected into the visible region of image j. The primary set is now matched against the features in image j by utilizing a large search radius. For refining the relative poses between both cameras and DEM a bundle adjustment run is performed using the primary set correspondences. The bundle adjustment cost function formulation contains beside reprojection and camera pose error terms also a scene error component, cf. Section 4.5.2. Furthermore the fundamental matrix is robustly estimated by applying the RANSAC algorithm, although it would be also possible to calculate it from the refined camera poses in principle, see Equation (3.10). The separate estimation of the fundamental matrix has the advantage, that the computation is neither influenced by camera pose

measurements nor by the applied DEM.

The secondary set consists of all image i features projected into the visible region of image j, whereby the projections are performed with the refined camera poses. Therefore a smaller search radius can be used for the subsequent matching. By excluding those features from matching, which have a distance larger than some threshold from the corresponding epipolar line, the area of the search region is further reduced. The resulting matches are checked by the distance ratio test and an epipolar verification procedure.

4.4.3 Reconstructing Matcher

Also the method described in this section applies a DEM as scene approximation. Additionally the feature matching procedure is now performed incrementally instead of pairwise. So at the beginning two images are matched and the obtained correspondences are used to triangulate world points. Furthermore the remaining feature points are back-projected onto the DEM. Now a third image, displaying the same scene content as at least one of the other two images, is matched against the sparsely reconstructed scene by projecting the world points into the third image. Triangulated world points are more precisely known than the back-projected ones and therefore smaller search regions can be applied for them. The newly determined feature correspondences with already triangulated world points are used to refine the existing triangulation by using three view correspondences. Feature correspondences with back-projected world points are used to replace the back-projected points by triangulated ones. Now the unmatched features of the third image are again back-projected onto the DEM. That is repeated until all images are matched. Finally the world points seen in only one camera are removed from the scene. The order of the matching process is determined by the results obtained from the view selection stage. That means only images are matched showing a scene content, which is also visible in an already matched image. This approach is similar to the Visual SLAM methods, where also a feature map is maintained.

To consider feature correlations within this method, the following procedure is applied at each incremental matching step. At first the primary set is computed. Therefore the world points are projected into the image selected for matching, where any point lying outside the image border is ignored. Note that only those world points are projected into the image, which are visible in images showing the same scene content as the current image, as determined in the view selection stage. For each projected world point a quality value is assigned, which is the number of observations used for triangulating the world

point. It is assumed that world point locations triangulated with more camera points are more precisely known than others [56, 57]. Then the image is overlain by a grid, where each grid cell corresponds to a bucket. Now randomly a bucket containing projected world points is select. From this bucket a projected world point is randomly chosen, whereas only the world points with the highest quality value within the selected bucket are considered. That is repeated until the desired primary set size is reached, whereas each projected world point can be selected at most once. Then the primary set is matched, followed by a bundle adjustment run. In the bundle adjustment run only the pose of the current camera and the triangulated world points seen in the current camera are allowed to be refined. Also here a scene error is included in the bundle adjustment cost formulation. In the next step the secondary set is determined, where all projected world points visible in the image to be matched are selected. The few primary set matches are discarded. Finally the secondary set is matched using search regions of reduced size. The distance ratio test and the epipolar verification procedure are applied for matching the primary and secondary set.

4.5 Bundle Adjustment

The bundle adjustment stage within this work has a modular design, so individual cost terms and parameters are selectable, depending on the requirements of the current task. This flexibility in defining the optimization problem as well as some user experience are necessary to achieve good results [60, 61]. For example when `GCPs` are used for aligning the scene reconstruction, then it is useful to apply a `GPS` shift parameter [60]. This parameter allows a reconstruction with zero mean error in the world coordinate frame, because the whole reconstruction can be freely shifted to achieve `GCP` observations with low reprojection error. The scale ambiguity is always resolved by including the deviation between measured and adjusted camera position [2].

For solving the bundle adjustment problem the Ceres Solver library [58] has been selected. It is a non-linear least squares solver supporting different algorithms, whereas here the commonly used `LM` algorithm is applied. The Ceres Solver can be applied to small and large sparse problems and supports robust loss functions as well as local parameterizations. Beside a numeric differentiation also an automatic analytical differentiation of the cost function is possible. The following sections give an overview of the state vector components including their initialization, the utilized cost function terms and the outlier removal step.

4.5.1 State Vector

In this work the lens distortion parameters are precisely known, so the distortion can be removed from the images before they are processed in the pipeline. Therefore the projection equation without distortion is applied in calculating the reprojection error, see Equation (3.2). When the lens distortion parameters are unknown, then they can be determined during the bundle adjustment stage by applying projection equations including distortion terms, see for example Equation (3.3) and Equation (3.5). The distortion parameters are in this case additional variables to be determined during bundle adjustment, which is known as self-calibration [5].

As illustrated in Fig. 4.17 the observed `GPS` position value is not identical to the projection center of the camera. On the one hand the lever arm offsets $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$ must be considered, which are specified in `INS` frame coordinates within this work. On the other hand the returned `GPS` position can contain systematic errors like the depicted constant `GPS` bias $\mathbf{b}_{gps}$ as well as a time dependent `GPS` drift $\mathbf{d}_{gps} \cdot (t - t_0)$, which is calculated relative to t_0 [62]. Here the time dependent drift is neglected. Furthermore the measured `INS` orientation differs from the camera orientation by the boresight misalignment rotation R_{ins}^{cam}. Recall the notation introduced in Section 3.3.4.1, where the source coordinate system is given in the subscript and the target coordinate system is placed in the superscript.

The state vector $\mathbf{x}$ contains all parameters, which are allowed to be refined during the bundle adjustment stage. For cases where not all elements of the state vector are optimized, the Ceres Solver offers the possibility to force those parameters constant. In general only the needed parameters should be optimized to avoid possible correlations between them. In the following the state vector elements and their initialization are listed.

- **World Point Locations**

  All world point locations are added to the state vector. To initialize these locations the previously introduced linear triangulation algorithm is applied. The required feature correspondences are either obtained from the guided feature matching step for the automatically detected interest points or they are obtained from a manual configuration in the case of `CPs`. The purpose of `CPs` is to evaluate the accuracy of the bundle adjustment result by comparing the refined triangulated world point locations with the previously surveyed ground truth locations. In contrast `GCPs` are used improve the accuracy of the bundle adjustment result by adding fixed world points to the problem formulation. The `GCP` locations are obtained from a previous surveying step.

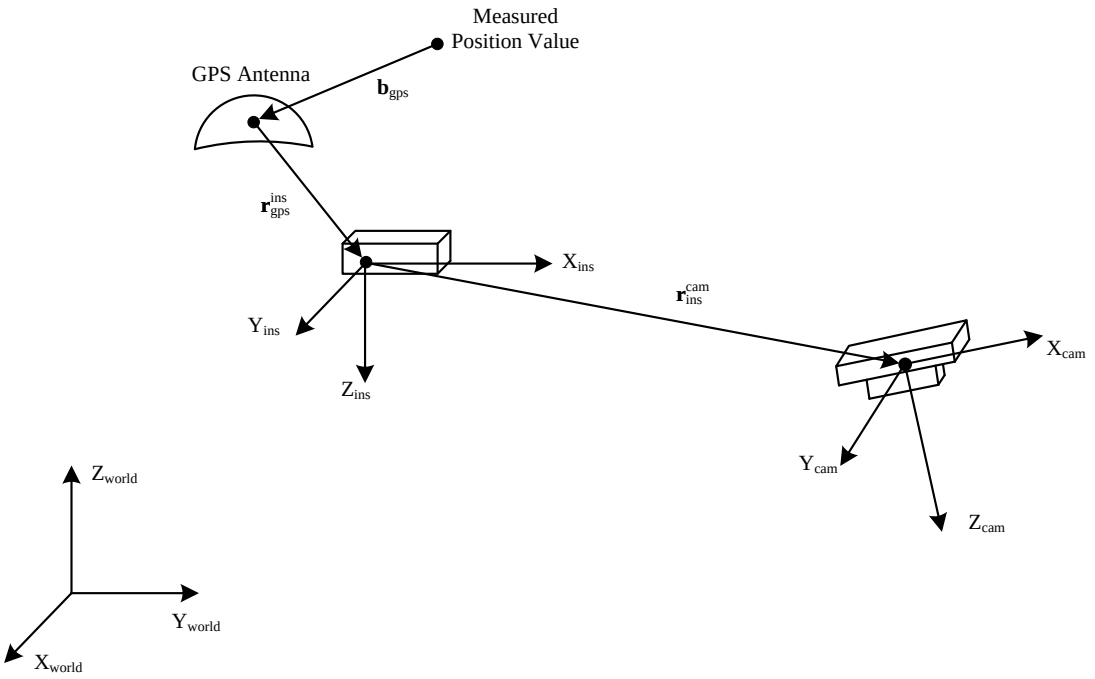


Figure 4.17: Modeled relationship between obtained position value, `GPS` antenna, `INS` and camera coordinate frame, adapted from [41]. The lever arm offsets $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$ indicate the shifts in sensor origins, while $\mathbf{b}_{gps}$ denotes a constant `GPS` bias.

- **Camera Poses**

  Furthermore the camera centers C_j and orientations $R_j{}_{world}^{cam}$ for all camera poses j are added to the state vector. They are initialized from measurements by considering the lever arm offsets $\mathbf{r}_{gps}^{ins}$ and $\mathbf{r}_{ins}^{cam}$, an eventually estimated `GPS` shift $\mathbf{b}_{gps}$ and the boresight misalignment R_{ins}^{cam}, see Fig. 4.17. This thesis handles the case, where obtained pose values are not accurate enough for skipping the bundle adjustment step to perform direct georeferencing.

- **Camera Intrinsics**

  Within this work laboratory calibrated cameras with accurately known intrinsics are used. However for the sake of completeness the intrinsics K_j for each camera pose j are added to the state vector. So an individual refinement of the intrinsics is possible, which allows the usage of different cameras within an aerial survey.

- **Calibration Parameters for Pose Measurement**
  If the lever arm offsets and the boresight misalignment are only approximately known, then they can be refined during the bundle adjustment stage. Therefore lever arm offset $\mathbf{r}_{gps}^{ins}$ and boresight misalignment R_{ins}^{cam} are included into the state vector and initialized with the approximations. The lever arm offset $\mathbf{r}_{ins}^{cam}$ is always fixed due to a correlation with $\mathbf{r}_{gps}^{ins}$ and thus not inserted into the state vector. This correlation exists, because both vectors are given with respect to the same `INS` coordinate frame within this work. Furthermore the `GPS` shift $\mathbf{b}_{gps}$ is included into the state vector and initialized with an estimate, for example zero. Be aware of, that depending on the concrete camera poses contained in a given dataset a correlation between $\mathbf{r}_{gps}^{ins}$ and $\mathbf{b}_{gps}$ may exists. For example if the camera view vectors are all perpendicular to a ground plane, then a correlation between the height components of $\mathbf{r}_{gps}^{ins}$ and $\mathbf{b}_{gps}$ exists.

World point positions and camera projection centers are represented by inhomogeneous coordinates, which is sufficient, because these locations are far from infinity. The camera orientations $R_{j\,world}^{cam}$ and the boresight misalignment R_{ins}^{cam} are parametrized by the angle-axis representation with the normalization mentioned in Section 3.8.5. An alternative would be the quaternion representation with the corresponding parametrization [5, 40].

4.5.2 Cost Function

The cost terms s_i depend on the state vector $\mathbf{x}$ and are of the type $s_i = \mathbf{\Delta z}_i(\mathbf{x})^T W_i \mathbf{\Delta z}_i(\mathbf{x})$, where the prediction error $\mathbf{\Delta z}_i(\mathbf{x}) = \underline{\mathbf{z}}_i - \mathbf{z}_i(\mathbf{x})$ is calculated as difference between an observation vector $\underline{\mathbf{z}}_i$ and the corresponding modeled prediction vector $\mathbf{z}_i(\mathbf{x})$. The prediction error is weighted by a matrix W_i, which is a diagonal matrix within this work for simplicity. Recall the earlier mentioned equivalence between weight matrix W_i and inverse covariance matrix Σ_i^{-1} for Gaussian distributed prediction errors. Note that the weights can be freely parametrized and therefore also set to zero to disable cost contributions. The following prediction errors $\mathbf{\Delta z}_i(\mathbf{x})$ are used to calculate the cost terms in the general case, whereas not always all terms are used for a concrete task.

- **Reprojection Error**
  To calculate the reprojection error of a world point in an image, the world point is projected into the image and the vectorial difference between projection and corresponding image observation is used as two dimensional prediction error. The

reprojection error is computed for all world point observations in each image. That means the world points obtained from triangulations of automatically determined feature correspondences as well as the world points triangulated from manual configured `CP` observations are used for reprojection error computations. Furthermore the reprojection errors of the surveyed `GCP` positions are also calculated.

- **Camera Pose Error**

  The scale ambiguity is implicitly resolved by adding prediction errors handling the deviations from measured position values and corresponding predicted position values, according Fig. 4.17. The predicted value is not directly contained in the state vector $\mathbf{x}$, therefore it must be calculated from the state vector elements. The computation is performed for a camera pose j by using the camera center C_j and considering the involved offsets as well as the camera rotation $R_{j\,world}^{cam}$ and the boresight misalignment R_{ins}^{cam}. Furthermore the difference rotation between measured and adjusted camera orientation is used as prediction error. This difference rotation is calculated from the measured and predicted `INS` frame orientations $R_{j\,world}^{ins} = (R_{ins}^{cam})^{-1} R_{j\,world}^{cam}$.

- **Calibration Parameter Deviations**

  To avoid an arbitrary change of the calibration parameters lever arm offset $\mathbf{r}_{gps}^{ins}$ and boresight misalignment R_{ins}^{cam} the deviation between measured and adjusted values are added as prediction errors to the bundle adjustment problem. Similarly the change of an initially estimated `GPS` bias $\mathbf{b}_{gps}$ is controlled by adding a prediction error formed by the difference between initial estimation and adjusted value. Furthermore the change of $\mathbf{b}_{gps}$ can be limited by adding `GCPs` to the bundle adjustment problem.

- **Camera Group Error**

  In some datasets groups of camera poses may exist, where the relative poses within the same group are very precisely known, while the relative poses between two different groups are imprecisely given. To exploit this information, it is possible to add prediction errors, which penalize deviations from the initial relative poses within a group.

- **Scene Error**

  For relating the position and orientation of the triangulated sparse structure to a scene approximation like a `DEM`, the distances between reconstructed points and scene approximation can be optionally included in the bundle adjustment problem. This

is mainly used in some of the above mentioned guided matching algorithms to refine camera poses relative to the DEM. Note that the usage of a DEM for reducing the number of GCPs is in general possible and has been theoretically investigated in [63].

Whenever difference rotations are used as prediction errors within this work, a computed difference rotation matrix is converted into the angle-axis representation, resulting in a three dimensional residual. Additionally a diagonal weight matrix $W_i = \text{diag}\{w_i, w_i, w_i\}$ with identical entries w_i is applied, meaning that the error is determined by the weighted rotation angle and that the rotation axis is not considered.

4.5.3 Outlier Removal

In general outliers can deteriorate the accuracy of the final sparse reconstruction of a scene, so on the one hand their amount should be reduced and on the other hand robust algorithms should be applied. To reduce the amount of outliers the feature correspondences are verified by the epipolar constraint, as mentioned in Section 4.4. In addition multiple bundle adjustment runs are performed, where each run is followed by an outlier removal step [20]. Outliers are identified by comparing the reprojection errors of a world point with a threshold. Is one of the associated reprojection errors above the threshold, then the world point and its observations are removed from the bundle adjustment problem. Furthermore the distances between world points and DEM are computed, see Section 4.2. If the calculated distance of a world point is above a fixed threshold, then again the world point with the corresponding image observations are removed. Finally the utilized Ceres Solver supports robust loss functions $\rho(s)$ for correctly considering the observation PDFs, therefore the bundle adjustment stage is robust against outliers.

4.6 Discussion

In the following the most important aspects of the designed AT pipeline are discussed. At the beginning of the pipeline the user is assisted in preparing the input data for the subsequent stages by an configuration utility. This utility provides already a guidance for parametrizing CPs and GCPs by suggesting approximate locations of their observations within the images. In future versions the usability can be further improved, when only one observation has to be specified by the operator. All other observations should then be automatically determined by an area based image matching algorithm leading to subpixel accurate locations as suggested in [64]. Sometimes CPs and GCPs are explicitly indicated

in a scene, for example with white rectangles. Here an automatic or semi-automatic fitting algorithm would on the one hand ease the configuration and on the other hand improve the accuracy.

The above outlined AT pipeline is focused on a fast processing of the distinct stages, where special attention is paid on the feature matching stage. One way to accelerate the matching step is to apply view selection, therefore it is not necessary to match each aerial image with all others. Within this work view selection is performed by exploiting the measured camera poses and a scene approximation. This avoids a time consuming image based view selection. Camera pose imprecisions can be considered by lowering the threshold for the image overlap $\mathcal{O}_{ij}$. Additionally the matching of images obtained from cameras with very different view vectors is prevented. Therefore the applied view selection strategy fits the requirements for AT.

A further way to speed up the feature matching stage is to reduce the number of features to be compared. Here feature location prediction is applied to place a search region into an image to be matched and only the features within the search region are compared. Furthermore it is attempt to reduce the size of the search region by exploiting the correlations between feature locations within one image, like in the Visual SLAM based methods. Additionally the epipolar lines are evaluated to reduce the search region sizes further. As shown in the following experiments chapter relatively few cross track matches are generated, which results in non radial symmetric reprojection error distributions. Also the average number of observations per reconstructed world point is relatively low. A well known property of the SIFT detector/descriptor is its sensitivity to perspective distortions. Consider for example a camera with a view vector perpendicular to a ground plane imaging the wall of a building. Depending on the camera position different amounts of perspective distortion occur, although the view vector orientation does not change. Here a more robust method should be applied to generate more cross track matches and to increase the average number of observations, see for example [25, 65]. The applied matching strategy tries to match all image features, so as side product one obtains a sparse scene reconstruction after the bundle adjustment stage. When only the camera poses should be determined, a feature matching within some predefined image regions would be sufficient, cf. von Gruber positions [64].

As mentioned earlier the applied feature matching strategies utilize some ideas of the Visual SLAM methods. However a direct usage of these algorithms without any modifications is not practicable due to the following reasons.

- While the Visual SLAM algorithms are designed for real time processing on images containing comparatively few feature points, in AT a batch processing on large scale images with huge amounts of interest points is performed, e.g. 100000 feature points per image. For example it is reported in [11] that the AM approach scales poorly with the number of features.

- Usually Visual SLAM methods estimate the camera poses, see Fig. 3.29. This estimation would fail at the beginning of a new flight line due to missing images during the turn of the aircraft. So the camera motion model cannot be applied here, instead GPS and IMU measurements must be evaluated to obtain camera pose estimates.

- Sweeping camera systems may produce images without significant overlap, cf. the VisionMap A3 aerial camera system in section Section 5.2.1. So each image within a sweep shows a different part of the scene. In these cases the incremental map building approach would fail, because it is not possible to triangulate world points from two subsequent images.

- Visual SLAM algorithms do not exploit scene approximations like a DEM and therefore neglect valuable information.

Finally the modular design of the bundle adjustment stage provides the flexibility for the user to select individual parameters, cost terms and loss functions. Therefore the pipeline can be applied to many different aerial survey datasets. A useful extension of the bundle adjustment stage would be the covariance calculation of all estimated parameters and their visualization with confidence spheres [61].

Chapter 5

Experiments

Contents

This chapter presents the evaluation results of the outlined `AT` pipeline on two different datasets. The first dataset consists of large format aerial images generated with the Microsoft UltraCamX camera pointing straight downwards to the earth. In contrast the second dataset contains much smaller aerial images obtained from the VisionMap A3 camera, where also oblique images are included.

5.1 Microsoft UltraCamX

At first the design of the large format Microsoft UltraCamX camera is briefly outlined. Afterwards the dataset description section gives an overview of the covered terrain. Finally major components of the `AT` pipeline are evaluated, which are the view selection, guided matching and bundle adjustment stages.

5.1.1 Camera Design

Microsoft UltraCamX [66] is a high resolution digital aerial camera system based on a multi cone multi sensor concept, see Fig. 5.1. It is built up of eight independent camera cones, whereas four of them are used to generate a high resolution panchromatic (gray scale) image. The other four cones generate a lower resolution multispectral image, consisting of

the channels Red, Green, Blue and Near Infrared. In total 13 Charge Coupled Device (CCD) sensors are installed, nine for the panchromatic image.

Figure 5.1: Microsoft UltraCamX camera cone placement, from [66]. The four cones with color filters are used for the multispectral channel, while the others are utilized for the panchromatic channel.

Fig. 5.2 illustrates the stitching procedure for combining the images of the nine sensors. The four CCD sensors of the Master Cone are combined with the CCD sensors of the other three cones to form the high resolution panchromatic image. Furthermore a temperature compensation is performed within this step to reduce geometric distortions. The obtained panchromatic and multispectral images are finally merged to a high resolution color image by applying a process called pan sharpening.

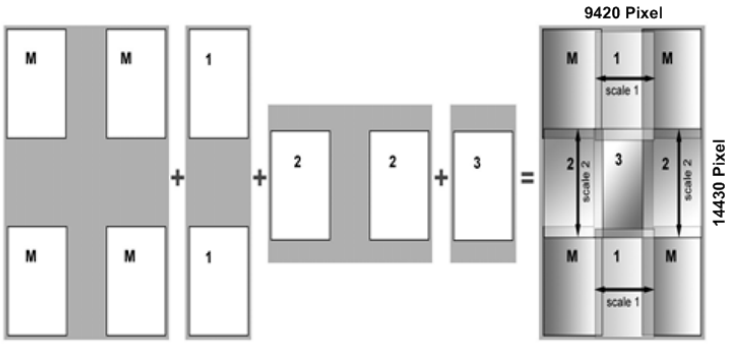


Figure 5.2: Microsoft UltraCamX stitching scheme for the panchromatic channel, from [57]. It is depicted how the individual CCD sensors of the Master Cone (M) and those of the three other camera cones are combined to a high resolution panchromatic image.

The camera is already manufacturer calibrated. Nevertheless a self calibration during

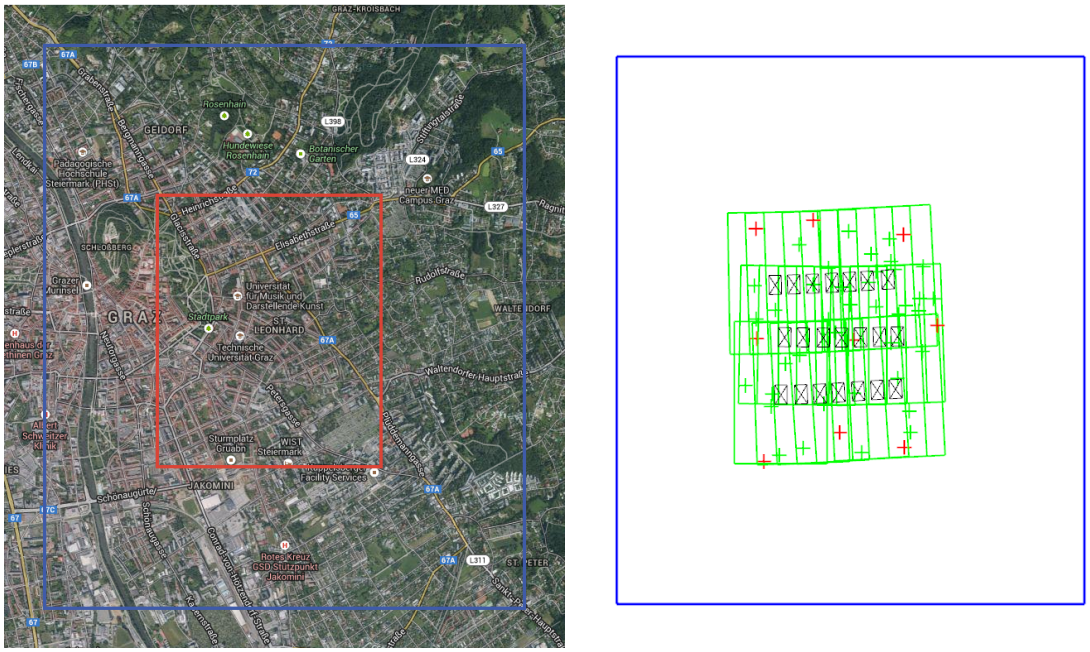
bundle adjustment is recommended due to small radial symmetric distortions introduced by atmospheric refraction while operation. Furthermore special UltraCamX self calibration parameters have been designed [57]. Table 5.1 summarizes the technical data of the UltraCamX used for the evaluation of the pipeline. While the UltraCamX uses an external GPS receiver, newer improved UltraCam versions are optionally equipped with an embedded GPS receiver. Furthermore the newer cameras generate images of higher resolution [7].

Panchromatic Channel	
Multi cone multi sensor concept	4 camera heads
Image size in pixel (cross track/along track)	14430 pixel × 9420 pixel
Physical pixel size	7.2 μm
Physical image format (cross track/along track)	103.9 mm × 67.8 mm
Focal length	100 mm
Lens aperture	$f = 1/5.6$
Angle of view (cross track/along track)	$55° \times 37°$
Multispectral Channel	
Four channels (Red, Green, Blue, Near Infrared)	4 camera heads
Image size in pixel (cross track/along track)	4992 pixel × 3328 pixel
Physical pixel size	7.2 μm
Physical image format (cross track/along track)	34.7 mm × 23.9 mm
Focal length	33 mm
Lens aperture	$f = 1/4$
General	
Shutter speed options	1/500 s - 1/32 s
Forward motion compensation	Time delayed integration controlled, 50 pixels
Frame rate per second	1 frame in 1.35 s
ADC resolution	14 bit (16384 levels)
Radiometric resolution	$>$ 12 bit/channel

Table 5.1: Microsoft UltraCamX technical data, from [66].

5.1.2 Dataset Description

To evaluate the distinct pipeline stages with the UltraCamX imagery, a small set of 21 images with a Ground Sampling Distance (GSD) of 8 cm has been utilized, see Fig. 5.3. CPs and GCPs are configured for determining the accuracy of the whole pipeline, where the ground truth locations were obtained from [33].



(a) (b)

Figure 5.3: Microsoft UltraCamX dataset. In (a) an orthophoto of the test site is shown, which has been obtained from Google Maps. The blue border depicts the area covered by the generated DEM while the red rectangle indicates the region contained in the images. (b) shows the individual image borders projected onto a ground plane as green polygons. The green crosses mark the CP locations while the red ones depict the GCP locations. The camera poses are given by the black symbols.

The images cover a flat region within the city Graz in Austria. Either the DEM shown in Fig. 5.4 or a ground plane with a height of 410 m above the WGS84 ellipsoid is used to approximate the scene in the individual processing stages. The Tagged Image File Format (TIFF) tags of the images contain directly the corresponding camera poses, but without the post processing step mentioned in Section 3.3.4.1, therefore they are imprecise [42]. Section 3.3.4.2 contains an overview of factors influencing the accuracy of the camera pose measurement.

5.1.3 View Selection

Two view selection methods are compared within this work, one approximates the scene with a ground plane and the other operates on a DEM. The view overlap $\mathcal{O}_{ij}$ between two images i and j is calculated according to Equation (4.1), additionally the angle between the

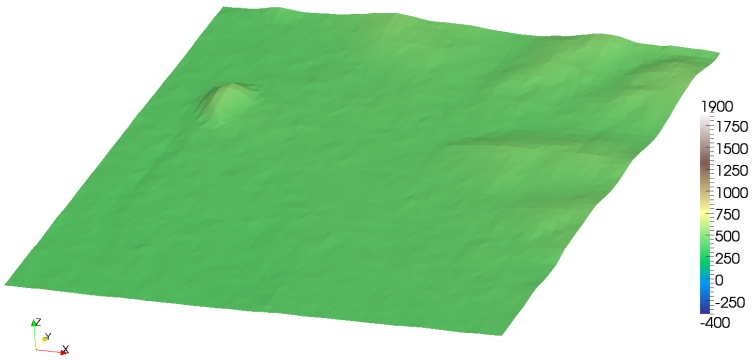
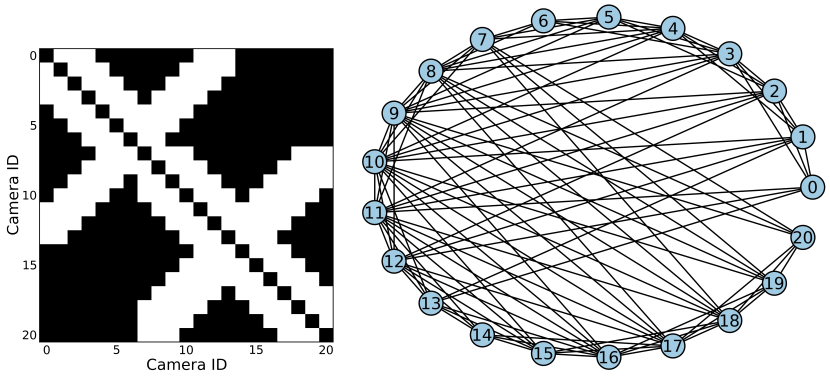


Figure 5.4: `DEM` applied for the Microsoft UltraCamX dataset. The region covered by the images is characterized by the following height values above the `WGS84` ellipsoid: min. 393.5 m, max. 471.4 m and median 416.4 m. The entire `DEM` has the following heights above `WGS84`: min. 387.5 m, max. 516.5 m and median 415.4 m.

corresponding view vectors α_{ij} is computed. Now a image pair is matched in the following guided matching step, when $\mathcal{O}_{ij} > 0.3$ and $\alpha_{ij} \leq 30°$. The result can be represented either by an adjacency matrix [21], where the entry (i, j) is set to 1 when image i and j have to be matched, or set to 0 in the opposite case. Another representation is the geometry graph [22], where each node corresponds to an image and the edges connect the images with overlap. For this dataset the results of the ground plane based view selection are identical to the `DEM` based, due to the flat terrain, see Fig. 5.5.



(a) (b)

Figure 5.5: View selection results for the Microsoft UltraCamX dataset. (a) shows the adjacency matrix, where the images i and j have to be matched when the entry (i, j) is white. (b) depicts the corresponding geometry graph.

5.1.4 Guided Matching

The considered pipeline uses `SIFT` features, which are extracted by the SiftGPU implementation [59]. On average ≈ 85000 features per image are obtained by the applied setting. To assess the performance of the three implemented guided matching methods, they are compared with the FLANN library [28]. This library contains `ANN` feature matching algorithms and additionally provides an auto tuning option to select the best suited method and corresponding parameters for a given dataset. For the evaluated UltraCamX imagery the hierarchical k-means tree algorithm has been selected by the auto tuning procedure. All evaluated matchers use the Euclidean distance for descriptor comparisons and apply the distance ratio test to eliminate correspondences likely to be outliers. More precisely for a positive feature correspondence the distance between both feature descriptors must be lower than 0.5 and the distance ratio between best and second best match has to be lower than 0.6. The obtained matches are verified by the epipolar constraint requiring an Sampson error $d_\perp$ below 0.25 pixel. The implemented pairwise matcher restrict the secondary set search region along the epipolar line, allowing an Sampson error $d_\perp$ of up to 15 pixel during the search.

Ground plane and `DEM` matcher use a primary set search radius of 200 pixel, while the reconstructing matcher applies a search radius of 750 pixels. For the pairwise matchers the primary set prediction error is determined by the accuracy of the scene approximation as well as the relative poses between both involved cameras and scene approximation [8]. In contrast the primary set prediction error for the reconstructing matcher depends beside the accuracy of the scene approximation on the relative poses between the camera to be matched, partial reconstruction and scene approximation. Furthermore the orientation and location of the partial reconstruction depend on the initial pairwise matching and the course of the incremental matching, so the prediction errors for the primary sets are in general larger. All implemented matcher use a secondary set search radius of 100 pixels. Assuming an ideal refinement procedure, the secondary set prediction error is determined by inaccuracies of the scene approximation, e.g. `DEM`, which of course doesn't consider buildings, trees, etc. . A too small secondary set search radius would therefore limit the height of the reconstructed 3D structure measured relative to the scene approximation, for example roofs of high buildings are possibly not reconstructed. The applied search region sizes are summarized in Table 5.2.

Table 5.3 summarizes the processing times obtained on an Intel Core $i7-3930K$ CPU @ 3.20GHz, where the matching task is always performed by a single thread, only

Method	Primary Set Radius	Secondary Set Radius	Secondary Set Epipolar Distance
Ground Plane Matcher	200 pixel	100 pixel	15 pixel
DEM Matcher	200 pixel	100 pixel	15 pixel
Reconstructing Matcher	750 pixel	100 pixel	–

Table 5.2: Parametrized search region sizes of the different matcher for the Microsoft UltraCamX dataset.

bundle adjustment and verification are implemented multithreaded. It can be seen that the time consumption of the implemented matcher is one magnitude lower than the baseline method. Note that the FLANN library is already highly optimized, while the implemented matcher have potential for further improvements.

Method	Processing Time Matching	Bundle Adj.	Verification	Correspondences All	Verified
FLANN Auto Tuned	28.33 min	–	1.55 min	1235446	260159
Ground Plane Matcher	1.72 min	–	1.81 min	1475062	355782
DEM Matcher	2.48 min	0.32 min	1.81 min	1431911	349015
Reconstructing Matcher	1.77 min	0.09 min	1.24 min	365681	97241

Table 5.3: Performance comparison of different matching methods for the Microsoft UltraCamX dataset.

FLANN returns fewer verified feature correspondences than the ground plane and DEM matcher. On the one hand only an ANN search is performed, while the other matcher perform an exact NN search within their search regions. On the other hand the implemented matcher predict feature locations and exclude features located far away from the predicted position from the matching process, so the probability for false matches is reduced. Further experiments showed, that the second reason has the largest impact on the amount of obtained correspondences. The amount of resulting correspondences for the reconstructing matcher is not directly comparable to the other, because here an incremental reconstruction is performed instead of a pairwise matching.

As mentioned earlier the three implemented matcher perform at first a primary set matching and improve their prediction afterwards by either estimating a homography or performing a bundle adjustment run. To see the effect of this refinement process the verified secondary set correspondences have been used to calculate the prediction errors. For the ground plane matcher the prediction errors obtained from the calculated homographies are compared with the prediction errors obtained from the estimated homographies. Similarly for the DEM matcher and the reconstructing matcher the prediction errors obtained

from the original camera poses are compared with that obtained from the refined camera poses. Table 5.4 summarizes the results. It can be seen that all matcher improve their prediction to the same error. Furthermore the requirement of a larger primary set search radius for the reconstructing matcher is confirmed.

Method	Before Refinement	After Refinement
Ground Plane Matcher	95.7 pixel	15.9 pixel
`DEM` Matcher	94.5 pixel	15.6 pixel
Reconstructing Matcher	306.5 pixel	13.5 pixel

Table 5.4: Median prediction errors of the different matcher for the Microsoft UltraCamX dataset.

Table 5.5 compares the average number of image observations per world point. All matcher perform equally well, except the reconstructing matcher, which has a lower value. Note that a higher number of observations per world point improves the triangulation accuracy [56, 57].

Method	Avg. Number of Observations
FLANN Auto Tuned	2.43
Ground Plane Matcher	2.50
`DEM` Matcher	2.49
Reconstructing Matcher	2.07

Table 5.5: Average number of observations per world point for the Microsoft UltraCamX dataset.

5.1.5 Bundle Adjustment

The bundle adjustment stage is evaluated by the usage of `CPs`. Therefore ground truth world point locations and their observations within the images are configured. These observations are used to triangulate world points, which are then refined during the bundle adjustment stage and afterwards compared with the ground truth locations. In the following the resulting position differences are termed as `CP` errors. In contrast `GCPs` are used to include ground truth information into the bundle adjustment stage, for example to align the result. Also there world point locations and corresponding observations are configured. Usually markers like painted white rectangles are placed into a test area, whereas their locations are precisely surveyed, for example with an accuracy of 1 cm. These marker are then automatically fitted in the images to achieve observations with subpixel accuracy [60]. Within this work a less involved and more imprecise procedure is applied. The world point locations are obtained from [33] and their image observations are manually configured.

As mentioned earlier the camera coordinates are only approximately known [42], so two different experiments are performed to see the effect of GCPs. In the first one, all 48 configured points are used as CPs, while in the second experiment 9 points are used as GCPs and the remaining 39 points are utilized as CPs, see Fig. 5.3(b).

5.1.5.1 Results without Ground Control Points

For the following evaluation without GCPs the bundle adjustment cost function contains the deviations between measured and adjusted camera positions as well as the reprojection errors. Experiments showed that depending on the used feature matcher different numbers of bundle adjustment iterations with subsequent outlier removal steps lead to the minimum CP error. Therefore the evaluation has been performed with three iterations for the FLANN and ground plane matcher and only two iterations for the DEM and reconstructing matcher. In all cases a squared loss has been applied.

At first mean, standard deviation and maximum value of the reprojection error x- and y-component have been evaluated. Let's denote the observation of the i-th camera point with $\mathbf{x}_i = [x_i, y_i, 1]^T$ and the projection of the corresponding world point into the same camera by $\hat{\mathbf{x}}_i = [\hat{x}_i, \hat{y}_i, 1]^T$. For the overall m camera points observed in the different images the mean value μ_{re_x} and the standard deviation σ_{re_x} of the reprojection error x-component are determined as stated in Equation (5.1). The corresponding maximal difference is computed as $\max_i |\hat{x}_i - x_i|$. Calculations for the y-component are performed accordingly.

$$\mu_{re_x} = \frac{1}{m}\sum_{i=1}^{m}(\hat{x}_i - x_i) \qquad \sigma_{re_x} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{x}_i - x_i - \mu_{re_x})^2} \tag{5.1}$$

Afterwards mean, standard deviation and maximum CP error have been evaluated for the X-, Y- and Z-component as well as for the L2 norm of the whole error vector. Therefore the i-th measured CP ground truth location is denoted with $\mathbf{X}_i = [X_i, Y_i, Z_i, 1]^T$ and the adjusted position is given with $\hat{\mathbf{X}}_i = [\hat{X}_i, \hat{Y}_i, \hat{Z}_i, 1]^T$. Mean μ_{cp_x} and standard deviation σ_{cp_x} for the X-component of the n CPs are computed as given in 5.2, while the maximum difference is determined by $\max_i \left|\hat{X}_i - X_i\right|$. The calculations for the other components as well as for the vector norm are performed accordingly.

$$\mu_{cp_x} = \frac{1}{n}\sum_{i=1}^{n}(\hat{X}_i - X_i) \qquad \sigma_{cp_x} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{X}_i - X_i - \mu_{cp_x})^2} \tag{5.2}$$

Table 5.6 lists the obtained reprojection error mean, standard deviation and maximum values per x- and y-component. It can be observed, that the maximum reprojection error is larger for the DEM and reconstructing matcher, due to fewer bundle adjustment iterations.

Feature Matcher		Mean	Std. Dev.	Max. Diff.
FLANN Auto Tuned	x	0.00 pixel	0.10 pixel	0.35 pixel
	y	0.00 pixel	0.05 pixel	0.35 pixel
Ground Plane Matcher	x	0.00 pixel	0.09 pixel	0.34 pixel
	y	0.00 pixel	0.05 pixel	0.34 pixel
DEM Matcher	x	0.00 pixel	0.12 pixel	0.47 pixel
	y	0.00 pixel	0.07 pixel	0.47 pixel
Reconstructing Matcher	x	0.00 pixel	0.14 pixel	0.46 pixel
	y	0.00 pixel	0.04 pixel	0.46 pixel

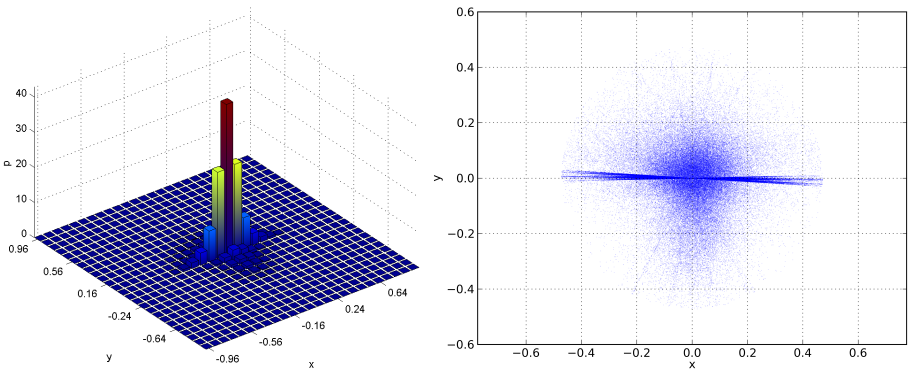
Table 5.6: Comparison of reprojection errors for the Microsoft UltraCamX dataset without GCPs.

The resulting mean, standard deviation and maximum CP errors for the X-, Y- and Z-component as well as for the L2 norm $\|\cdot\|$ are listed in Table 5.7. It can be seen that the mean and median length of the CP error vectors are similar for all four matcher, however the error is very large. This is originated in the less precisely configured CPs and the only approximately known camera poses.

Feature Matcher		Mean	Median	Std. Dev.	Max. Diff.
FLANN Auto Tuned	X	−3.184 m	−3.413 m	1.239 m	5.653 m
	Y	0.572 m	0.656 m	1.452 m	4.236 m
	Z	−4.991 m	−4.976 m	2.107 m	8.828 m
	$\|\cdot\|$	6.368 m	6.579 m	–	9.318 m
Ground Plane Matcher	X	−2.972 m	−3.117 m	1.125 m	4.945 m
	Y	−1.922 m	−1.998 m	1.371 m	6.737 m
	Z	−5.331 m	−5.508 m	2.368 m	9.561 m
	$\|\cdot\|$	6.778 m	6.591 m	–	11.244 m
DEM Matcher	X	−2.714 m	−2.989 m	1.404 m	5.837 m
	Y	2.315 m	2.346 m	1.507 m	5.016 m
	Z	−4.970 m	−4.989 m	1.257 m	7.863 m
	$\|\cdot\|$	6.473 m	6.715 m	–	8.416 m
Reconstructing Matcher	X	−3.371 m	−3.926 m	1.763 m	6.407 m
	Y	1.953 m	1.758 m	2.003 m	5.603 m
	Z	−3.500 m	−3.416 m	2.474 m	9.251 m
	$\|\cdot\|$	6.092 m	6.318 m	–	9.794 m

Table 5.7: Comparison of CP errors for the Microsoft UltraCamX dataset without GCPs.

In the following only one matcher is evaluated in more detail, because the outcomes for the others are similar. For that purpose the DEM matcher has been selected, due to its applicability for flat and mountainous terrains. Fig. 5.6 shows the reprojection error histogram after the bundle adjustment stage, it can be observed that the error is orientated along the image x direction. This is reasoned in the fact, that the vast majority of correspondences are located on epipolar lines orientated nearly parallel to the y axes of the images. To see why this circumstance causes such a histogram, consider a pairwise match between two images. When both measured feature points are located on the epipolar lines, then the bundle adjustment stage can modify the world point location in a way, that the reprojection error becomes zero. In contrast when the features are not located on the epipolar lines, then it is not possible to achieve a zero reprojection error without a relative movement between both cameras. Now there are usually many correspondences between two images, therefore there exists no single movement to achieve zero reprojection error for all correspondences. Furthermore camera position deviations are penalized, which causes that such movements are avoided. So one reason for remaining reprojection errors are the perpendicular distances between feature points and corresponding epipolar lines. The observed reprojection errors are perpendicular to the epipolar lines.

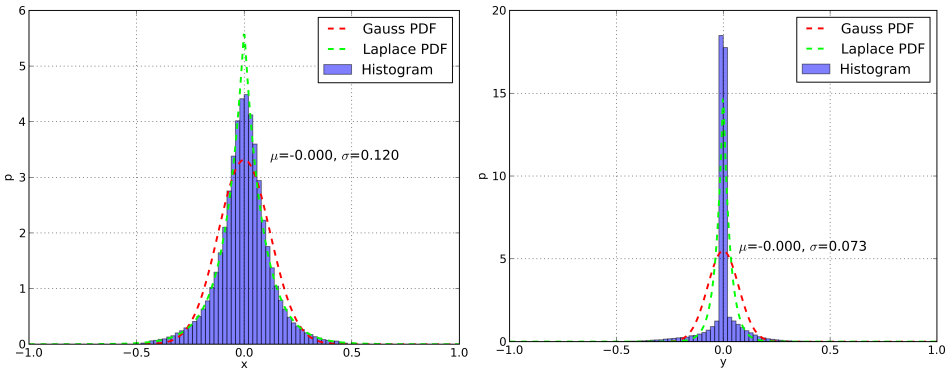


(a) (b)

Figure 5.6: Reprojection error histogram and scatter plot for the Microsoft UltraCamX dataset without GCPs. While (a) shows the reprojection error histogram, (b) displays the corresponding scatter plot.

In Fig. 5.7 histograms of the reprojection error x- and y-component after the bundle adjustment stage are given. It can be observed that the reprojection error x-component can be roughly approximated by a Gaussian distributed Random Variable (RV). So a squared

loss is a reasonable choice for the observed reprojection errors. A better approximation would be a Laplace distribution, which would lead to a Huber or Pseudo-Huber loss function. Note that the obtained standard deviation $\sigma = 0.12$ is about half the threshold applied for the epipolar verification, which is 0.25 pixel.



(a) (b)

Figure 5.7: Reprojection error histograms of the x- and y-component for the Microsoft UltraCamX dataset without `GCPs`. In (a) the histogram of the reprojection error x-component is given, while (b) shows the corresponding histogram for the y-component. Additionally mean μ and standard deviation σ are given. The red and green curves depict `PDFs` of Gauss and Laplace distributed `RVs` respectively. They have been generated as `MLEs` for the shown errors.

Fig. 5.8 displays the mean reprojection error vector depending on the image location. The plot has been generated by using all 21 images. There are small systematic errors observable, like a small radial component for large distances from the center of projection.

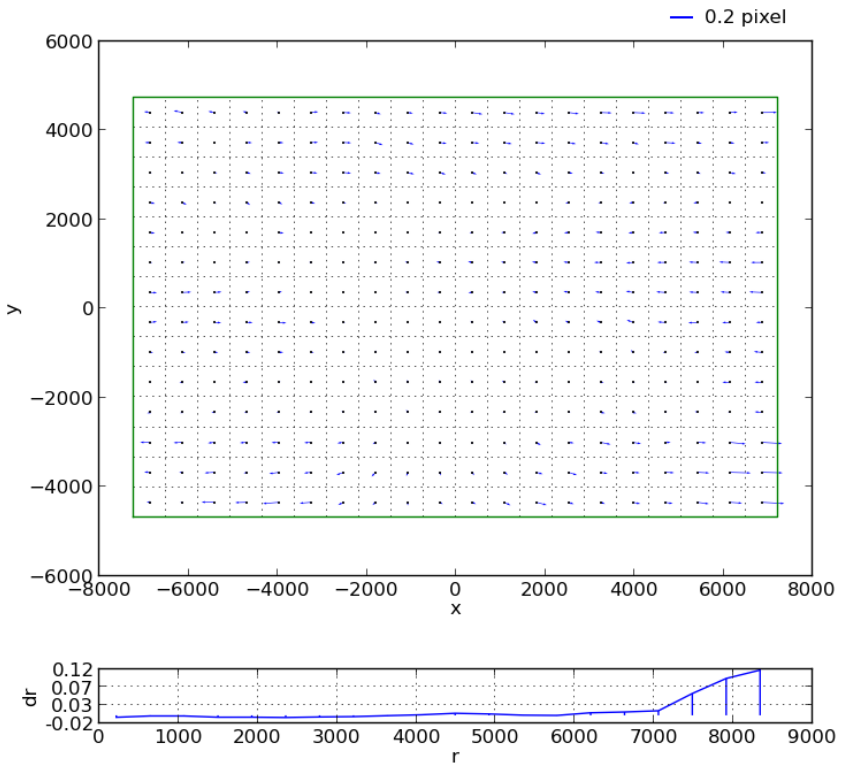


Figure 5.8: Reprojection error depending on the image location for the Microsoft UltraCamX dataset without GCPs. The upper part shows the mean reprojection error vector for each image tile, while the lower part displays the reprojection error in radial direction dr depending on the distance r from the principal point.

5.1.5.2 Results with Ground Control Points

To reduce the large CP errors observed in the last section, 9 CPs are used as GCPs. Also here the bundle adjustment cost function contains reprojection errors and camera position residuals. In contrast to the previous experiment the GPS shift parameter $\mathbf{b}_{gps}$ is now allowed to be freely adjusted. More precisely a single arbitrarily $\mathbf{b}_{gps}$ value for all involved cameras is allowed to be selected from the optimizer. For the FLANN and ground plane matcher correspondences three bundle adjustment iterations with subsequent outlier removal steps have been applied, while for the DEM and reconstructing matcher correspondences two and one iterations have been used respectively. The resulting reprojection errors for the automatically detected feature points are summarized in Table 5.8. Standard deviation and maximum reprojection error differ for the individual matcher, due to the different number of outlier removal steps. Furthermore the reprojection errors are larger than in the experiment without GCPs, because the configured weight for the GCP

reprojection errors is much larger than the corresponding weight for the other reprojection errors and the GCP observations are only imprecisely given.

Feature Matcher		Mean	Std. Dev.	Max. Diff.
FLANN Auto Tuned	x	0.01 pixel	0.50 pixel	1.64 pixel
	y	−0.03 pixel	0.26 pixel	1.63 pixel
Ground Plane Matcher	x	0.01 pixel	0.51 pixel	1.70 pixel
	y	−0.03 pixel	0.28 pixel	1.70 pixel
DEM Matcher	x	0.01 pixel	0.74 pixel	2.51 pixel
	y	−0.05 pixel	0.36 pixel	2.49 pixel
Reconstructing Matcher	x	0.00 pixel	0.99 pixel	4.28 pixel
	y	−0.02 pixel	0.37 pixel	4.27 pixel

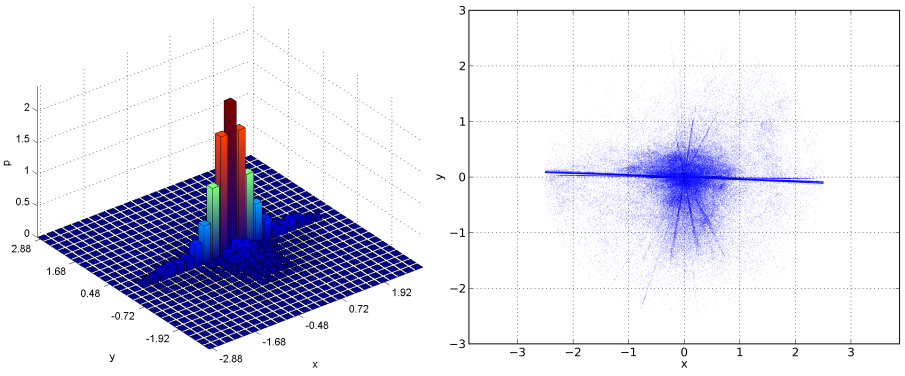
Table 5.8: Comparison of reprojection errors for the Microsoft UltraCamX dataset with 9 GCPs.

Table 5.9 compares the CP errors obtained for the different matcher with a result published in [42]. All matcher achieve a similar accuracy, however the results of [42] are superior. One major reason for the difference is, that they use CPs and GCPs with precisely determined coordinates.

Feature Matcher		Mean	Median	Std. Dev.	Max. Diff.
FLANN Auto Tuned	X	0.047 m	0.010 m	0.827 m	3.218 m
	Y	−0.470 m	−0.255 m	1.717 m	8.067 m
	Z	0.235 m	0.175 m	1.805 m	6.125 m
	$\|\cdot\|$	1.657 m	0.656 m	–	9.471 m
Ground Plane Matcher	X	0.053 m	0.027 m	0.828 m	3.202 m
	Y	−0.416 m	−0.246 m	1.704 m	7.965 m
	Z	0.313 m	0.057 m	1.518 m	5.482 m
	$\|\cdot\|$	1.565 m	0.735 m	–	9.157 m
DEM Matcher	X	0.051 m	0.099 m	0.838 m	3.201 m
	Y	−0.346 m	−0.242 m	1.694 m	7.531 m
	Z	0.522 m	0.099 m	1.320 m	4.979 m
	$\|\cdot\|$	1.563 m	0.730 m	–	8.364 m
Reconstructing Matcher	X	0.097 m	0.098 m	0.867 m	3.386 m
	Y	−0.247 m	−0.197 m	1.726 m	7.032 m
	Z	0.764 m	0.299 m	1.706 m	6.562 m
	$\|\cdot\|$	1.757 m	0.773 m	–	7.826 m
Results from [42]	X	0.033 m	–	0.008 m	0.061 m
	Y	0.037 m	–	0.01 m	0.067 m
	Z	0.048 m	–	0.005 m	0.059 m

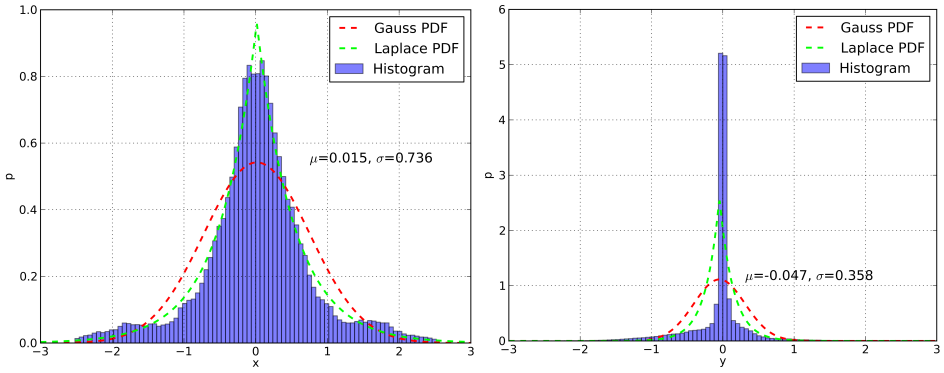
Table 5.9: Comparison of CP errors for the Microsoft UltraCamX dataset with 9 GCPs.

The corresponding reprojection error histograms for the DEM matcher are given in Fig. 5.9 and Fig. 5.10. As already mentioned the error has increased, additionally the reprojection error x-component cannot be approximated by a Gaussian RV anymore, while the approximation with a Laplacian distribution is still valid.



(a) (b)

Figure 5.9: Reprojection error histogram and scatter plot for the Microsoft UltraCamX dataset with 9 GCPs. While (a) shows the reprojection error histogram, (b) displays the corresponding scatter plot.



(a) (b)

Figure 5.10: Reprojection error histograms of the x- and y-component for the Microsoft UltraCamX dataset with 9 GCPs. In (a) the histogram of the reprojection error x-component is given, while (b) shows the corresponding histogram for the y-component. Additionally mean μ and standard deviation σ are given. The red and green curves depict PDFs of Gauss and Laplace distributed RVs respectively. They have been generated as MLEs for the shown errors.

Of course the reprojection error depending on the image location has also increased, see Fig. 5.11. Furthermore the orientation of the error vectors has changed.

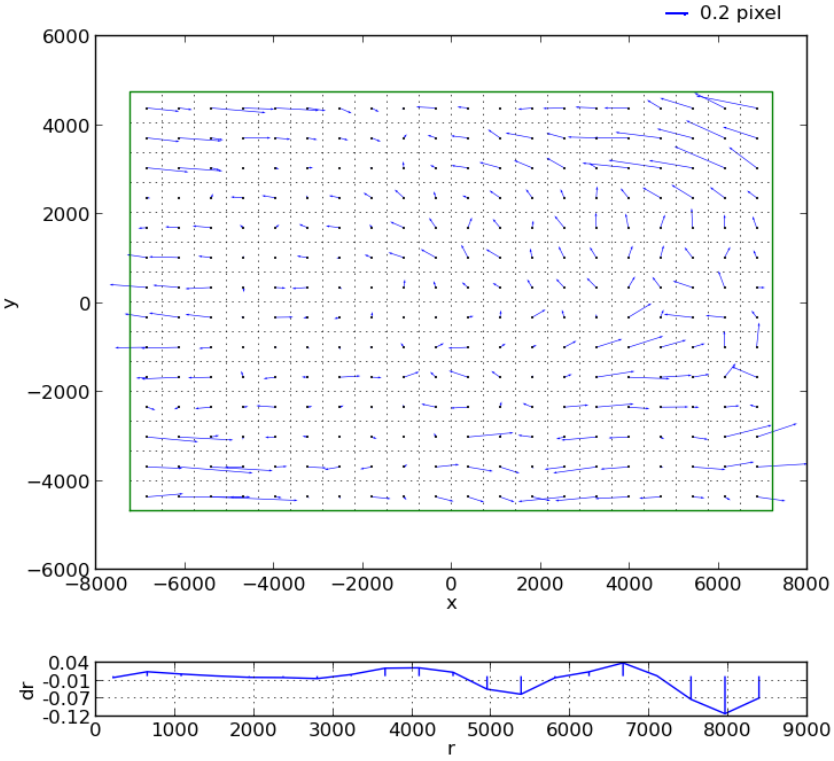


Figure 5.11: Reprojection error depending on the image location for the Microsoft UltraCamX dataset with 9 `GCPs`. The upper part shows the mean reprojection error vector for each image tile, while the lower part displays the reprojection error in radial direction dr depending on the distance r from the principal point.

Fig. 5.12 displays the resulting sparse scene reconstruction obtained from the DEM matcher experiment. It can be observed that the buildings are well reconstructed, especially in the center of the scene where many images overlap.

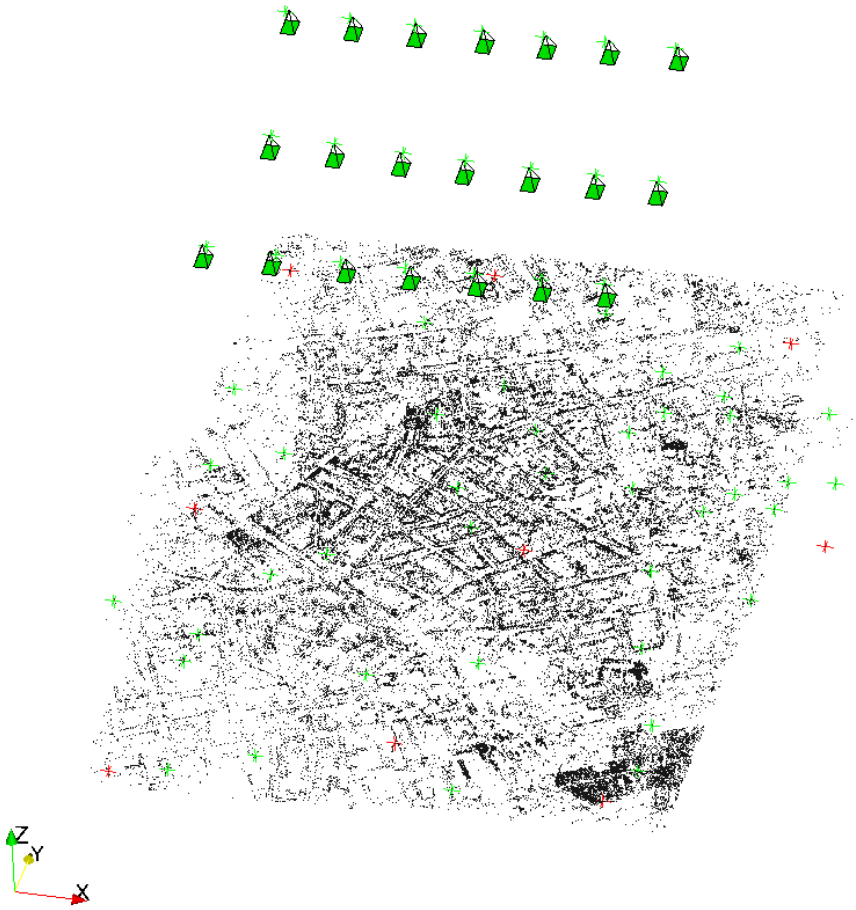


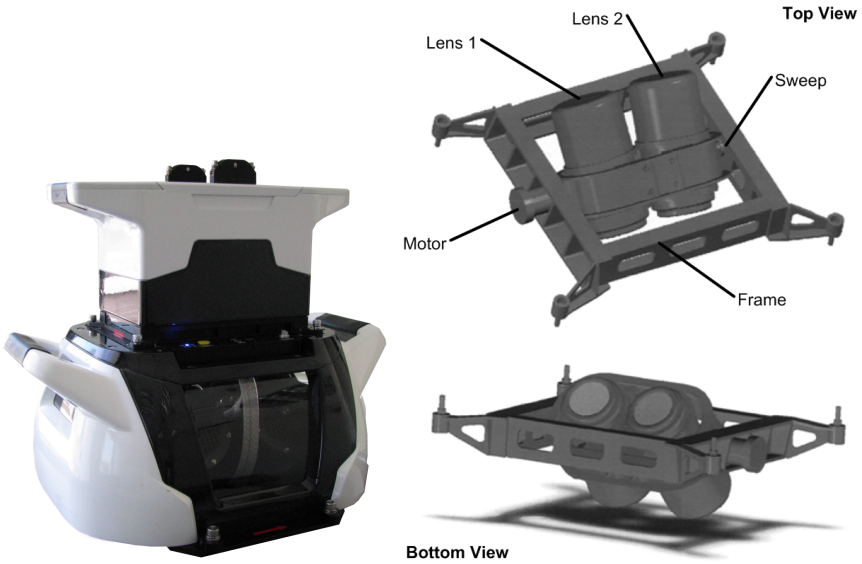
Figure 5.12: Reconstructed scene for the Microsoft UltraCamX dataset with 9 GCPs. The red crosses indicate the GCP locations, while the green crosses within the scene show the CP positions. The green crosses beside the cameras indicate the centers of projection as given in the TIFF tags.

5.2 VisionMap A3

This section starts with a short introduction of the VisionMap A3 camera design followed by a dataset description. The same AT pipeline elements as for the UltraCamX camera are also evaluated here, that are the view selection, guided matching and bundle adjustment stages.

5.2.1 Camera Design

The VisionMap A3 aerial camera system [67] consists of two lenses, each having one RGB CCD sensor, see Fig. 5.13. Both lenses are mechanically coupled and mounted onto a sweep mechanism. The camera is usually orientated in a way that the sweep direction is perpendicular to the flight direction. A large focal length of $f = 300$ mm allows high flying altitudes for a given ground resolution, while a mirror based folding optical system reduces the lens dimensions. The orientation of the interior mirror is controlled to compensate the sweep motion of the lenses during the exposure as well as the forward motion and vibrations of the aircraft.



(a) (b)

Figure 5.13: VisionMap A3 aerial camera system, from [67, 68]. A photo of the camera is given in (a), while (b) shows a drawing of the sweep mechanism.

During the flight the lenses sweep from one side to the other and generate up to 27 double frames, as illustrated in Fig. 5.14. All double frames within one sweep can be combined to a Super Large Frame. That leads to a large total Field Of View (FOV) of 104° cross track and allows therefore large distances between flight strips. Table 5.10 summarizes the technical data of the VisionMap A3 aerial camera system. As stated, the camera contains already an integrated GPS receiver.

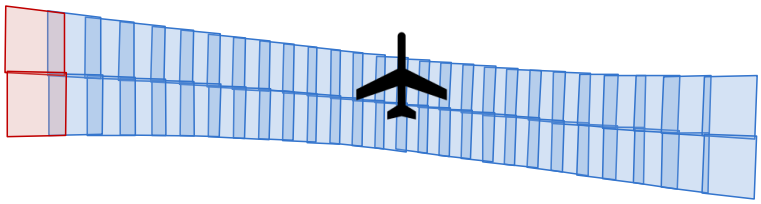


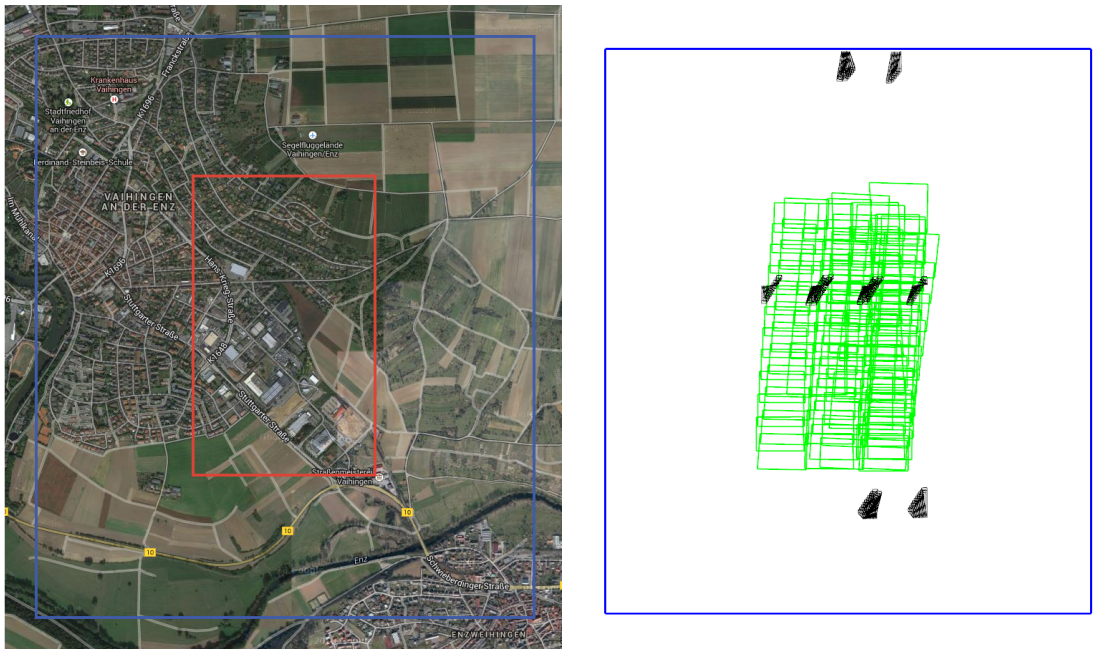
Figure 5.14: Illustration of the VisionMap Super Large Frame consisting of all images obtained during a single sweep. The red polygons indicate the last double frame of the sweep.

CCD	2 × Kodak KAI-11002
Single frame size	4006 pixel × 2666 pixel
Double frame size	∼ 7812 pixel × 2666 pixel
Single frames overlap (along track)	∼ 100 pixel
Synthetic Super Large Frame	∼ 62517 pixel × 7812 pixel
Single lens FOV	6.9° × 4.6°
Max. sweep FOV	104°
Number of double frames per max. FOV	27
Sweep time	$3 - 4s$
Color	RGB
Radiometric resolution	12 bit/channel
Pixel size	9 μm
Optics	Dual reflective lens system
Focal length	300 mm
Aperture	$f/4.5$
Forward motion compensation	Mirror based optical compensation and stabilization
Sweep motion compensation	
Vibration (Stabilizer)	
Frame rate	7 fps
Readout rate	155 MB/s
On-Board compression	Jpeg2000 (lossless)
Data collection capacity	5 hours
Storage type	Flash Drive
Storage capacity	500 GB
GPS (Omni Star supported)	Dual-frequency GPS + L-Band
Total weight	∼ 30 kg
Camera unit size	70 cm × 70 cm × 30 cm
Operating temperature	$-30°C$ to $+45°C$

Table 5.10: VisionMap A3 technical data, from [67].

5.2.2 Dataset Description

A subset of 116 images with a GSD of 6 cm from the dataset examined in [60] has been used to evaluate the pipeline with the VisionMap A3 imagery, see Fig. 5.15. Here mainly the matching performance is assessed, so neither CPs nor GCPs are used.



(a) (b)

Figure 5.15: VisionMap A3 dataset. In (a) an orthophoto of the test site is shown, which has been obtained from Google Maps. The blue border depicts the area covered by the generated DEM while the red rectangle indicates the region contained in the images. (b) shows the individual image borders projected onto the ground plane as green polygons. The camera poses are given by the black symbols. The 116 cameras are grouped into 8 blocks.

Camera center C_j and rotation R_j for camera j are given in a report file. These poses are already precisely available, as shown in the following sections. The images cover a terrain with some height variations at Vaihingen in Germany. Either a ground plane with a height of 330 m above the WGS84 ellipsoid or the DEM displayed in Fig. 5.16 is applied to approximate the scene.

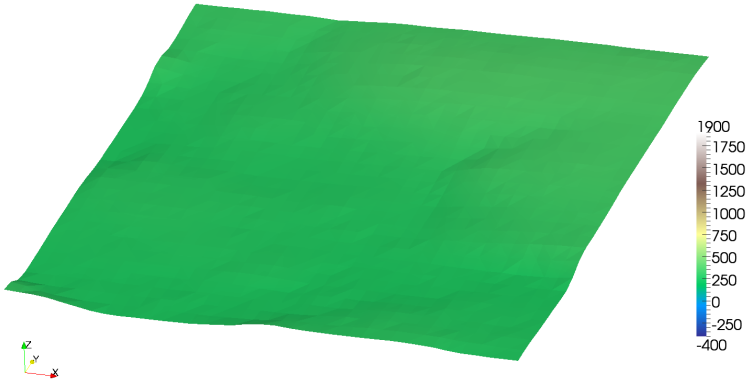


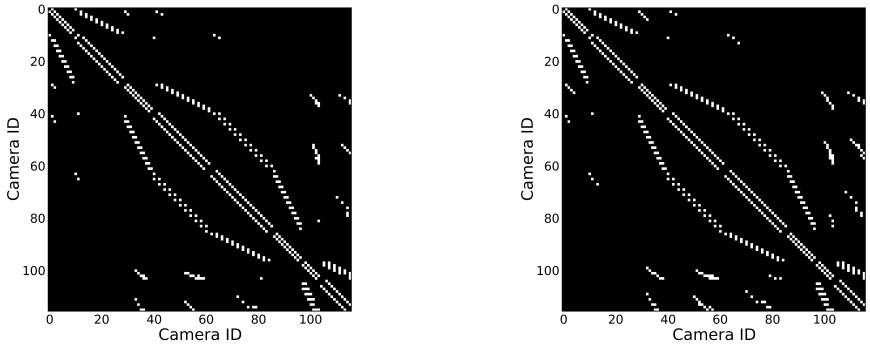
Figure 5.16: DEM applied for the VisionMap A3 dataset. The region covered by the images is characterized by the following height values above the WGS84 ellipsoid: min. 247.5 m, max. 349.5 m and median 266.5 m. The entire DEM has the following heights above WGS84: min. 244.5 m, max. 364.5 m and median 273.5 m.

5.2.3 View Selection

For the view selection stage a minimum view overlap $\mathcal{O}_{ij}$ of 0.2 and a maximal angle between view vectors α_{ij} of 30° is parametrized. The ground plane based and the DEM based algorithm achieve slightly different adjacency matrices, see Fig. 5.17. This is reasoned in the larger height variations of the terrain compared to the previous dataset, leading to inaccurate results for the ground plane based overlap calculation. Therefore the adjacency matrix of the DEM based view selection procedure is applied in the subsequent stages.

5.2.4 Guided Matching

The individual VisionMap A3 images are relatively small compared to the Microsoft UltraCamX images, therefore the SIFT detector threshold has been lowered to achieve a higher amount of features for the matcher evaluation. With this modification on average ≈ 40000 features are extracted per image. The FLANN auto tuning algorithm selected also here the hierarchical k-means tree algorithm for the VisionMap A3 images. The settings of all matchers are equal to that used in the UltraCamX experiment, only the search region sizes differ as can be seen in Table 5.11. The ground plane matcher uses a large primary set search region, because the initial ground plane is a bad scene approximation. Nevertheless the homography estimation procedure within the matcher chooses indirectly planes locally better approximating the scene, therefore the secondary set search region is still small. In contrast the primary set search regions for the DEM matcher and reconstructing matcher



(a) (b)

Figure 5.17: View selection results for the VisionMap A3 dataset. (a) shows the adjacency matrix for the ground plane based view selection, while (b) displays the adjacency matrix for the DEM based view selection procedure.

are already as low as the secondary set search regions due to the precisely known camera poses and the good scene approximation by the applied DEM.

	Primary Set	Secondary Set	
Method	Radius	Radius	Epipolar Distance
Ground Plane Matcher	750 pixel	100 pixel	15 pixel
DEM Matcher	100 pixel	100 pixel	15 pixel
Reconstructing Matcher	100 pixel	100 pixel	–

Table 5.11: Parametrized search region sizes of the different matcher for the VisionMap dataset.

In Table 5.12 the performance evaluation results for the different matcher are summarized. The matching time differences are mainly reasoned in the different primary set search radii. Because of a higher feature density compared to the UltraCamX experiment, the search radius has a huge impact on the query times of the utilized kD tree and the subsequent matching procedure. It can be observed that the DEM matcher is the fastest, followed by the reconstructing matcher. The reconstructing matcher has a higher time consumption for the included bundle adjustment run, because the cameras within a sweep are handled as a camera group, assuming that the relative poses of the cameras within a group are more precisely known than between groups. Therefore always the cameras of the whole group are optimized, which consumes more time than optimizing a single camera. The count of verified features differ due to the same reasons as mentioned in the

UltraCamX experiment.

	Processing Time			Correspondences	
Method	Matching	Bundle Adj.	Verification	All	Verified
FLANN Auto Tuned	30.62 min	–	5.59 min	2447922	734556
Ground Plane Matcher	19.80 min	–	5.79 min	2552365	797030
DEM Matcher	6.15 min	0.18 min	5.88 min	2593432	805062
Reconstructing Matcher	5.58 min	8.94 min	5.73 min	1103030	493860

Table 5.12: Performance comparison of different matching methods for the VisionMap dataset.

The feature location prediction errors are given in Table 5.13. It can be observed, that after the refinement all matcher are able to predict feature locations with low error. Furthermore due to accurately known camera positions and orientations and the good scene approximation by the applied DEM, the primary set prediction errors are low for the DEM and reconstructing matcher. Although the median prediction error before refinement for the ground plane matcher is much lower than the corresponding primary set search radius, the search radius cannot be further reduced. That is reasoned in the circumstance that the prediction error between some image pairs is much larger than the median value. So a matching between those pairs would become impossible, which would result in a poor scene reconstruction.

Method	Before Refinement	After Refinement
Ground Plane Matcher	148.6 pixel	1.4 pixel
DEM Matcher	10.2 pixel	3.9 pixel
Reconstructing Matcher	15.4 pixel	2.1 pixel

Table 5.13: Median prediction errors of the different matcher for the VisionMap dataset.

5.2.5 Bundle Adjustment

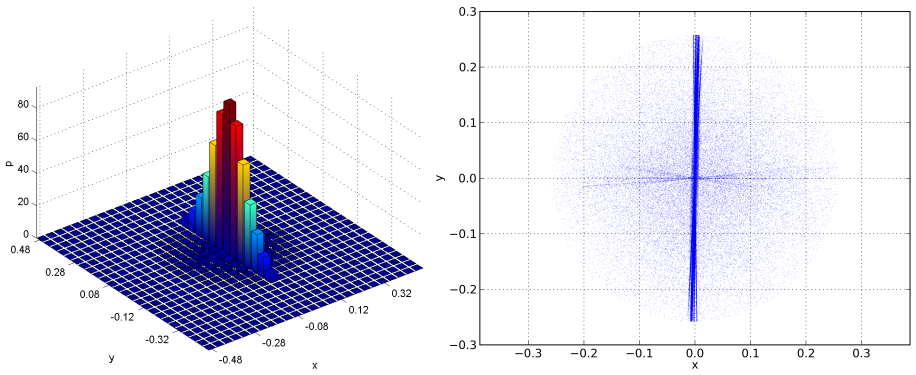
For the VisionMap dataset the bundle adjustment cost function contains reprojection errors, camera position residuals and camera group errors. A camera group is composed of all camera poses within one sweep. Only the reprojection errors are considered for evaluating the bundle adjustment stage, so neither CPs nor GCPs are used. For all feature matcher three bundle adjustment iterations have been performed and a squared loss has been chosen for the reprojection error. Table 5.14 lists the resulting reprojection errors for the investigated feature matchers, which are all similar. In the following the DEM matcher

is evaluated in more detail.

Feature Matcher		Mean	Std. Dev.	Max. Diff.
FLANN Auto Tuned	x	0.00 pixel	0.05 pixel	0.29 pixel
	y	0.00 pixel	0.10 pixel	0.29 pixel
Ground Plane Matcher	x	0.00 pixel	0.06 pixel	0.29 pixel
	y	0.00 pixel	0.10 pixel	0.29 pixel
`DEM` Matcher	x	0.00 pixel	0.04 pixel	0.26 pixel
	y	0.00 pixel	0.10 pixel	0.26 pixel
Reconstructing Matcher	x	0.00 pixel	0.03 pixel	0.22 pixel
	y	0.00 pixel	0.09 pixel	0.23 pixel

Table 5.14: Comparison of reprojection errors for the VisionMap A3 dataset.

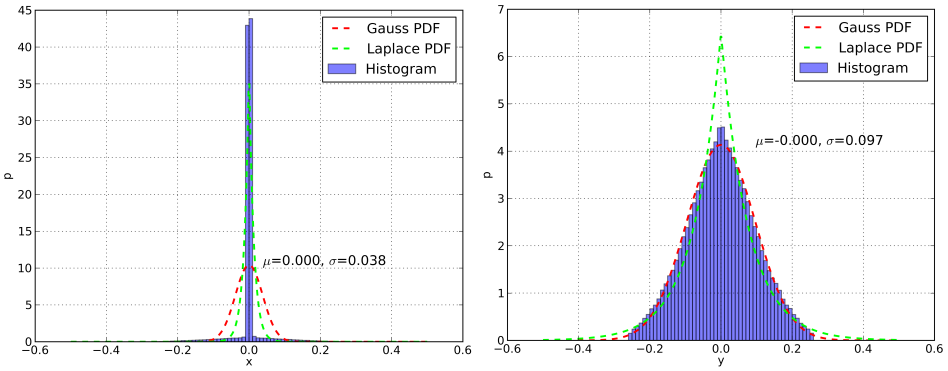
The reprojection error histogram obtained after three bundle adjustment iterations is given in Fig. 5.18. It has the same shape as for the Microsoft UltraCamX dataset, but in contrast the vast majority of epipolar lines corresponding to feature matches are orientated parallel to the x-axis. That results in reprojection errors orientated along the y-axis.



(a) (b)

Figure 5.18: Reprojection error histogram and scatter plot for the VisionMap A3 dataset. While (a) shows the reprojection error histogram, (b) displays the corresponding scatter plot.

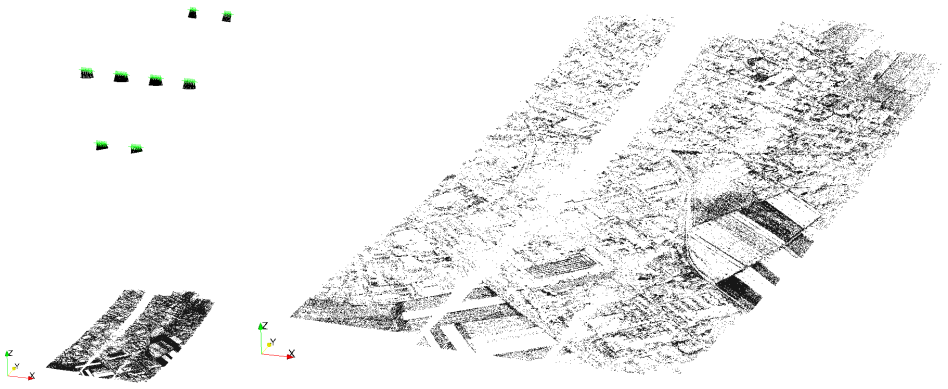
In Fig. 5.19 the histograms for the reprojection error x- and y-component are given. It can be seen that the Gaussian distribution approximates the y-component well, so also here a squared loss is selectable.



(a) (b)

Figure 5.19: Reprojection error histograms of the x- and y-component for the VisionMap A3 dataset. In (a) the histogram of the reprojection error x-component is given, while (b) shows the corresponding histogram for the y-component. Additionally mean μ and standard deviation σ are given. The red and green curves depict `PDFs` of Gauss and Laplace distributed `RVs` respectively. They have been generated as `MLEs` for the shown errors.

Fig. 5.20 shows the reconstructed scene. It can be observed that the sparse structure is split into two parts. The empty region originates from the circumstance that in this area few images overlap and those which are oblique with varying scale. As consequence too few feature correspondences are obtained.



(a) (b)

Figure 5.20: Reconstructed scene for the VisionMap A3 dataset. In (a) an overview containing the 116 camera poses is given, while (b) displays a close-up view of the scene.

The reconstruction of a larger VisionMap dataset containing 3450 images is given in Fig. 5.21. It contains also the 116 images of the smaller dataset described above. Also here exist holes, which are located mainly in the boundary regions.

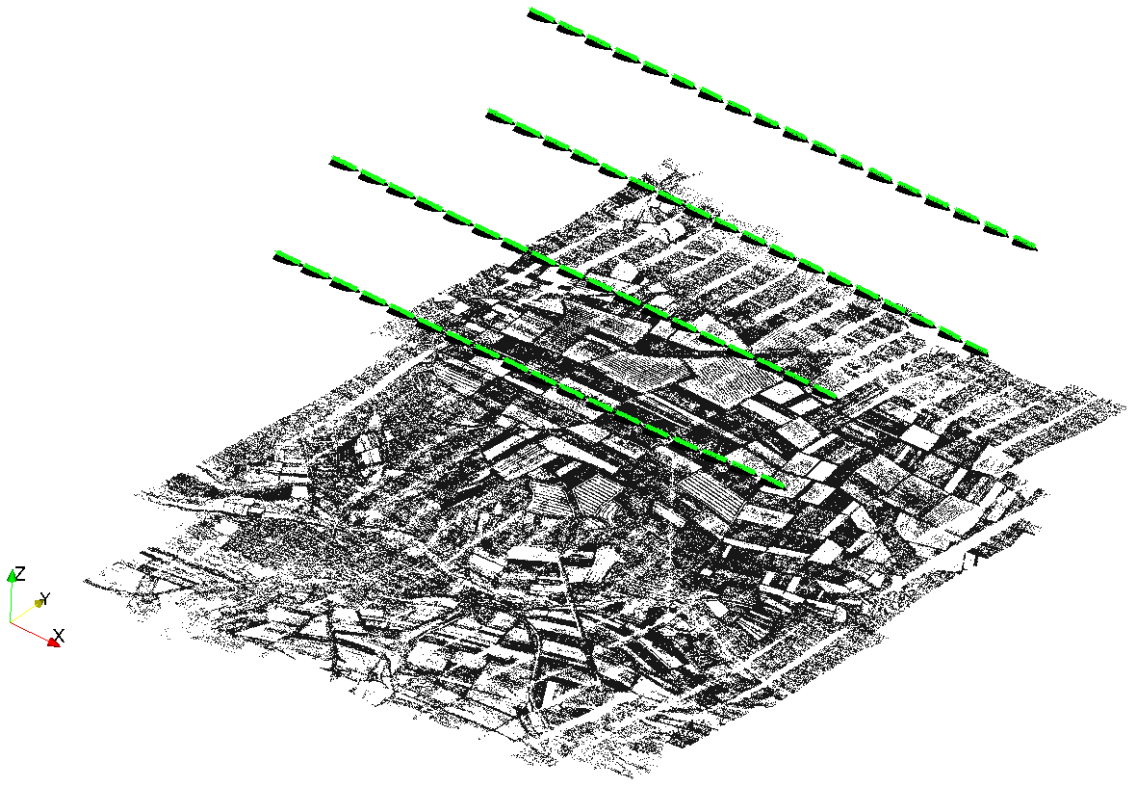


Figure 5.21: Reconstructed scene for a larger VisionMap dataset composed of 3450 images.

Chapter 6

Summary and Conclusion

Contents

In the following the main findings gained during the work on this thesis are summarized. Afterwards the contributions are outlined. Finally suggestions for the directions of future work are given.

6.1 Conclusions

Within this thesis a state of the art `AT` pipeline based on `SIFT` features is presented. It consists of the steps configuration, feature extraction, view selection, guided matching, 3D structure computation and bundle adjustment. The configuration utility assists the user in preparing the input data for the processing within the subsequent stages. Additionally the operator is able to specify `CPs` and `GCPs`, perform plausibility checks and visualize the configuration. Within the feature extraction stage only `SIFT` features are utilized, because they have been previously successfully applied in `AT` pipelines, for example in [2, 23, 52]. Both implemented view selection strategies are suitable for oblique aerial images. While the ground plane based view selection is faster, the `DEM` based one can also be applied for terrains with significant height variations.

Three different matching strategies have been realized and evaluated. They perform a prediction of the corresponding feature location in an image for a given feature point

in another image. Two of them match images pairwise and use a ground plane or a DEM as scene approximation. Also here the ground plane based matcher is restricted to flat terrains, because otherwise a large feature search radius would be required, leading to a poor performance. The third implemented matcher performs a incremental scene reconstruction, so the 3D structure computation stage is not necessary. This matcher uses also a DEM as scene approximation.

Due to the feature location prediction the amount of compared features is lowered and becomes independent of the image size. So these matcher are especially well suited for large scale images. A comparison with the FLANN library showed a performance gain of one magnitude for the Microsoft UltraCamX images. Note that the speed advantage depends on the chosen search radii. Another advantage of feature location prediction is the higher number of obtained features, because only the features within the search radius are considered for matching. So a confusion with similar features outside the search region is prevented.

The bundle adjustment stage is based on the Ceres Solver library [58] and allows the inclusion of CPs and GCPs. Additionally an unknown GPS shift can be determined and the calibration of lever arm offset and boresight misalignment rotation are possible. The following recommendations can be given as summary of the outcomes from the experiments chapter.

- The AT pipeline should use a DEM based view selection and a DEM based guided matching stage due to their universal applicability for different terrain types. In general also the reconstructing matcher may be a suitable matcher, but here larger primary set search radii are required. For images with high feature point densities these large search regions deteriorate the performance.

- As shown in the experiments section the usage of GCPs improves the accuracy of the scene reconstruction. When GCPs are applied, it is suggested to use also the GPS shift parameter [60].

6.2 Contributions of the Thesis

The main contributions of this thesis are the design, implementation and evaluation of a state of the art AT pipeline. Special attention has been paid on a fast processing of high resolution aerial images by exploiting as many prior information as available. Such prior knowledge includes GPS/INS measurements as well as scene approximations via DEM. The

most important libraries utilized for performing the individual tasks of the pipeline stages are listed in Table 6.1.

Library	Major Tasks
Ceres Solver [58]	Bundle adjustment
CGAL [69]	`DEM` geometry processing
FLANN [28]	kD tree implementation, base line method for feature matching
OpenCV [70]	Robust fundamental matrix and homography estimation, image undistortion
ICG `SfM` library	Epipolar verification, scene representation, ...
SiftGPU [59]	Feature extraction

Table 6.1: Overview of used libraries.

6.3 Direction for Future Work

Although the implemented configuration utility has already a guidance for the parametrization of `CPs` and `GCPs` included, the time required for adjusting them is still large. Furthermore it is not possible to achieve subpixel accurate observations. To overcome this an area based feature matching algorithm could be applied. Here the operator would specify only one observation and the others are found automatically with high precision [64].

In the experiments chapter it has been shown that the reprojection error is not radial symmetric, which is reasoned in the low amount of cross track matches. Also the average number of observations per world point is small, see Table 5.5. One reason for that behavior is the sensitivity of the `SIFT` detector/descriptor against perspective distortions. Here a more robust approach would generate more cross tracks, cf. [25, 65].

The implemented pairwise matchers use a circular shaped search region for the kD tree queries. The epipolar geometry is considered by discarding those features returned from a kD tree query, which have a distance above some threshold to the corresponding epipolar line. A more efficient way would be to directly use rectangular shaped search regions, which are orientated along the corresponding epipolar lines. To ease the query, a rectifying transformation [40] could be applied onto the extracted features, so that the epipolar lines are orientated parallel to either the image x- or y- axis. As result the rectangular search regions would be aligned with the image axes. This would especially improve the performance for images with high feature point densities. Currently the obtained two view feature correspondences are verified by the epipolar constraint. In the

future additionally three view feature matches should be identified and verified with the trifocal constraint. As result the amount of outliers would be further reduced.

Within this work only a first rough accuracy evaluation of the `AT` pipeline has been performed using imprecise `CP` and `GCP` locations. A more involved analysis with precisely surveyed `CP` and `GCP` locations as well as subpixel accurate image observations should be performed to achieve results comparable with published ones, cf. [42, 60]. Nevertheless a check with synthetic data has already been successfully carried out. A useful extension of the bundle adjustment stage would be the estimation of camera pose and world point location covariances including a visualization of the confidence spheres as given in [61]. With this information one can easily identify regions with low accuracy within the sparse reconstruction.

Appendix A

Acronyms and Symbols

List of Acronyms

AABB	Axis Aligned Bounding Box
ADC	Analog Digital Converter
AM	Active Matching
ANN	Approximate Nearest Neighbor
AT	Aerial Triangulation
BB	Branch and Bound
BoW	Bag of Words
CCD	Charge Coupled Device
CLAM	Chow Liu Active Matching
CORS	Continuously Operating Reference Station
CP	Check Point
CRS	Coordinate Reference System
DEM	Digital Elevation Model
DGPS	Differential Global Positioning System
DoF	Degrees of Freedom
DoG	Difference of Gaussian
ECEF	Earth-Centered, Earth-Fixed
EGM96	Earth Gravitational Model 1996
EGM08	Earth Gravitational Model 2008
EKF	Extended Kalman Filter
ENU	East North Up

ETRS89	European Terrestrial Reference System 1989
FBM	Feature Based Matching
FOV	Field Of View
GCP	Ground Control Point
GIS	Geographic Information System
GMM	Gaussian Mixture Model
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRS80	Geodetic Reference System 1980
GSD	Ground Sampling Distance
GTM	Graph Transformation Matching
IC	Individual Compatibility
ICNN	Individual Compatibility Nearest Neighbor
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ITRS	International Terrestrial Reference System
JC	Joint Compatibility
JCBB	Joint Compatibility Branch and Bound
JCPL	Joint Compatible Pair Linking
KF	Kalman Filter
KNN	K Nearest Neighbors
LM	Levenberg-Marquardt
LoG	Laplacian of Gaussian
LSM	Least Squares Matching
MAP	Maximum-A-Posteriori
MLE	Maximum Likelihood Estimation
MSL	Mean Sea Level
NCC	Normalized Cross Correlation
NN	Nearest Neighbor
OBB	Oriented Bounding Box
PCA	Principal Component Analysis
PDF	Probability Density Function
RANSAC	Random Sample Consensus
RMS	Root Mean Square

ROI	Region of Interest
RSOC	Restricted Spatial Order Constraints
RTK	Real Time Kinematic
RV	Random Variable
SCNN	Sequential Compatibility Nearest Neighbor
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SOC	Spatial Order Constraints
SRTM	Shuttle Radar Topography Mission
SST	Sea Surface Topography
SubAM	Subset Active Matching
SVD	Singular Value Decomposition
TIFF	Tagged Image File Format
TRF	Terrestrial Reference Frame
TRS	Terrestrial Reference System
UTM	Universal Transverse Mercator
VRS	Virtual Reference Station
WGN	White Gaussian Noise
WGS84	World Geodetic System 1984

List of Symbols

$\mathbb{E}$	Expectation operator
$\mathcal{N}(\mu, \sigma)$	Normal distribution
$\chi^2(n, \sigma)$	Chi-square distribution
∇	Nabla operator

Bibliography

[1] A. Irschara, M. Rumpler, P. Meixner, T. Pock, and H. Bischof, "Efficient and globally optimal multi view dense matching for aerial images," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, pp. 227–232, 2012.

[2] D. Nilosek, S. Sun, and C. Salvaggio, "Geo-accurate model extraction from three-dimensional image-derived point clouds," in *Proceedings of the SPIE, SPIE Defense and Security Sensing, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII, Modeling and Simulation*, 2012.

[3] F. Artés and J. Hutton, "GPS and inertial navigation," *GEOconnexion International Magazine*, pp. 52 – 53, 2005.

[4] Intergraph, *ERDAS Field Guide*. 2013. `http://geospatial.intergraph.com`.

[5] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, 2000.

[6] A. Ip, N. El-Sheimy, and M. Mostafa, "System performance analysis of INS/DGPS integrated system for mobile mapping system (MMS)," in *Proceedings of the 4th International Symposium on Mobile Mapping Technology*, 2004.

[7] A. Wiechert, M. Gruber, and M. Ponticelli, "UltraCam: The new super-large format digital aerial camera," in *Proceedings of the American Society of Photogrammetry and Remote Sensing Annual Conference*, 2011.

[8] T. Schenk, "Digital aerial triangulation," in *International Archives of Photogrammetry and Remote Sensing*, 1996.

[9] A. Jarvis, H. I. Reuter, A. Nelson, and E. Guevara, "Hole-filled SRTM for the globe version 4," 2008. `http://srtm.csi.cgiar.org`.

[10] R. Reid, "Jointly compatible pair linking for visual tracking with probabilistic priors," in *Australasian Computer Science Conference*, 2011.

[11] A. Handa, M. Chli, H. Strasdat, and A. Davison, "Scalable active matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[12] M. Chli and A. J. Davison, "Active matching," in *Proceedings of the 10th European Conference on Computer Vision: Part I*, 2008.

[13] J. Neira and J. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 890 –897, 2001.

[14] Trimble, October 2013. `http://www.trimble.com`.

[15] Intergraph, October 2013. `http://geospatial.intergraph.com/Homepage.aspx`.

[16] Y. Wang, "Principles and applications of structural image matching," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 53, pp. 154 – 165, 1998.

[17] A. W. Gruen, "Adaptive least squares correlation: A powerful image matching technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography*, vol. 14, pp. 175–187, 1985.

[18] P. Krzystek, "On the use of matching techniques for automatic aerial triangulation," in *International Archives of Photogrammetry and Remote Sensing*, 1998.

[19] J. Yao and W. K. Cham, "Robust multi-view feature matching from multiple unordered views," *Pattern Recognition*, vol. 40, pp. 3081 – 3099, 2007.

[20] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, pp. 189–210, 2008.

[21] A. Irschara, C. Hoppe, H. Bischof, and S. Kluckner, "Efficient structure from motion with weak position and orientation priors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2011.

[22] M. Abdel-Wahab, K. Wenzel, and D. Fritsch, "Efficient reconstruction of large unordered image datasets for high accuracy photogrammetric applications," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, pp. 1–6, 2012.

[23] J. R. Tsay and M. S. Lee, "SIFT for dense point cloud matching and aero triangulation," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX-B3, pp. 69–74, 2012.

[24] Y. Ming and X. Yuan, "An effective methodology of using GPS/IMU data for automatic point transformation in aerial triangulation," in *International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences*, 2008.

[25] J. Bartelsen, H. Mayer, H. Hirschmüller, A. Kuhn, and M. Michelini, "Orientation and dense reconstruction of unordered terrestrial and aerial wide baseline image sets," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-3, pp. 25–30, 2012.

[26] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[27] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[28] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISAPP International Conference on Computer Vision Theory and Applications*, 2009.

[29] Ordnance Survey, "A guide to coordinate systems in Great Britain." `http://www.ordnancesurvey.co.uk/oswebsite/gps/docs/A_Guide_to_Coordinate_Systems_in_Great_Britain.pdf`.

[30] S. Voser, "Map projections for the layman," *Map Projections for Europe*, vol. EUR 20120 EN, pp. 29–34, 2003.

[31] GeographicLib. `http://geographiclib.sourceforge.net/`.

[32] S. Drake, *Converting GPS Coordinates [ϕ, λ, h] to Navigation Coordinates (ENU)*. Technical note, DSTO Electronics and Surveillance Research Laboratory, 2002.

[33] Land Steiermark, "Geografisches Informationssystem Steiermark," May 2013. `http://www.gis.steiermark.at/`.

[34] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

[35] G. Neyer, "dD Range and segment trees," in *CGAL User and Reference Manual*, CGAL Editorial Board, 2013.

[36] G. Bergen, "Efficient collision detection of complex deformable models using AABB trees," *Journal of Graphics Tools*, vol. 2, pp. 1–13, 1998.

[37] P. Alliez, S. Tayeb, and C. Wormser, "3D Fast intersection and distance computation (AABB tree)," in *CGAL User and Reference Manual*, CGAL Editorial Board, 2013.

[38] F. Cacciola, "Triangulated surface mesh simplification," in *CGAL User and Reference Manual*, CGAL Editorial Board, 2013.

[39] E. Fogel, R. Wein, B. Zukerman, and D. Halperin, "2D Regularized Boolean set-operations," in *CGAL User and Reference Manual*, CGAL Editorial Board, 2013.

[40] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[41] NGA Community Sensor Model Working Group (CSMWG), *Frame Sensor Model Metadata Profile Supporting Precise Geopositioning*. Version 2.1.

[42] U. Mansholt and R. Ladstädter, "Geometrical analysis of Vexcel Imaging UltraCamX test flights," in *Proceedings of the XXI ISPRS Congress*, 2008.

[43] J. Heikkilä and O. Silvén, "Calibration procedure for short focal length off-the-shelf CCD cameras," in *Proceedings of the 13th International Conference on Pattern Recognition*, 1996.

[44] P. Johansson, "Improving camera calibration, applying a least squares method to control point measurement," Master's thesis, Umeå University, 2005.

[45] A. Ip, M. Mostafa, J. Hutton, and J. Barriere, "An optimally integrated direct georeferencing and flight management system for increased productivity of airborne mapping and remote sensing," in *Proceedings of the XXI ISPRS Congress*, 2008.

[46] M. Mostafa and J. Hutton, "Direct positioning and orientation systems: How do they work? What is the attainable accuracy?," in *Proceedings of the American Society of Photogrammetry and Remote Sensing Annual Conference*, 2001.

[47] J. Hutton, T. Bourke, and B. Scherzinger, "New developments of inertial navigation systems at Applanix," in *Proceedings of the Photogrammetric Week*, 2007.

[48] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of Fourth Alvey Vision Conference*, 1988.

[49] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[50] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of the British Machine Vision Conference*, 2002.

[51] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, 2006.

[52] Z. Liu, J. An, and Y. Jing, "A simple and robust feature point matching algorithm based on restricted spatial order constraints for aerial image registration," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, pp. 514 –527, 2012.

[53] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence*, vol. 78, pp. 87–119, 1995.

[54] D. Nistér and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, 2006.

[55] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, 2003.

[56] M. Rumpler, A. Irschara, and H. Bischof, "Multi-view stereo: Redundancy benefits for 3D reconstruction," in *Proceedings of the 35th Workshop of the Austrian Association for Pattern Recognition*, 2011.

[57] R. Ladstädter and M. Gruber, "Geometric aspects concerning the photogrammetric workflow of the digital aerial camera UltraCamX," in *Proceedings of the XXI ISPRS Congress*, 2008.

[58] S. Agarwal, K. Mierle, and Others, "Ceres solver." `https://code.google.com/p/ceres-solver/`.

[59] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)." `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[60] D. Fritsch and M. Cramer, "The geometrical accuracy of the VisionMap A3 camera system," tech. rep., Institute for Photogrammetry (ifp), University of Stuttgart, 2012.

[61] E. J. Kruck, "Developments and challenges in bundle triangulation," in *Proceedings of the American Society of Photogrammetry and Remote Sensing Annual Conference*, 2010.

[62] C. M. Ellum and N. El-Sheimy, "New strategies for integrating photogrammetric and GNSS data," in *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology', Vol. ISPRS Volume XXXVI*, 2006.

[63] H. Ebner and G. Strunz, "Combined point determination using digital terrain models as control information," in *International Archives of Photogrammetry and Remote Sensing*, 1988.

[64] G. Büyüksalih and Z. Li, "Practical experiences with automatic aerial triangulation using different software packages," *The Photogrammetric Record*, vol. 18, pp. 131–155, 2003.

[65] J. M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 438–469, 2009.

[66] M. Gruber, "UltraCamX, the new digital aerial camera system by Microsoft Photogrammetry," in *Proceedings of the Photogrammetric Week*, 2007.

[67] M. Pechatnikov, E. Shor, and Y. Raizman, "VisionMap A3 - super wide angle mapping system basic principles and workflow," in *Proceedings of the XXI ISPRS Congress*, 2008.

[68] M. Pechatnikov, E. Shor, and Y. Raizman, "The new VisionMap A3 airborne camera system," in *Photogrammetric Week 2009*, pp. 241–249, 2009.

[69] CGAL, "Computational Geometry Algorithms Library." `http://www.cgal.org`.

[70] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, vol. 25, pp. 120, 122–125, 2000.