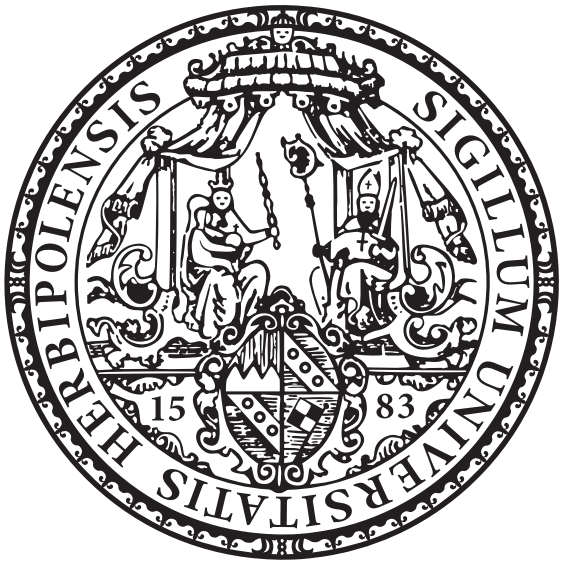
JULIUS-MAXIMILIANS-UNIVERSITÄT WÜRZBURG

IN COOPERATION WITH THE

ZENTRUM FÜR TELEMATIK E.V.



INTERNSHIP

Prototype Implementation of Image-Based Attitude and Position Determination for Absolute Visual Servoing

JOSHUA REDELBACH

August 7, 2024

Supervisor:
Prof. Dr. Andreas Nüchter

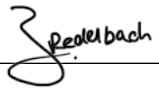
Advisor:
M.Sc. Johannes Dauner

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

ZENTRUM
FÜR
TELEMATIK

Declaration of Authorship

I, Joshua Redelbach, declare that the work of this internship was carried out in accordance with the requirements of the regulations of the Julius-Maximilians-Universität Würzburg and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others is indicated as such. Any views expressed in this work are those of the author.

Signed: 

Date: August 7, 2024

Contents

List of Figures

List of Tables

List of Abbreviations

ADCS	Attitude Determination and Control System
AVS	Absolute Visual Servoing
BRIEF	Binary Robust Independent Elementary Features
DEM	Digital Elevation Model
DLT	Direct Linear Transform
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
EOS	Earth Observation Simulator
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximate Nearest Neighbors
FOV	Field of View
IMU	Inertial Measurement Unit
KLT	Kanade-Lucas-Tomasi
LEO	Low Earth Orbit
GCP	Ground Control Point
NASA	National Aeronautics and Space Administration
OpenCV	Open Source Computer Vision Library
ORB	Oriented FAST and Rotated BRIEF
P3P	Perspective-3-Point
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RVS	Relative Visual Servoing
TOA	Top of Atmosphere
TOM	Telematics Earth Observation Mission
USGS	United States Geological Survey
UTM	Universal Transverse Mercator
VS	Visual Servoing
WGS-84	World Geodetic System-84
WRS-2	Worldwide Reference System-2
ZfT	Zentrum für Telematik e.V.

Chapter 1

Introduction

1.1 Motivation

The significance of small Low Earth Orbit (LEO) satellites in the space market has markedly increased in recent years, particularly of pico, nano, and micro satellites weighing less than 1, 10, and 100 kilograms, respectively. These small satellites offer several advantages over their larger, more complex counterparts. They are more cost-efficient to build as well as to launch into Earth's orbit, and their development time is significantly shorter, typically ranging from 1 to 2 years, compared to the lengthy development periods of larger satellites. These advancements also democratize access to space technology, enabling smaller entities like universities to participate in the field of satellite missions, which was previously dominated by large corporations. One of the primary functions of these small satellites is Earth observation, mainly using visual cameras. To obtain high-quality image data during satellite missions, precise alignment with the observation target is essential during overflight. Consequently, these missions impose stringent requirements on the Attitude Determination and Control System (ADCS). Usually, precise attitude determination is achieved through sensors such as sun sensors, magnetometers, gyroscopes, and star trackers. However, due to the limited size and power of small satellites, alternative approaches to bulky and power consuming sensors are necessary. A practical approach for Earth observation missions is to utilize image data from the payload camera for this purpose. Advances in imaging technology have enabled even these small onboard cameras to capture high-resolution images. Moreover, current microprocessor technology allows for the computationally intensive analysis of these images in real-time onboard the satellite. This approach can save space and mass compared to traditional sensors.
These developments have led to numerous studies dealing with the method of determining a satellite's attitude and position based on image data from an onboard camera. Koizumi et al. [27] utilize a method that first detects the Earth's edge in the captured image to determine the 2-axis attitude (nadir direction). Then, they achieve 3-axis attitude determination by matching land patterns using deep learning techniques. Dagvasumberel and Asami [9] employ feature detection and matching algorithms to identify identical patterns in successive images and subsequently estimate the relative attitude. Then, they propagate the obtained relative attitude using gyroscope measurements within an unscented Kalman filter. In the study of Kouyama et al. [28], the

attitude of the satellite is derived from its onboard images and known satellite position by matching detected land features with well-registered base-map images. Campagna et al. [7] use a deep learning feature detector to detect coastlines and match these with an onboard database consisting of descriptor vectors of coastlines combined with their on ground position. Based on these matches, they compute the distance from the camera to the covered scene as well as the latitude and longitude of the image centroid. However, all of these approaches suffer from individual drawbacks, e.g. due to questionable real-time applicability or large amount of required memory space.
Besides for determining the attitude of a satellite orbiting the Earth, image-based attitude and position determination gain in importance for entry, descent and landing systems for lunar or Mars missions as well. The most popular example is the lander vision system developed for the Mars 2020 mission. It enabled a precise landing of the Perseverance rover on the surface of Mars within 5m of the targeted landing point by first, matching detected structures with a surface map to obtain absolute attitude and position estimation and, second, by matching detected structures in a sequence of descent images for estimating the linear velocity of the lander [32]. Using this approach, they were able estimate the attitude with an error less than 0.15 deg about each axis and the linear velocity with an error of less than 0.25 $\frac{m}{s}$ [23]. Even though the approach of the Mars 2020 mission cannot directly be adapted for satellites orbiting the Earth, as the method is optimized for a known, limited surface map of the landing site, it gets clear that the attitude and position of a spacecraft can be estimated accurately using an image-based approach. The attitude derived from onboard image data can be used to calculate the control input for attitude-controlling actuators, such as reaction wheels. These vision-based control concepts are also known as Visual Servoing (VS). The concept of VS has been adapted to the application of satellites, i.e. by Hladký [21], Klančar et al. [25], Dauner et al. [10], and Elsner [12]. The studies of Dauner et al. [10] and Elsner [12] were conducted within the Telematics Earth Observation Mission (TOM) project, a satellite formation mission by the Zentrum für Telematik e.V. (ZfT) aimed at observing ash clouds during volcanic eruptions. To meet the high demands on the ADCS, a Relative Visual Servoing (RVS) approach was developed, enabling three satellites to jointly track the same target area. This approach involves detecting highly distinctive features in the images from each satellite and calculating the control input based on the movement of these features between consecutive images. For the satellites to ensure the coordinate tracking of a target area, the detected features and their positions in the images are shared among the satellites. This process allows the satellites to align relative to the target object. However, RVS only works successfully if the images from the different satellites initially overlap to some extent. To ensure this overlap, the satellites must first be aligned to within approximately 2 deg using conventional attitude sensors such as sun sensors or magnetometers. The concept of RVS is now enhanced to Absolute Visual Servoing (AVS), which should enable the precise pointing of a single satellite towards a specific location on Earth's surface by determining the absolute attitude and position using an image-based approach. The basic idea is to detect features onboard in incoming image frames, match these features with Ground Control Point (GCP)s stored in a database. Based on the position of the GCPs in the image and

on ground, the attitude and position can be estimated with respect to the Earth Centered Earth Fixed (ECEF) frame (more details in section 2.2). In the work of Redelbach [41], the process of creating the GCP database as well as of matching detected features onboard have already been investigated with respect to the algorithms used for feature detection, description, matching and validation.

1.2 Scope of Work

Based on the work of [41], within this internship a pipeline for the entire process of onboard feature matching and the subsequent image-based attitude and position determination should be implemented and tested. Besides adding the procedure of estimating the attitude and position based on the matched GCPs, the process of database creation and of onboard feature matching needs to be adapted as well. Thus, this internship is intended to show the applicability of the approach of AVS and to provide a possible configuration of the onboard pipeline and of the creation of the database as a basis for the design and concrete implementation of the ADCS in future missions.

1.3 Structure of Work

In order to achieve this objective, a research on image-based attitude and position estimation algorithms is done. The basic principle of how this can be performed together with the required theoretical background knowledge is presented in chapter 2. In chapter 3, the design of AVS is described in detail. Chapter 4 describes the concrete implementation of the creation of the database and the onboard pipeline. Here, the used image data that is required for the database as well as for testing the approach is given as well. The pipeline is tested, evaluated and improved in chapter 5. An outlook on how the approach of AVS could be extended and further improved as well as what aspects need to be investigated in the future is given in chapter 6. Finally, the work is concluded in chapter 7 by a summary of the contributions.

Chapter 2

Theoretical Background

The following chapter provides essential background knowledge to comprehensively understand the following five key aspects: first, what is meant by attitude and position determination; second, the motivation and conceptual idea of AVS; third, the principles of feature detection, description, matching, and validation; fourth, the significance of GCPs in the context of AVS; and fifth, the working principle of image-based pose estimation.

2.1 Attitude and Position Determination

In order to determine the attitude $(\theta, \phi, \psi)^{I \to B}$ and position $(\vec{x}, \vec{y}, \vec{z})^{I \to B}$ of the satellite and, thus, how the body frame is shifted and rotated with respect to the Earth Centered Inertial (ECI) frame, the following three steps need to be performed:

1. Determine position and attitude of the onboard camera with respect to the ECEF frame ($E \to C$) based on the matched features.
2. Determine position and attitude of the satellite with respect to the ECEF frame ($E \to B$) based on the position and attitude of the camera.
3. Transform the position and attitude of the satellite from the ECEF frame to the ECI frame ($I \to B$).

The different coordinate frames are illustrated in fig. 2.1. As will be shown in section 2.5, the attitude and position of the onboard camera relative to the ECEF frame $(x, y, z, \theta, \phi, \psi)^{E \to C}$ can be calculated based on the matched image and Earth position of at least four GCPs and the intrinsic camera parameters which can be obtained after calibration. As the relative attitude and position of the built in camera with respect to the body frame is known, the position and attitude of the satellite with respect to the ECEF frame $(x, y, z, \theta, \phi, \psi)^{E \to B}$ can be easily computed. The last step is to transform these parameters to the ECI frame in order to obtain the desired attitude and position $(x, y, z, \theta, \phi, \psi)^{I \to B}$, which is a routine task for the ADCS and thus is not described in detail within this work.

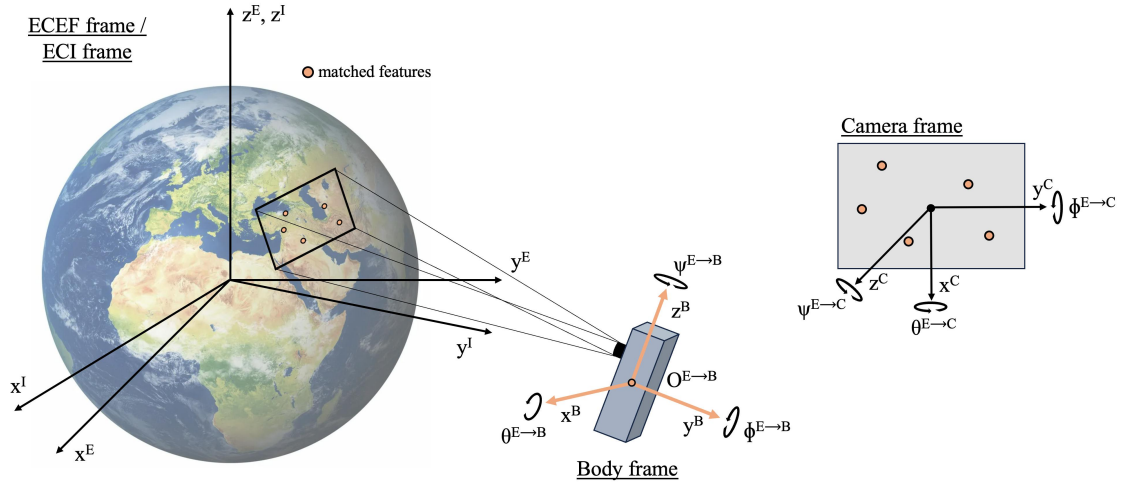


FIGURE 2.1: Illustration of the different reference frames used to determine the attitude and position of the satellite based on the onboard matched features. Vector arrows are omitted. Credits for image of Earth: SEO-Analyse [44]

2.2 Absolute Visual Servoing

VS is a method used to control the motion of a robot based on image data. Originating from the field of robotics, this technique was first introduced in the early 1980s. The primary goal of VS is to minimize the difference between the current and desired state, as determined through image measurements, in order to perform positioning or tracking tasks [8, 22, 5]. To adapt VS for controlling the attitude of a satellite, the current attitude of the spacecraft must first be derived from image data provided by an onboard camera. The control input for actuators, such as reaction wheels, is then calculated based on the difference between the current and desired attitudes. As described in section 1.1, for the satellite formation mission of the ZfT, called TOM, a RVS approach was developed to fulfill the high demands on the ADCS and thus to enable the joint tracking of the same target area by three satellites [10, 12].

This method is now refined from a relative to an absolute approach to extend the use of image-based attitude control algorithms for further Earth observation missions. This refinement allows a single satellite to precisely point to a specific location on Earth by determining its absolute attitude and position from onboard images. The entire work flow during AVS is illustrated in fig. 2.2. In AVS, new image frames are analyzed to detect features, which are then compared and matched with pre-known features. These pre-known features, called GCPs, are unique structures on Earth's surface identifiable in satellite imagery and stored in an onboard database with their absolute positions. These GCPs, as described in detail later, are obtained by detecting and describing features in external satellite images. After filtering only suitable features, the feature descriptors of the GCPs are stored in a database, combined with their position on Earth, that can be obtained using the provided meta data of the input satellite images. These positions are referenced using the ECEF coordinate system. After matching and validating the detected features of the onboard image with the GCP database, a list of $2D$ image and $3D$ ECEF point correspondences result. Based on this list, representing the

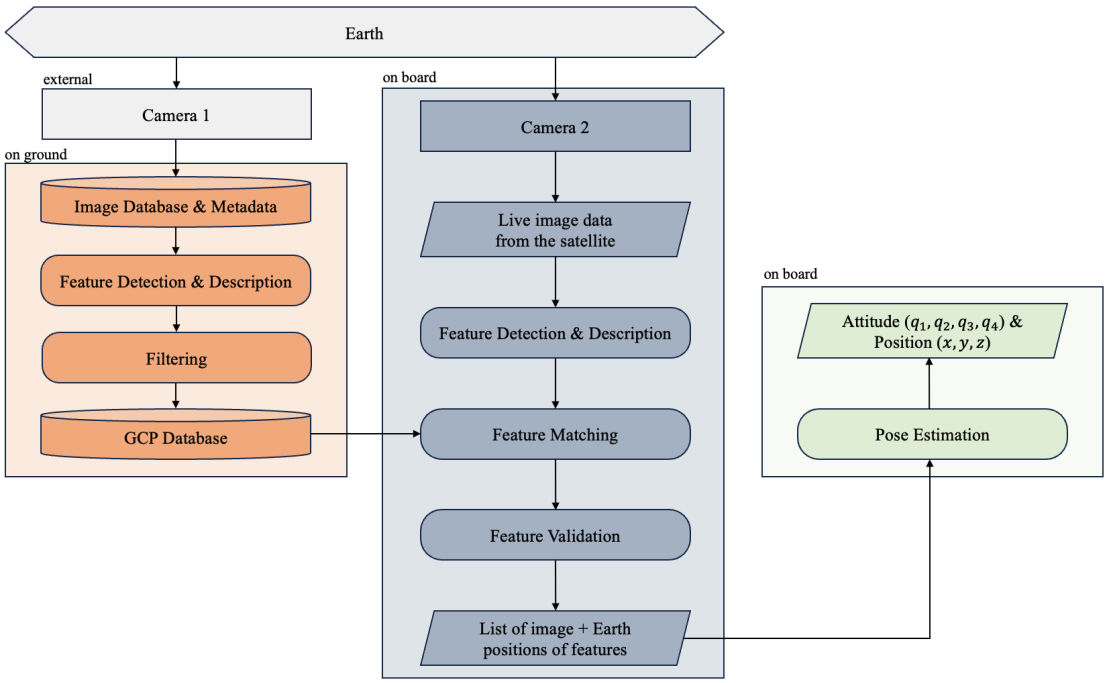


FIGURE 2.2: Block diagram of the processes performed during AVS.

appearance of on ground GCPs in the current image, the absolute attitude and position of the camera relative to the ECEF coordinate system is calculated. As stated out in the previous section, this attitude and position can then be easily transformed in order to obtain the attitude and position of the satellite (body frame) with respect to the ECI frame. This image-based attitude and position determination is referred to as pose estimation. This data can then be used to compute the control inputs necessary for precise target pointing. Figure 2.3 illustrates the ideal scenario where the image positions of the detected GCPs are matched with their absolute positions on Earth.

The main objective of AVS is to ensure reliable position and attitude determination throughout the satellite's orbit by detecting enough features with the onboard camera and matching these to those in the database. Several conditions must be met for this procedure to be successful. Firstly, the GCP features in the database must be evenly distributed globally to ensure that the onboard camera is always able to detect a minimum number of features within its Field of View (FOV). Secondly, the matched features must be free of false matches to ensure accurate position and attitude determination. Reliable feature matching requires that GCP features have a unique appearance in satellite images and a distinctive signature, enabling correct detection and matching onboard even if a different camera is integrated onboard than the one used to create the database.

In the thesis of Redelbach [41], the processes of creating the database and of matching the onboard detected image features with those stored in the database have already been analyzed especially regarding the choice of the algorithms used for detecting, describing and matching image features. Therefore, this internship is intended to build upon this work and to implement an onboard pipeline that combines the feature matching process and the attitude and position determination based on the matched GCPs.

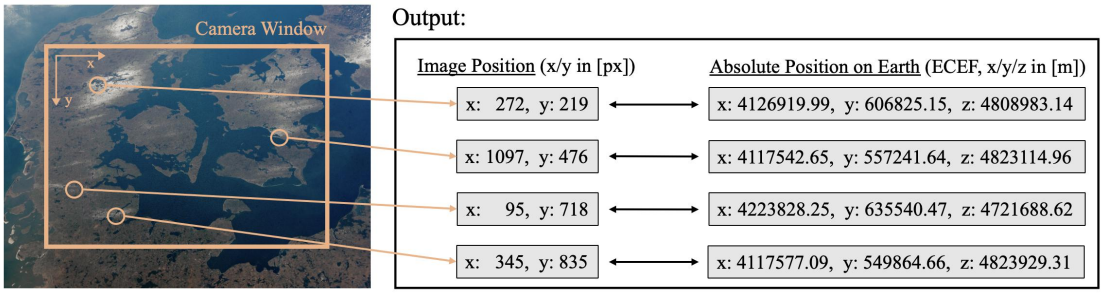


FIGURE 2.3: Illustration of a scenario in which the image position (x/y-coordinates in [px]) of detected features are successfully matched with the absolute position on Earth (x/y/z-coordinates in ECEF frame in [m]) of known features. Satellite image credit: NASA [34]

2.3 Image Analysis

Within the proposed work, the operating principles of absolute visual servoing are based on the detection and matching of features in satellite imagery. Therefore, the following section shortly introduces the concept of features, and how they can be detected, described and matched as well as how these matched features can be validated.

2.3.1 Features

Features are distinct points in an image that stand out from their surrounding neighborhood and can be repeatedly detected across various images of the same scene. For reliable detection, these features must not only be unique but also remain unchanged under geometric transformations (such as translation, scaling, and rotation), photometric variations (like changes in brightness or exposure), and noise. The primary goal of the concept of features is to describe the image by a set of feature vectors, which captures the local structure of the image. Each feature vector, also known as feature descriptor, represents a specific point of interest in the image, called keypoint [20, 45].

2.3.2 Feature Detection

In order to detect and locate these points of interest in an image, feature detection algorithms search for areas with significant structural information and specific shape, such as edges, corners or blobs. This is done by defining a certain salience measure, such as pixel intensity gradient, and looking for local extrema across the image pixels [2]. One of the earliest and most widely used feature detection algorithms is the Harris Corner Detector, published in 1988 [19]. Since its introduction, many other feature detection algorithms have been developed, each offering various advantages depending on the specific application, e.g. SURF [4], FAST [42], ORB [43] or BRISK [30].

2.3.3 Feature Description

To compare features detected in different images of the same scene, each feature needs a unique, compressed signature. Feature description algorithms achieve this by analyzing the area surrounding a feature and encoding relevant information into a descriptor

vector, which acts as a numerical fingerprint. Just like the features, these descriptor vectors must be robust against noise, viewpoint changes, and variations in photometric conditions. Additionally, they need to be distinctive while at the same time of limited length to ensure that the description and matching processes remain real-time applicable [2]. Among others, the current state-of-the-art feature description algorithms include SURF [4], BRIEF [6], ORB [43], BRISK [30] and FREAK [1].

2.3.4 Feature Matching

The primary goal of feature matching is to identify specific parts of an image by comparing two sets of detected feature descriptors. After matching, only features present in both sets should remain. Typically, this involves matching two images, with the second set of feature descriptors belonging to a different image. In the case of AVS, the second set consists of feature descriptors from GCPs stored in a database, though the concept remains unchanged. The matching is achieved by calculating the Euclidean or Hamming distance between all feature descriptors in both sets. Each vector from the first set is matched with its nearest neighbor in the second set based on descriptor distance [2].

2.3.5 Feature Validation

Since not every feature in the first set will necessarily appear in the second set, mismatches with similar features are unavoidable. To ensure reliable feature matching, the final step is to validate the matches. This involves identifying and discarding mismatches, which can be achieved through various outlier removal algorithms. In related work, this validation process is often included as a sub-task under feature matching. However, due to its importance in AVS, this work treats matching and validation as separate processes. To discard false matches and improve reliability, the plausibility of all matches is assessed based on different criteria, such as selecting matches with a descriptor distance below a certain threshold or using the RANSAC algorithm [13] to check the relative pixel positions of the matched features.

2.3.6 Algorithms

As already mentioned, for each part during the image analysis process, there exist many different algorithms, which all have different advantages depending on the specific application they are used in. Thus, in the work of Redelbach [41] different combinations of algorithms for the application of AVS are tested with respect to multiple aspects: number of matches, reliability, invariance to rotation, memory space and runtime.

ORB

When all aspects are considered, the combination using the Oriented FAST and Rotated BRIEF (ORB) algorithm [43] for detecting and describing features achieved the best results during the tests performed by Redelbach [41]. Thus, ORB is used for all feature detection and description tasks that are performed within this work. The ORB

algorithm is based on the Features from Accelerated Segment Test (FAST) keypoint detector [42] and the Binary Robust Independent Elementary Features (BRIEF) descriptor [6]. Both of these algorithms are very attractive due to their computational speed. However, FAST as well as BRIEF contain several limitations, e.g. large response along edges during detection or no invariance to rotation of the descriptor, which are addressed and improved in the ORB algorithm. Details on the implementation of the algorithm can be read as a summary in the work of Redelbach [41] or in detail in the original paper of Rublee et al. [43].

RANSAC

The most common outlier removal algorithm for matched features is Random Sample Consensus (RANSAC) [13]. The goal of RANSAC is to identify a model that describes the correlation between two sets of data points, in this case, image features. Once the model is established, any data points that do not fit the model are identified as outliers. Unlike many other sampling techniques that use as much data as possible to derive the model, RANSAC starts with the smallest possible subset to determine a model and progressively expands it with data points that also fit the model. The general procedure can be summarized as follows [11]:

1. Select a minimum subset of data required to determine the model parameters.
2. Determine the parameters of the model for the selected data.
3. Evaluate how many datapoints from all data fit the model depending on a certain threshold.
4. If the current model contains more inliers than the existing best model, update it.
5. If the current model contains all datapoints, the algorithm stops. Otherwise, repeat step 1 to 4.

2.4 Ground Control Points

Traditionally, GCPs are identifiable points on the Earth's surface with precisely known geographic locations, used in remote sensing and geospatial mapping to align aerial or satellite imagery with real-world coordinates. These GCPs are often physical markers or targets visible in both imagery and on the ground, such as specially designed markers, painted crosses, or natural features like road intersections. They are stable and permanent, allowing reliable identification and measurement over time, with their locations determined using high-precision surveying techniques.
In the context of AVS, GCPs are similarly defined as identifiable points on the Earth's surface with known geographic locations. However, unlike traditional GCPs, these points are selected by feature detection algorithms rather than manually. These algorithms analyze satellite imagery to extract unique locations, which are not easily interpretable or found on the ground. In order to be able to determine their Earth's position anyway, the feature detectors are applied to satellite imagery with given metadata that

provide useful information about the image such as the precise geographic location of the corners. After detection, the pixel positions of the feature keypoints combined with this metadata enable the determination of their exact ground positions. Afterwards, these keypoints receive unique signatures from feature description algorithms. Each GCP in AVS then consists of a unique description mapped to its absolute ground position in the ECEF coordinate system and is stored in a database. This allows GCPs to be easily found in other satellite images by detecting features and matching them with the GCP database.

2.5 Image-based Pose Estimation

In this section, all the necessary background knowledge is presented to understand how the attitude and position of the camera can be determined based on correspondences of $2D$ image points and $3D$ GCPs. All presented concepts are taken from the book of Förstner and Wrobel [14].

2.5.1 Camera Model

The field of photogrammetry is about measuring the real world based on images. To to so, a model of the camera is required. The most common one is the pinhole camera model which can be imagined as a black box with an infinitesimal small hole, called projection center. The back wall of the box represents the image plane (see fig. 2.4). The distance between the projection center an the image plane is called focal length or camera constant. Through the small hole, light rays enter the box and intersect with the image plane. In this way, the $3D$ world is projected onto the $2D$ camera sensor plane. This projection is neither angle- nor length-preserving but it is straight-line-preserving. Using this camera model and the concept of homogeneous coordinates the mapping of a real world $3D$ point $(X, Y, Z)^W$ onto the $2D$ camera sensor plane $(x, y)^S$ can be expressed as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}^S = \mathbf{P}^{W\to S} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^{W\to S} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W \tag{2.1}$$

$$\Rightarrow x^S = \frac{u^S}{w^S} \text{ and } y^S = \frac{v^S}{w^S}$$

The projection of the world coordinate system into the sensor coordinate system is divided into several transformation steps. This is done by introducing two additional frames, namely the camera and the image coordinate systems. All described frames are illustrated in fig. 2.5. During the projection, the points in the world frame are transformed to the camera frame, then to the image frame and lastly to the sensor frame.
The camera coordinate frame is centered at the projection center and its z^C-axis is

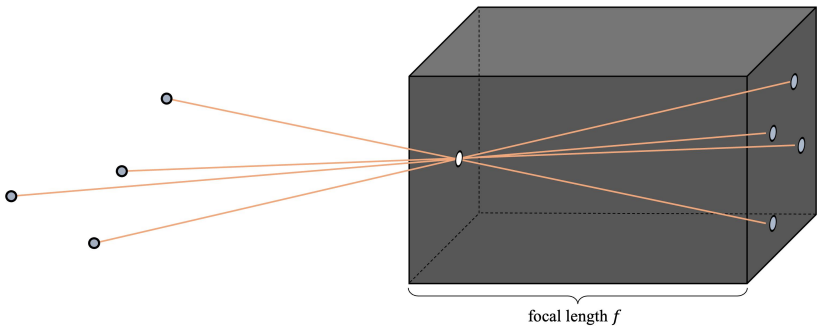


FIGURE 2.4: Illustration of the pinhole camera model.

aligned with the normal of the image and the sensor plane respectively. The transformation from the world to the camera frame is a rigid body transformation that can be performed by a rotation and a translation leading to 6 parameters: $(X_0, Y_0, Z_0, \theta, \phi, \psi)^{W \to C}$.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^C = \begin{bmatrix} \mathbf{R}(\theta,\phi,\psi)^{W\to C} & -\mathbf{R}(\theta,\phi,\psi)^{W\to C} \cdot T_0^C \\ 0^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W \tag{2.2}$$

$$\text{with } T_0^C = (X_0, Y_0, Z_0)^{W\to C}$$

These parameters are called the extrinsic parameters and they represent the attitude and position of the camera with respect to the world frame. In photogrammetry, this is referred to as the pose of the camera. (In the following, for simplification, the rotation matrix $\mathbf{R}(\theta,\phi,\psi)^{W\to C}$ is denoted as $\mathbf{R}$).
The image coordinate system is a $2D$ reference frame that lies, as the name suggests, in the image plane. The distance to the camera frame is the focal length f. When transforming from the camera to the image frame the projection from $3D$ to $2D$ takes place:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}^P = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{R} & -\mathbf{R} \cdot T_0^C \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W \tag{2.3}$$

$$\Rightarrow x^P = \frac{u^P}{w^P} \text{ and } y^P = \frac{v^P}{w^P}$$

The sensor frame is, like the image frame, a $2D$ coordinate system. But depending on the used camera it can be shifted (x_H, y_H), scaled (m) and sheered (s). All of this can be compensated by linear transformations. Additionally, there can occur non-linear effects, e.g. due to an imperfect lens. These non-linear errors are tried to be compensated by a location dependent distortion shift $\Delta x(\vec{x}, \vec{q})$, $\Delta y(\vec{x}, \vec{q})$, where $\vec{q}$ are the distortion coefficients of the used distortion model. There exist several different models to compute

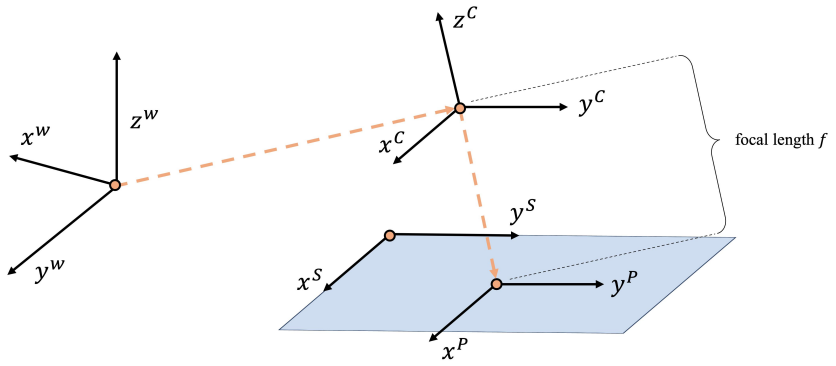


FIGURE 2.5: Illustration of the different coordinate systems used to project $3D$ real world points into $2D$ sensor plane.

the distortion shift, e.g. the Barrell-distortion model.

$$\begin{aligned}
\begin{bmatrix} u \\ v \\ w \end{bmatrix}^S &= \begin{bmatrix} 1 & s & x_H + \Delta x(\vec{x}, \vec{q}) \\ 0 & 1+m & y_H + \Delta y(\vec{x}, \vec{q}) \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{R} & -\mathbf{R} \cdot T_0^C \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W \\
&= \underbrace{\begin{bmatrix} f & f \cdot s & x_H + \Delta x(\vec{x}, \vec{q}) \\ 0 & f \cdot (1+m) & y_H + \Delta y(\vec{x}, \vec{q}) \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \cdot \begin{bmatrix} \mathbf{R} & -\mathbf{R} \cdot T_0^C \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^W
\end{aligned} \tag{2.4}$$

All parameters except the six extrinsic parameters of the rigid body transformations are referred to as intrinsic parameters. Ignoring the compensation of the non-linear errors, there are five intrinsic parameters, so in total, eleven parameters are required to model the projection of $3D$ points into to $2D$ sensor frame using the pinhole camera model. Depending on the distortion model, the list of intrinsic parameters can be extended accordingly. The matrix $\mathbf{K}$ summarizes all transformations that depend on these intrinsic parameters and is called the calibration matrix.

2.5.2 Camera Calibration

For many tasks in photogrammetry, it is mandatory to know the intrinsic camera parameters. The procedure of obtaining these parameters is called camera calibration. There are many different ways of calibrating a camera and it counts as a routine task in photogrammetric computer vision applications. Regarding AVS, this step is done in advance of the mission using the onboard camera and is thus not covered in this work. For details on the topic, one can refer to [14]. In the following, it is assumed, that the onboard camera is calibrated and the intrinsic parameters are known.

2.5.3 Pose Estimation

In computer vision applications, one fundamental task is to determine the position and orientation (the pose) of the camera with respect to a specific world coordinate system. If the intrinsic parameters of a camera are known, the pose, represented by the six extrinsic parameters, can be estimated based on correspondences between $3D$ points, that are specified with respect to a specific world frame, and $2D$ image points, that are specified with respect to the sensor frame. This task is referred to as the Perspective-n-Point (PnP) problem and can be solved by the method of spatial resection.

Perspective 3-Point Problem

One of the first approaches for a direct solution was presented by Grunert [18] in 1841. For this approach, and also in general, the minimum number of three point correspondences is required. This is why the task of estimating the pose of the camera is oftentimes referred to as the Perspective-3-Point (P3P) problem. Let $\mathcal{X}_i^W$ and x_i^S with $i = 1, 2, 3$ be the observed points in the world and sensor frame respectively. Then, the normalized ray direction from the projection center $\mathcal{Z}^W$ to the observed points $\mathcal{X}_i$ in the camera frame can be computed:

$$\mathrm{r}_i^C = \mathrm{normalize}(\mathbf{K}^{-1}\mathrm{x}_i^S) \tag{2.5}$$

The coordinates of the observed points in the camera frame can then be described using the distance $d_i = \left|\mathcal{X}_i^W - \mathcal{Z}^W\right|$ from the projection center:

$$d_i \cdot \mathrm{r}_i^C = \mathcal{X}_i^C = \mathbf{R} \cdot (\mathcal{X}_i^W - \mathcal{Z}^W) \tag{2.6}$$

The described scenario is illustrated in fig. 2.6. The first step is to compute the distances d_i. This is done by by determining the three sides a, b and c as well as the three angles α, β and γ of the tetrahedron using the law of cosines:

$$\begin{aligned} a^2 &= d_2^2 + d_3^2 - 2d_2d_3\cos\alpha \\ b^2 &= d_3^2 + d_1^2 - 2d_3d_1\cos\beta \\ c^2 &= d_1^2 + d_2^2 - 2d_1d_2\cos\gamma \end{aligned} \tag{2.7}$$

By substituting $u = \frac{d_2}{d_1}$ and $v = \frac{d_3}{d_1}$ into the equations 2.7, and solving for d_1, we obtain:

$$d_1^2 = \frac{a^2}{u^2 + v^2 - 2uv\cos\alpha} \tag{2.8}$$

$$= \frac{b^2}{1 + v^2 - 2v\cos\beta} \tag{2.9}$$

$$= \frac{c^2}{u^2 + 1 - 2u\cos\gamma} \tag{2.10}$$

Rearranging again results in a polynomial of degree four in which v needs to vanish:

$$A_4v^4 + A_3v^3 + A_2v^2 + A_1v + A_0 = 0 \tag{2.11}$$

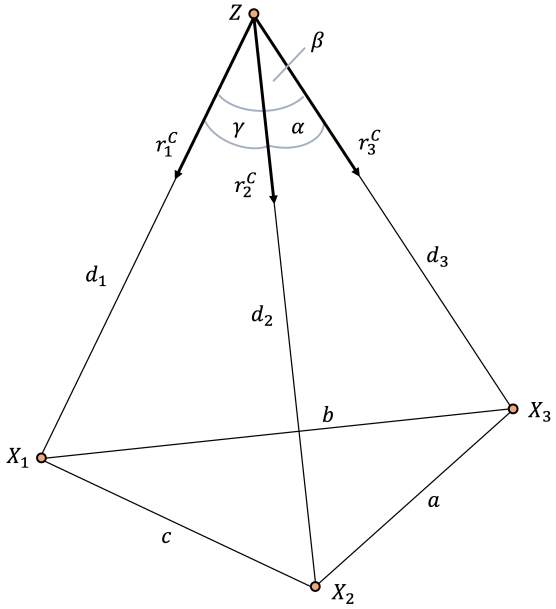


FIGURE 2.6: Spatial resection with three observed points $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$.

The coefficients are given in appendix A.1. For details on the derivation one cann refer to [14]. After solving the polynomial for v, the distances d_1,d_2 and d_3 can be computed using equations 2.8-2.10. But, as the polynomial is of degree four, there can be up to four solutions for the distances d_1, d_2 and d_3. To eliminate the ambiguity, a fourth point needs to be observed to identify the correct solution. Thus, the term P3P problem is misleading as actually four points are required.
After the distances are computed, the $3D$ coordinates of the observed points with respect to the camera frame can be computed using equation 2.6. Then, three $3D$-$3D$ point correspondences are obtained. Based on these, the rigid body transformation (translation and rotation) from the world to the camera frame can be computed. How this can be done, can be read in [14].
Over time, many other approaches for solving the P3P problem have been developed. These new methods still have ambiguities when using three observed points, but are faster to compute, numerically more stable or partially less complex. A few recent approaches are the ones of Gao et al. [16], Kneip, Scaramuzza, and Siegwart [26], Banno [3], Persson and Nordberg [39] and Nakano [33].

Perspective N-Point Problem

The problem of the approach that only considers three (or four) observed points is that no measurement noise is considered. As in practical applications, the coordinates of the observed points in the sensor and in the world frame always include measurement errors, the solution of the P3P problem is not statistically optimal. This can be improved by taking more than three observed points into account which leads to an overdetermined system refered to as the PnP problem. This can be solved using an iterative least-squared approach. The basic procedure is as follows:

1. **Initialization**: At the beginning, an initial estimate of the camera pose is required, which can be obtained using methods like Direct Linear Transform (DLT).
2. **Projection and Residual Calculation**: For each iteration, the $3D$ points are projected onto the image plane using the current estimate of the camera pose. Then, the residuals are calculated, which are the differences between the projected $2D$ points and the actual observed $2D$ points.
3. **Jacobian Computation**: The Jacobian matrix is computed, which represents the partial derivatives of the residuals with respect to the camera pose parameters. This matrix helps in understanding how changes in the pose affect the projection of the points.
4. **Pose Update**: Afterwards, the camera pose is updated using an optimization algorithm. Commonly used methods include the Gauss-Newton algorithm or the Levenberg-Marquardt algorithm. These methods iteratively minimize the sum of squared residuals by updating the pose parameters.
5. **Convergence Check**: Then, it is checked for convergence by evaluating whether the changes in the pose parameters or the residuals fall below a certain threshold. If convergence is not achieved, repeat the steps starting from 2. Once convergence is achieved, the final camera pose estimate is taken as the solution to the PnP problem.

The advantages of this methods are that it is robust and can handle noisy data as well to find an optimal solution. However, the quality of the convergence depends on the quality of the initial pose estimate. Furthermore, this method can be computationally intense when dealing with a large number of points. As the focus of this work is on the practical implementation, for more details on the iterative approach, one can refer to Förstner and Wrobel [14]

2.5.4 Theoretical Precision

In order to get an rough impression of how accurate an image-based pose estimation can be, the theoretical precision of the pose estimate based on the direct solution of the spatial resection for the P3P problem is derived for a standard case based on the approach introduced in [14].
Some simplifications are made for the analyzed scenario which is illustrated in fig. **??**. First, the four $3D$ observed points on ground $\mathcal{X}_i$ with $i = 1, 2, 3, 4$ further called GCPs, lie on a quadratic plane (blue) and represent its corners. It is assumed that their coordinates are given with respect to the world frame without any errors. Additionally, the intrinsic parameters are given without any errors as well. The camera is pointing with nadir view to the center of the plane and the projection center O has the coordinates $(X_0, Y_0, Z_0)^W$ with respect to the world frame. The appearance of the on ground plane spanned by the GCPs is colored in gray in the image plane (orange) and has the width and height of $2d$. The angle γ describes the FOV with respect to the observed scene, not the total horizontal FOV. So if the GCPs lie closer to each other, γ is smaller. The

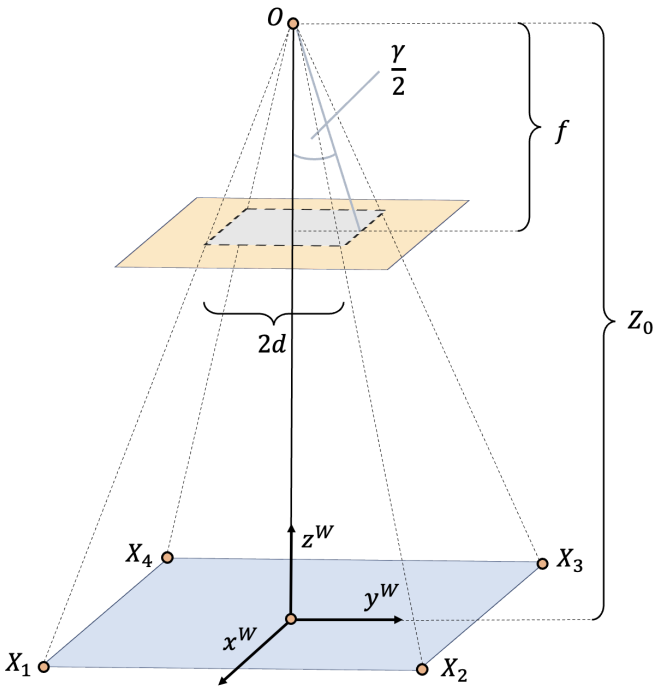


FIGURE 2.7: Illustration of the used scenario for assessing the accuracy of image-based pose estimation.

coordinates of the observed points with respect to the sensor frame are assumed to be measured with a standard deviation of $\sigma_0 = \sigma_{x^S} = \sigma_{y^S}$. This leads to the following expressions for the standard deviations of the extrinsic parameters:

$$\sigma_{X_0} = \sigma_{Y_0} = \frac{\sqrt{2}}{4}\frac{Z_0}{f}\sqrt{1+\frac{1}{\sin^4\frac{\gamma}{2}}}\,\sigma_0 \tag{2.12}$$

$$\sigma_{Z_0} = \frac{Z_0}{4d}\,\sigma_0 \tag{2.13}$$

$$\sigma_\phi = \sigma_\theta = \frac{\sqrt{2}}{4}\frac{f}{d^2}\,\sigma_0 \tag{2.14}$$

$$\sigma_\psi = \frac{1}{4d}\,\sigma_0 \tag{2.15}$$

It can be seen that by the formulas for the described scenario, that the accuracy of the extrinsic parameters mainly depend on the accuracy of the measurement of the GCPs in the image (σ_0) and of the distance between the GCPs, encoded in d and γ. Especially, the measurement of the GCPs has a strong impact as the influences the extrinsic parameters in a linear way. For the application of AVS, the measurement of the GCPs in the image is done by feature detection algorithms. Thus, it is difficult to make a statement about an exact value for σ_0. In the following, a subpixel accuracy of $\sigma_0 = 0.1$ is assumed. For the parameters of the onboard camera integrated in the satellites for TOM (image width of $w = 2448$ pixel, horizontal FOV of $hFOV = 4.8$ deg and resulting focal length $f = \frac{1}{2}\cdot\frac{w}{\tan\frac{hFOV}{2}}$), the standard deviations of the extrinsic parameters depending on the angle γ are plotted in fig. 2.8. Qualitatively, the plots show that a wider distribution of

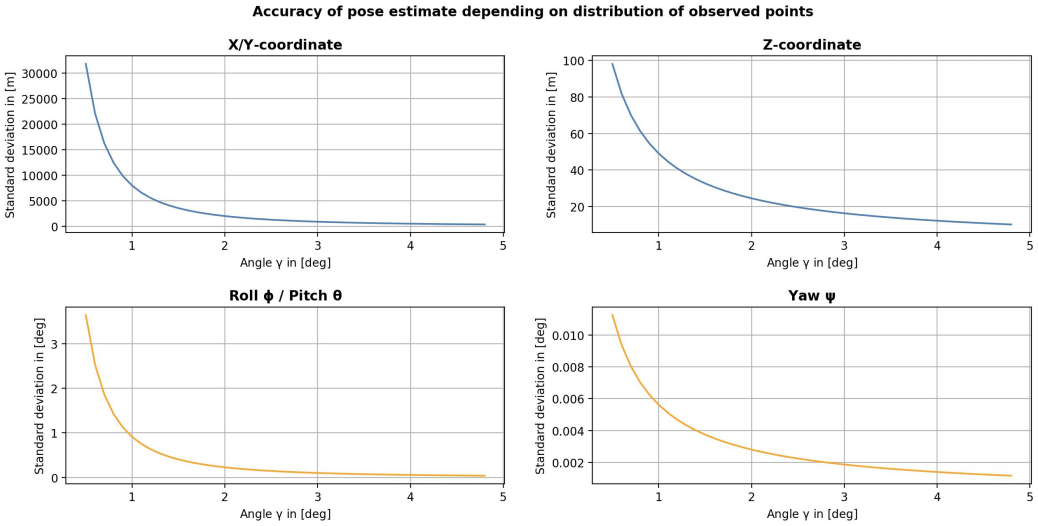


FIGURE 2.8: Theoretical precision of image-based pose estimation using the parameters of the onboard camera of TOM for the position (X, Y, Z) and the attitude (ϕ, θ, ψ) of the camera.

the GCPs leads to more accurate pose estimate regarding all six parameters. Quantitatively, it can be seen that, when a realistic angle γ of about $3 - 4$ deg is assumed, the position in lateral direction and in nadir direction can be estimated by an image-based approach with an accuracy of about 750 m and 15 m respectively. Regarding the attitude of the camera, a roll as well as a pitch angle and a yaw angle can be estimated with an accuracy of about 0.075 deg and 0.00175 deg respectively. This would result in an accuracy for the absolute position of about 1 km and in an accuracy of about 0.1 deg for the angle of the attitude quaternion.

However, it is important to mention that regarding the application of AVS these results can only be taken as a rough assessment since many assumptions made do not fit to AVS. First, the intrinsic parameters are determined when calibrating the camera and this procedure is never error-free. Second, the approach of AVS should also be applied when the satellite is not pointing in exact nadir direction. Both circumstances degrade the accuracy of the pose estimate. Additionally, as will be described later, the final pose estimation performed in AVS is generally based on more than 4 points using the iterative least-squares approach and not the direct solution of the spatial resection, which also affects the accuracy of the final pose estimate.

Chapter 3

Design

In this chapter, the design is given of how the GCP database is created as well as of how the process of matching features and estimating the pose of the satellite onboard is performed.

3.1 Creation of the Ground Control Points Database

As stated out before, the main objective of the GCP database is to provide features that are equally distributed over the globe and have a unique appearance in satellite imagery so they can be found by the onboard camera of a satellite. In order to create a database with GCPs consisting of feature descriptors and their geographic location, a suitable satellite image dataset is required. ‹Suitable› in this context means to meet the following requirements:

- The dataset provides RGB-images.
- Metadata is included which provides detailed information about the geographic location of each image and thereby it is possible to determine the exact on Earth position of each pixel in the images of the dataset.
- The dataset contains images covering the entire globe so that GCPs can be extracted all over the world.
- The included images are up-to-date to ensure that the detected GCPs can be found by a satellite that will be launched in the near future.
- The images have a low cloud-coverage so that features of the Earth's surface can be detected.

If such a dataset is given, the database can be created following the block diagram depicted in fig. 3.1. All following steps are performed on ground in advance of the launch of the satellite. First, two RGB-images capturing the same area on ground are selected. Next, features are extracted from each of the two RGB-images. It's important to note, that the camera which later should perform the onboard matching and pose estimation might have a different FOV than the camera with which the used dataset was taken. Thus, the onboard images might cover a smaller area on ground. To ensure that the onboard camera captures a sufficient number of GCP features within its FOV, the GCPs

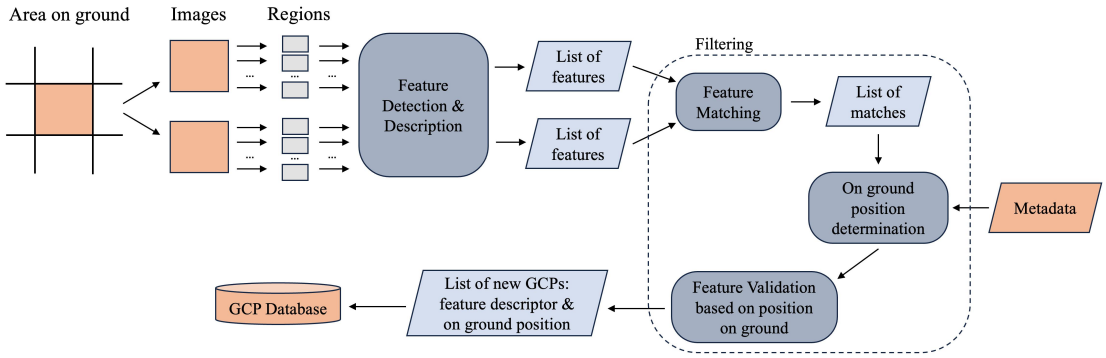


FIGURE 3.1: Block diagram of the design of the GCP database creation.

must be evenly distributed. Thus, each taken RGB-image of the dataset is divided into smaller regions. A feature detection algorithm is applied to each region to detect features, and afterwards, descriptor vectors are created for these keypoints. Subsequently, all features from one RGB-image are collected, resulting in a list of evenly distributed features. The division of the image depends on the FOV as well as of the swath width and height of the onboard camera and of the camera that was used to create the used dataset.
Next, all the obtained features of the two images needs to be filtered. Therefore, the two sets of detected features are matched. The resulting matches should include features that remain stable over time, can be identified in different images, and have unique signatures for recognition during the matching process. These characteristics increase the likelihood of correct detection and matching onboard.
To ensure that only features of correct matches are included in the database, an additional validation step is carried out. This involves determining and comparing the geographic positions of the matched features on the ground. These positions on ground can be determined based on the features' pixel positions and the provided metadata of the dataset. After filtering, a list of GCPs consisting of feature descriptors and the corresponding position on ground results. This data is then stored in the database. This process is repeated for each area on ground, that should be covered by the database.
In order to be able to estimate the pose of the camera and thus of the satellite, the stored on ground positions of the GCPs should be defined with respect to the ECEF frame in cartesian form $(x, y, z)^{ECEF}$. Thus, depending on the used image dataset and metadata, some additional coordinate transformation steps might need to be considered in the implementation.

3.2 Onboard Pipeline

The goal of the onboard matching and pose estimation is to accurately determine the position and attitude of the satellite with respect to the ECI frame based on an onboard image. A block diagram of how this process is designed is shown in fig. 3.2.
In order to be able to validate the onboard image-based pose estimation, satellite image data is required that provide RGB-images combined with information of following aspects:

1. **Camera parameters**: For each image to be tested, the intrinsic parameters of the used camera are required, as these are needed to be able to estimate the pose of the camera based on an image and 2D-3D point correspondences (see section 2.5.3).
2. **Reference pose**: In order to be able to validate the pose estimate of the developed pipeline, it is mandatory to have the corresponding ‹true› pose of the satellite at the time of image acquisition. Otherwise, no evaluation of the entire pipeline is possible.

If such satellite image data is given, at first, features are detected and described using the ORB algorithm. Then, the set of features is matched with the features stored in the GCP database. This results in 2D-3D point correspondences which includes correct matches as well as outliers since not each feature detected in the onboard image is also a GCP stored in the database.
In order to eliminate all mismatches, two additional feature validation steps are performed. First, only matches are considered further which have a descriptor distance smaller than a certain threshold α. Even though the feature detection algorithm should only extract unique appearances of the Earth's surface, covering a wide area in the database can lead to similarities among the detected keypoints and thus among the descriptor vectors. Therefore, only relying on selecting matches, that have a small distance could cause false matches still being included. In order to circumvent this difficulty an additional, second validation method is applied, that should guarantee that the resulting matches include only correct ones.
This is implemented by a RANSAC based algorithm (see section 2.3.6) using a projection from 3D to 2D as the underlying model. In detail, the algorithm comprises of the steps described in algorithm 1. $\vec{f}$ represents a match consisting of a pixel position and a corresponding ECEF position. $\boldsymbol{K}$ is the camera matrix as introduced in section 2.5.1. It is obtained by calibrating the camera in advance and, as mentioned above, should be provided by test dataset. ρ denotes the reprojection error used for determining whether a match is an inlier or not. M and n denote the number of combinations consisting of four different matches and the number of matches in total respectively. Unfortunately,

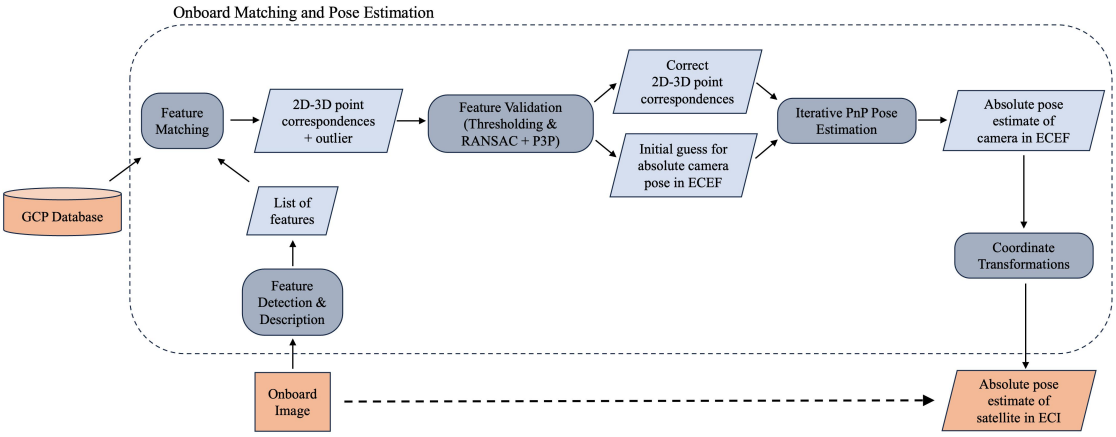


FIGURE 3.2: Block diagram of the design of the onboard matching and pose estimation.

Algorithm 1 P3P-RANSAC

$n_{inliers} \leftarrow 0$
$[\boldsymbol{R}, \vec{t}] \leftarrow [eye(3,3), \vec{0}]$
for $i \leftarrow 1 : M$ **do**
    Select systematically the next tuple of four correspondences $(\vec{f}_1, \vec{f}_2, \vec{f}_3, \vec{f}_4)$
    $[\boldsymbol{R}_r, \vec{t}_r] =$ solveP3P$(\vec{f}_1, \vec{f}_2, \vec{f}_3, \vec{f}_4, \boldsymbol{K})$
    Compute the number of inliers n_r with respect to $\boldsymbol{K}$, $\boldsymbol{R}_r$,$\vec{t}_r$ and ρ

    if $n_r > n_{inliers}$ **then**
        /* *Better projection found* */
        $n_{inliers} \leftarrow n_r$
        $\boldsymbol{R} \leftarrow \boldsymbol{R}_r$
        $\vec{t} \leftarrow \vec{t}_r$

    if $n_r == n$ **then**
        /* *Projection fits to all correspondences* $\Rightarrow$ *all matches are correct* */
        break

the method has one main disadvantage. The solution of the P3P problem is computed based on four points and therefore always results in at least four inliers. If the best rotation and translation returned by the algorithm only fits for 4 features, no statement can be made about whether the features are correct matches or not. In this case, all inliers are discarded to ensure that no false matches are passed on to estimating the satellite's attitude and position.
After these validation steps, a set of $2D$-$3D$ point correspondences, that should not include any mismatches and an initial guess for the pose of the camera is obtained. This is then taken as input for an iterative algorithm solving the PnP problem in order to get an optimal solution for all correspondences (rotation and translation from ECEF to camera frame). Next, the resulting pose is transformed, first, to ECEF with respect to body frame, based on the position and orientation of the camera relative to the satellite's body, and second, to ECI with respect to body frame, based on the current time stamp. This, results in a pose estimate of the satellite that can be used further, e.g. for attitude control algorithms.

Chapter 4

Implementation

In this chapter, the implementation of creating the database as well as of matching the features and estimating the pose of the satellite onboard based on the design explained in the previous chapter is described. Furthermore, the used image data used for both processes, the applied software libraries as well as the hardware on which the project is executed are presented.

4.1 Creation of the Ground Control Points Database

4.1.1 Image Data for Database

A dataset that meats all the requirements mentioned in section 3.1is provided by the Landsat 8 satellite. The Landsat program of National Aeronautics and Space Administration (NASA) and United States Geological Survey (USGS) provides the longest continuous acquisition of satellites imagery of Earth [35, 52]. Landsat 8 was launched on 11 February 2013 into a sun-synchronus, near-polar orbit (98.2° inclination) with an altitude of 705 km and has a 16-day repeat cycle. It carries two sensors: An Operational Land Imager Sensor that collects data from nine spectral bands (Band 1-9) and a Thermal Infrared Sensor that collects data from two spectral bands (Band 10-11) [46]. Each imagery data acquired represents one scene of the Worldwide Reference System-2 (WRS-2). Therefore, each image taken by Landsat 8 can be identified by a path, a row and the date of recording. For each image a metadata-file is provided. Among others, it includes the date of recording, WRS-2 path and row, sun elevation angle, cloud coverage, pixel height and width as well as the geographic location of the four image's corners in Universal Transverse Mercator (UTM) and World Geodetic System-84 (WGS-84) coordinate systems. As the satellite has been in orbit for over 10 years now, a large amount of image data is available. It can be accessed and downloaded at no charge from a variety of data portals such as EarthExplorer.
The Landsat 8 data is divided into different Collections, Levels and Tiers resulting in many different datasets. What the characteristics of each category are, is summarized in [41] or can be read in detail in [47, 48, 50, 51]. As Collection 1 data is no longer available to download and Collection 2 offers several improvements, such as improved geographic location accuracy and radiometric calibration as well as a more detailed and consistent metadata-file, the Collection 2 data is chosen. The pictures taken by the

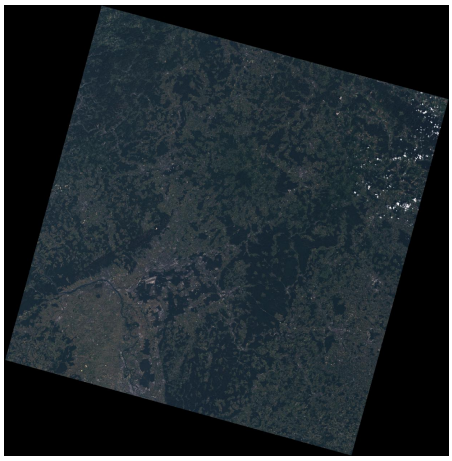


FIGURE 4.1: Example of generated RGB-images based on the provided Landsat 8 data.

onboard camera won't be preprocessed to eliminate effects of the atmosphere but represent the Top of Atmosphere (TOA) reflectance. Thus, the Landsat Level-1 data is used. Additionally, to be able to extract high-quality features in terms of precise geographic location the Tier-1 data is selected, resulting in the Landsat 8 Collection-2 Level-1 Tier-1 data set.
In order to obtain an image that can be analyzed for creating the database, the three single band color channel images (Red, Green, Blue) needs to be combined in order to generate a RGB-image. This conversion is not trivial and is described in detail in [41]. When this conversion is applied, realistic RGB-images are generated (see fig. 4.1). These images have a shape of approximately $8000 \times 8000 \times 3$ pixels. With a resolution of 30 m, each image covers an area of about 240 km$\times$ 240 km (including the black borders).

4.1.2 Software Structure

In order to test and evaluate the onboard matching and image-based pose estimation for AVS, a software is implemented for creating a database based on the design described in section 3.1. Fig. 4.2 illustrates the structure of the developed software. Initially, a set of WRS-2 scenes to be included in the database is selected. For each scene, two Landsat 8 data packages are chosen to generate RGB-images. During selection, it is ensured that these scenes have 0% cloud cover to fully reveal the ground, capturing only distinct features of Earth's surface and not of cloud appearances. Since the impact of seasonal changes on feature recognition hasn't been explored yet, the two selected data packages should have been recorded roughly a year apart within the same month ($\pm$1). Additionally, the more recent data package must be from no later than 2020 to ensure that the extracted GCPs are up to date and recognizable in future missions as well.
As described during the design, the division of the dataset images depends on the FOV as well as on the swath width and height of the onboard camera and the Landsat 8 Operational Land Imager Sensor. For this study, we use the parameters of TOM, whose

camera has a swath width of about 50 km and a swath height of about 40 km. Since the Landsat 8 RGB-images cover approximately 240 km × 240 km, they are divided into $\frac{240\,\text{km}}{50\,\text{km}} = 4.8 \approx 5$ columns and $\frac{240\,\text{km}}{40\,\text{km}} = 6$ rows. Due to the black border of the image (see fig. 4.1), the regions in the four corners hardly depict any Earth's surface which is why these regions are neglected. This leads to a total number of 26 regions for each scene.

Features are then detected and described within each of the regions using the ORB algorithm (see section 2.3.6). The used parameters of the algorithm are listed in appendix A.3. If the number of detected features exceeds a value of 700, only the best 700 keypoints with respect to their Harris score are selected. This value is chosen based on the work of [41]. The next step involves matching the detected features of the two images depicting the same scene. During matching, for each feature in the first image, the nearest neighbor in the second image is identified based on the Hamming distance of the descriptors. Since the database is prepared before the mission, computational effort is not a concern, and high accuracy of the matches is essential. Therefore, the Brute-Force method is used, comparing each feature from the first set with every single feature in the second set.

Afterwards, the on ground position with respect to the UTM coordinate system of each matched feature is determined. These on ground positions can be determined based on the features' pixel positions and the UTM positions of the corners of the image, which are provided by given Landsat 8 metadata. How this is done in detail can be read in [41]. Next, the validation is performed based on the on ground positions of the matched features. During this validation, a match is only marked as a correct one, if the x- and y-coordinate regarding the UTM system of the matched features f and g differ within a certain threshold ϵ:

$$\left|f_x^{utm} - g_x^{utm}\right| \leq \epsilon \quad \text{and} \tag{4.1}$$

$$\left|f_y^{utm} - g_y^{utm}\right| \leq \epsilon. \tag{4.2}$$

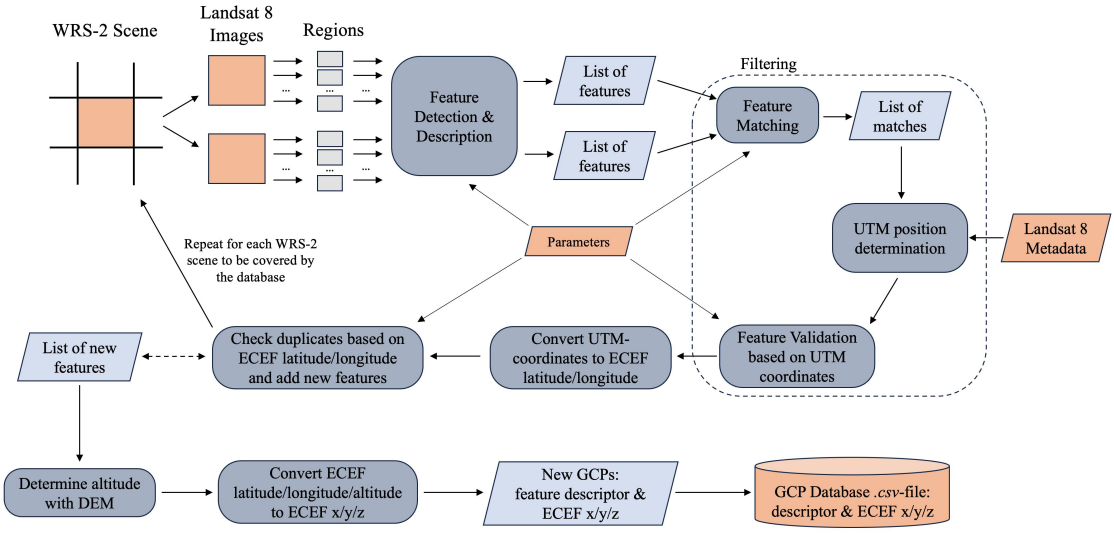


FIGURE 4.2: Block diagram of the implementation of the GCP database creation.

For this implementation, a threshold of $\epsilon = 60$ is chosen corresponding to an accepted error of 60 m in each direction. This value is chosen on the basis of the pixel resolution of 30 m px^{-1} and to catch errors in the UTM position calculation due to rounding.
All of these remaining correctly matched features are candidates for GCPs. Next, for each match, the UTM position of the feature, that was detected in the more recent image, is converted to a latitude and longitude with respect to the ECEF frame. As the WRS-2 scenes do overlap to some extend, it can happen that the same feature is detected and correctly matched in two different scenes. To avoid including duplicates in the database, the next step consists of comparing all newly selected features with all features that have already been chosen as GCPs. This is done by simply checking the latitude and longitude of a new feature n with all already selected features s in a similar way as during feature validation:

$$\left| n_{lat}^{ECEF} - s_{lat}^{ECEF} \right| \leq \epsilon_{angle} \quad \text{and} \tag{4.3}$$

$$\left| n_{lon}^{ECEF} - s_{lon}^{ECEF} \right| \leq \epsilon_{angle} \quad \text{with} \tag{4.4}$$

$$\epsilon_{angle} = \frac{180 \cdot \epsilon}{\pi \cdot r_{earth}} \text{ deg} \ . \tag{4.5}$$

For r_{earth} the radius of the Earth is used in meters. After repeating this process for all selected WRS-2 scenes, a list of GCPs is obtained consisting of their feature descriptors mapped with the latitude and longitude of their on ground positions with respect to the ECEF frame. For the feature descriptor, the ones extracted from the more recent images are used. In the next step, the altitude of the GCPs is determined using the latitude as well as the longitude and a Digital Elevation Model (DEM). Based on the latitude, longitude and altitude of a point, the cartesian coordinates can easily be computed. This leads to a list of GCPs consisting of feature descriptors and the corresponding $(x, y, z)^{ECEF}$ positions. This data is then stored in the database. All parameters for the processes of feature detection, description, matching, validation as well as for checking duplicates are given as input in a separated file.

4.2 Onboard Matching and Pose Estimation

4.2.1 Image Data for Testing

Unfortunately, during the research of this internship, no acquired satellite image data that is suitable regarding the requirements mentioned in section 3.2 could be found. However, Friesen [15] developed an Earth Observation Simulator (EOS), which enables the creation of individual satellite imagery data by specifying the orbit position $(x, y, z)^{ECI}$, camera attitude $(q_1, q_2, q_3, q_4)^{ECI}$, time stamp and camera parameters (image width w and height h in px as well as horizontal FOV $hFOV$ in deg). One example of an image, that was generated using the EOS is depicted in fig. 4.3. The camera matrix can be determined based on the parameters that can be specified for the EOS. The images of the EOS are created with the assumption of an ideal camera, so no scaling

differences, no sheering and no distortions occur:

$$m = s = \Delta x(\vec{x}, \vec{q}) = \Delta y(\vec{x}, \vec{q}) = 0$$

For the offset between sensor and image frame, half of the image width and height are used:

$$x_H = \frac{w}{2} \text{ and } y_H = \frac{h}{2}$$

The focal length can be determined as follows:

$$f = \frac{1}{2} \cdot \frac{w}{\tan \frac{hFOV}{2}}$$

This leads to the simplified camera matrix with f, x_H and y_H as described above:

$$\boldsymbol{K} = \begin{bmatrix} f & f \cdot s & x_H + \Delta x(\vec{x}, \vec{q}) \\ 0 & f \cdot (1+m) & y_H + \Delta y(\vec{x}, \vec{q}) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f & 0 & x_H \\ 0 & f & y_H \\ 0 & 0 & 1 \end{bmatrix}$$

4.2.2 Software Structure

In order to test the onboard matching and pose estimation regarding multiple aspects, two different pipelines are implemented. For both pipelines, test images of the EOS are used and features are detected and described using ORB. If the number of detected features exceeds a value of 840, only the best 840 keypoints with respect to their Harris score are selected. This value is chosen based on the work of [41]. The feature matching is performed by finding the nearest neighbor regarding the descriptor Hamming distance in the GCP database for each detected feature. For searching the nearest neighbor, the Brute-Force approach is used as high accuracy is needed and the number of false matches must be kept to a minimum since they could be fatal for the upcoming pose estimation. During validation, as described in the design, first the matches face a

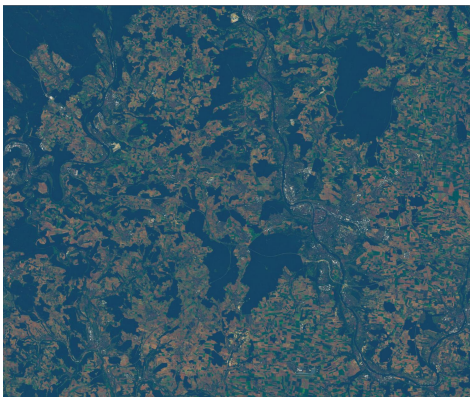


FIGURE 4.3: Example of an image that was generated by the EOS using the parameters of the camera integrated in TOM and an appropriate orbit with pointing to the ground station located in Wuerzburg.

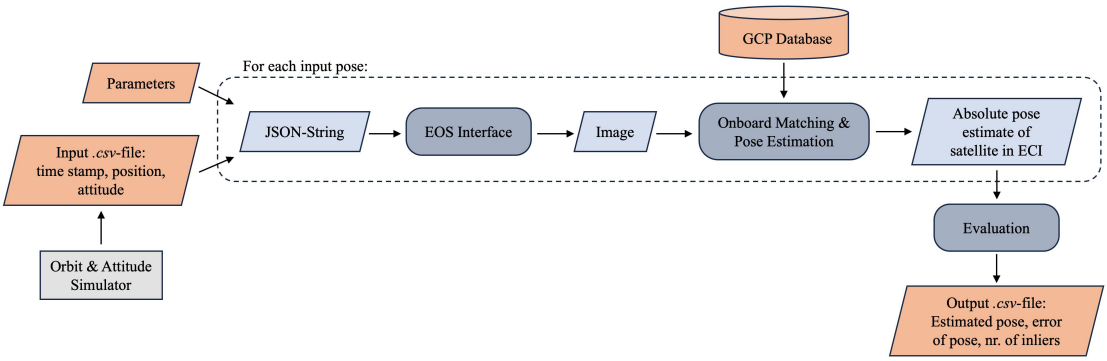


FIGURE 4.4: Block diagram of the implementation of the simulation pipeline.

thresholding with respect to the descriptor Hamming distance The work of Redelbach [41] has shown that a threshold value of $\alpha = 30$ and a reprojection error of $\rho = 30$ are an appropriate choice. If more than 50 matches fulfill this requirement, only the best 50 matches regarding the descriptor Hamming distance are selected. For solving the P3P problem during the validation using RANSAC, the algorithm of Gao et al. [16] is used. For the final iterative pose estimation using all inliers a least-squared approach is applied as described in section 2.5.3.

Simulation Pipeline

First, a simulation pipeline is implemented to be able to easily test and evaluate multiple different scenarios for the onboar matching and pose estimation process. The block-diagram of this implementation is illustrated in fig. 4.4. First, an orbit and attitude simulator, that is described later, is used to create a *.csv*-file containing a list of time stamps as well as of positions and attitudes with respect to the ECI frame. Thus, each line represents one input pose for which the onboard matching and pose estimation will be performed. This *.csv*-file and a file specifying all parameters of the used algorithms and the camera parameters are given as input for the pipeline. For each input pose, a corresponding satellite image needs to be generated. This is done by using the EOS of [15]. Therefore, a JSON-string specifying the pose of the satellite and the parameters of the camera is created and sent to the EOS via a HTTP request. The received image is then together with the GCP database fed as input to the onboard matching and pose estimation process. This is implemented as described in section 3.2 using the algorithms specified above. The estimated pose of the satellite with respect to the ECI frame is then evaluated and the results are saved in the form of an output *.csv*-file containing for each input pose the estimated pose, the error of the pose and the number of inliers used for pose estimation.

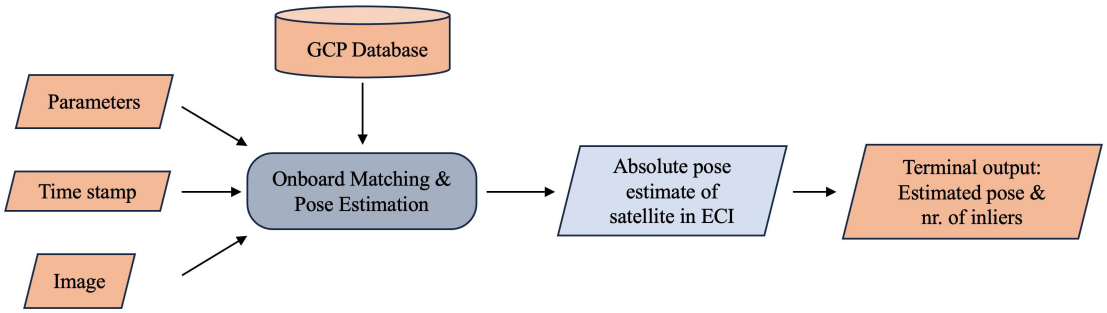


FIGURE 4.5: Block diagram of the implementation of the embedded pipeline.

Embedded Pipeline

During the final application of AVS, the onboard software should only receive an input image, captured by the onboard camera. The onboard feature matching and pose estimation is then performed based on this image, the current time stamp, the specified parameters and the database. To simulate this process, especially to analyze the runtime in section 5, another pipeline is implemented, in the following referred to as the embedded pipeline. The structure is depicted in fig. 4.5. The output of the pipeline, consisting only of the estimated pose and the number of used inliers, is printed to the terminal. Here, the input image is generated in advance using the data of the orbit and attitude simulator and the EOS. This block named *onboard matching and pose estimation* is implemented in the same way as for the simulation pipeline.

4.3 External Software Libraries

For implementing the creation of the database as well as the two onboard matching and pose estimation pipelines, the programming language Python 3.9.13 is used. As it is sufficient for the purposes of this application the database is implemented as a *.csv*-file. *Open Source Computer Vision Library (OpenCV)* is integrated to get access to the implementations of the algorithms used for image analysis and pose estimation. In general, *OpenCV* is a cross-platform library that can be used to develop computer vision and machine learning applications. It supports the operating systems Linux, Windows, Android and MacOS and has interfaces for Python, C++, Java and Matlab. In order to get the altitude for a certain latitude and longitude during database creation, the *Open-Elevation API* is used which provides a DEM that can be accessed using HTTP requests [31]. In order to convert UTM-coordinates to ECEF latitude and longitude, the python package *pyproj* is used [40]. In order to get realistic orbit and attitude data that can be used to generate satellite image data with the EOS, an *Orekit* based orbit and attitude simulation is used. Orekit is an open source library for space dynamcis [38]. For the implementation of the onboard matching and pose estimation, besides *OpenCV* no other library is used to enable an easy integration on other embedded systems.

4.4 Hardware

To execute and test the software, three hardware devices are used: The first one is an *Apple MacBook Pro 14" 2021* with a *Apple M1 Pro* chip and 16 GB unified memory. It is used for creating the database as well as for the simulation pipeline. The second one is a *Raspberry Pi Zero 2 W*. The third device is a *Raspberry Pi Compute Module 4*. The two Raspberry Pi models are used to evaluate the runtime performance of the embedded pipeline.

Chapter 5

Tests and Evaluation

In the following chapter, the onboard pipeline is tested. The used data is presented in section 5.1. As the first tests revealed some problems with the original version of the pipeline, the implementation is adapted to some extend. Therefore, first, the simulation pipeline is used to qualitatively evaluate the performance with respect to the accuracy and to improve the pipeline. This is described in sections 5.3, 5.4 and 5.5. Afterwards, the simulation pipeline is used to quantitatively evaluate the accuracy in section 5.6. Lastly, the embedded pipeline is used to evaluate the runtime of the pipeline section 5.7.

5.1 Test Data

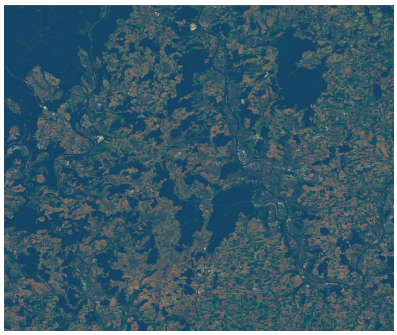
5.1.1 Database

First, the database needs to be created in order to be able to test the onboard pose estimation. This is done by using the implementation presented in section 4.2.2. Therefore, 12 WRS-2 scenes are chosen covering approximately the area of Germany with a displayed area of about 550 km $\times$ 700 km. The corresponding Landsat 8 image data (see section 4.1.1) is taken between April and June to circumvent possible influences of seasonal changes. If no image data with 0% cloud cover was available, images from March and July are also considered. The exact IDs of the used data can be found in appendix A.2. The created database contains 20318 features.

5.1.2 Onboard Images

For testing the onboard pose estimation, four overflights with pointing to a ground station in Wuerzburg performed at the end of June 2024 are chosen. The overflights are simulated using the Orekit orbit simulator and using possible orbit parameters of the TOM mission. For each overflight, 15 datasets are taken at intervals of 4 seconds comprising of a time stamp as well as of a position $(x, y, z)^{ECI}_{true}$ and an attitude $(q_1, q_2, q_3, q_4)^{ECI}_{true}$. For each dataset of each overflight, an image is created using the EOS based on the generated data and the camera parameters of the TOM camera. As 15 datasets are taken, each image sequence covers a time period of 60 seconds. It has not yet been investigated how strong the impact of the elevation angle of the satellite on the performance of onboard feature matching is. Thus, during the chosen overflights the satellite passes the ground station with roughly nadir view at a maximum elevation

(A) Overflight 4 - Image 1



(B) Overflight 4 - Image 15

FIGURE 5.1: Two test images created by the EOS of the fourth overflight. Figure (A) and (B) shows the first and last image of the sequence respectively.

angle of 88.1, 87.2, 80.4 and 79.3 deg respectively. An example of start and end image of the first overflight is shown in fig. 5.1.

5.2 Evaluation Metrics

In order to evaluate the estimated position and attitude of the satellite given as cartesian coordinates $(x, y, z)^{ECI}_{est}$ and as a quaternion $(q_1, q_2, q_3, q_4)^{ECI}_{est}$ respectively, a metric needs to be defined. The accuracy of the position is evaluated based on the absolute position error to the used input position of the EOS:

$$p_{err} = \sqrt{(x^{ECI}_{true} - x^{ECI}_{est})^2 + (y^{ECI}_{true} - y^{ECI}_{est})^2 + (z^{ECI}_{true} - z^{ECI}_{est})^2} \tag{5.1}$$

The accuracy of the attitude is evaluated based on the angle of the error quaternion of the estimated and the true attitude:

$$\vec{q}_{err} = \left(\vec{q}^{ECI}_{true}\right)^{-1} \odot \vec{q}^{ECI}_{est} \tag{5.2}$$

$$\alpha_{err} = 2 \cdot \arccos(q_{err,0}) \tag{5.3}$$

where $\odot$ denotes the Hamilton product for quaternions.

5.3 Onboard Feature Validation

5.3.1 Standard RANSAC

During the first tests, the simulation pipeline, implemented as described in section 4.2.2, is executed. In the following, this version of the pipeline is referred to as the *Standard RANSAC* one. During the first test, the simulation pipeline is executed for each pose of the four overflights which includes generating the image and estimating the position and attitude of the satellite. The resulting cumulative distribution of the position and angle error over all 60 test cases is illustrated in fig. 5.2. For a better visualization, the maximum position and angle error that is illustrated is set to 1000 km and 180 deg respectively. If the error of a test case exceeds one of these thresholds, the error is set to

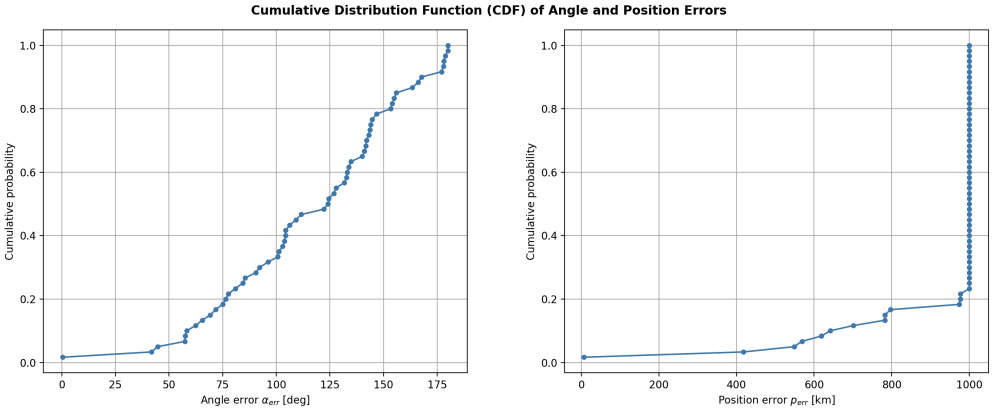
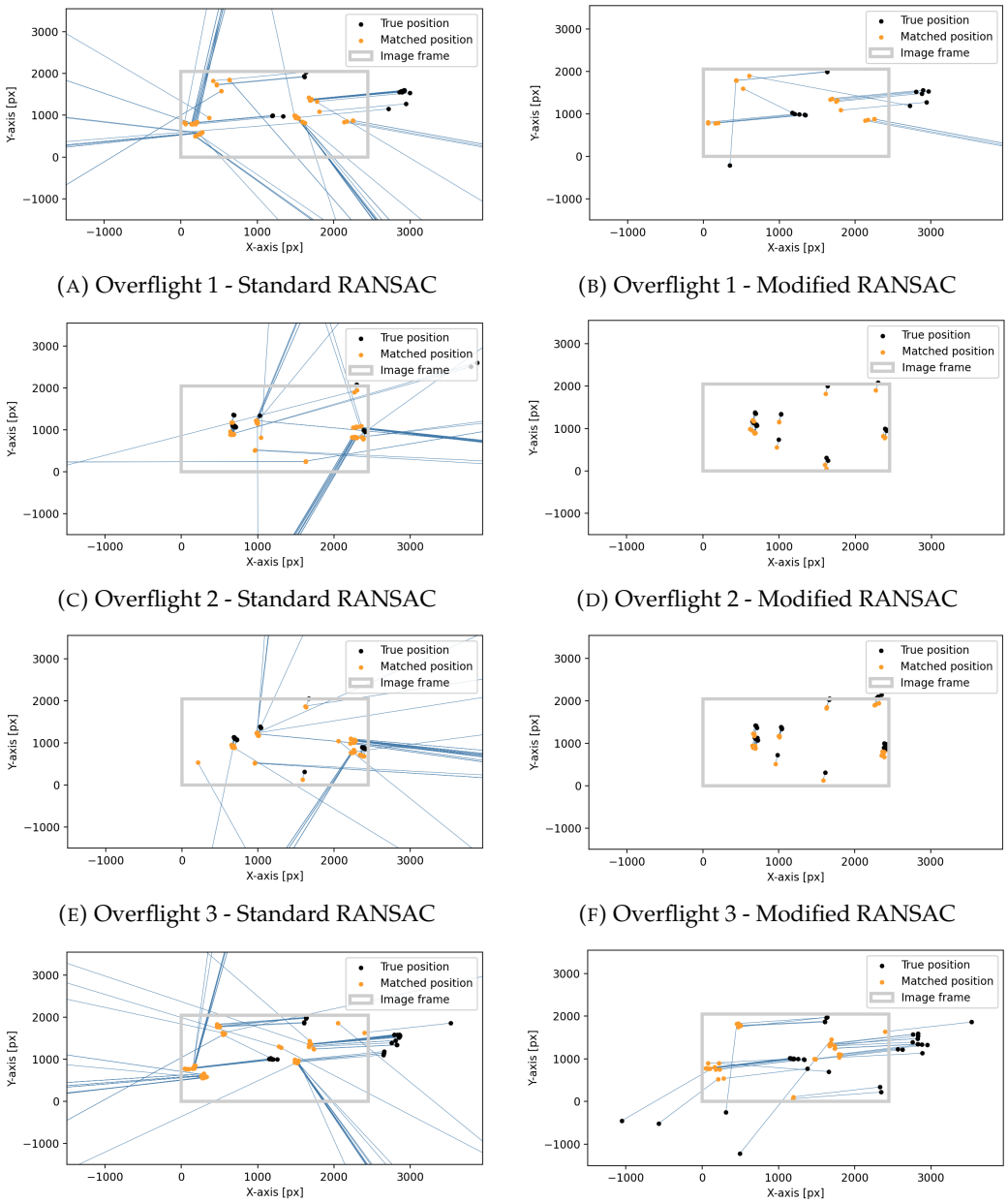


FIGURE 5.2: Cumulative distribution function of the position and angle error over all 60 test cases using the first version of the pipeline.

the maximum. Additionally, if no pose estimation was possible, because after matching and validation less than 4 matches remain, the position and angle error for these cases are also set to the maximum illustrated value. This is done for all plots of the cumulative distribution of the position and angle error. As can be seen, except of one case the error of the estimated attitude and position is larger than about 40 deg and 420 km respectively for all cases. It needs to be noted that for all cases an attitude estimate is achieved because all errors are less than the maximum illustrated error. So the high amount of data points in the plot for the position error at a value of $p_{err} = 1000$ km is because all of these cases have actually a much larger error but are set to 1000 km for a better visualization.

These results let assume that mismatches are fed to the pose estimation, even though a validation step is performed by applying the thresholding and the P3P-RANSAC algorithm. This assumption can be confirmed when looking at the left column of fig. 5.3. In these plots the image positions of the matched features used for pose estimation are marked in orange. To be able to see if a match is correct, the $3D$ coordinates with respect to the ECI-frame of the GCPs are projected to the image plane. This results in the true image positions of the used GCPs. The true and the matched position of a GCP are connected by a blue line. The projection of the $3D$ coordinates of the GCPs is done by computing the true projection between ECI-frame and image plane based on the the known calibration matrix of the camera, $(x, y, z)^{ECI}_{true}$ and $(q_1, q_2, q_3, q_4)^{ECI}_{true}$.

It can be seen, that for all GCPs the true and the matched image positions differ. But when having a closer look, for each overflight there is a set of GCPs that have the same shift between the true and matched position. These corresponds to the correct matched GCPs. A reason for the shift could be systematic errors of the images generated by the EOS, e.g. the created image does not exactly corresponds to the input position or attitude. The variation of the shift in magnitude and direction for the different overflights can be explained by the different viewing angles and thus a different error propagation. Another reason could be the rounding errors when computing the true projection. In the following, the focus is only on achieving the same shift between true and matched image position for all GCPs that remain after validation.



(A) Overflight 1 - Standard RANSAC

(B) Overflight 1 - Modified RANSAC

(C) Overflight 2 - Standard RANSAC

(D) Overflight 2 - Modified RANSAC

(E) Overflight 3 - Standard RANSAC

(F) Overflight 3 - Modified RANSAC

(G) Overflight 4 - Standard RANSAC

(H) Overflight 4 - Modified RANSAC

FIGURE 5.3: Plot of the image plane of the camera. The orange points mark the image positions of the features that are matched to GCPs and are used for pose estimation. The black points mark the image positions of the projected $3D$ coordinates with respect to the ECI-frame of the matched GCPs using the input position and attitude of the satellite, called true position. The light gray frame shows the size of the image frame. The blue lines connect the matched position with the true position of each GCP. Each row presents the results for all images of each of the four overflights applying the standard RANSAC version and the modified RANSAC version of the implementation respectively.

The implemented feature validation has two steps: thresholding and P3P-RANSAC. A reason why mismatches are not removed during thresholding is because due to the large database, the probability is high that for a detected feature in the onboard image, the database contains by chance a descriptor that does not correspond to the detected feature but has a very small descriptor distance. Thus these features are matched and

accepted during thresholding. When a set of matches that contains mismatches is fed to the P3P-RANSAC, a randomly chosen subset is used to determine the projection. This projection is applied to all other matches and if the reprojection error ρ is below a certain threshold, in this case $\rho = 30$, a match is marked as an inlier. The problem is that there are regions that contain many correctly matched features. This can be seen in fig. 5.3, where a few hotspots of a lot of data points appear. When now a randomly chosen subset of matches is chosen, that contains one of the features of a hotspot, it is likely that this computed projection could fit to the other correct matches of the same hotspot as well. As a result, a projection that is far away from the true projection is chosen as the best guess due to the larger number of inliers. If this initial guess and the subset that contains wrong matches is fed to the final pose estimation step, the iterative RANSAC algorithm will also compute a pose estimate with a large error.

5.3.2 Modified RANSAC

To improve the validation step and thus to provide a reliable feature matching process, the P3P-RANSAC is replaced by a modified, self-developed algorithm that is also based on the RANSAC concept. In the following, this version of the implementation is called *modified RANSAC*. For this purpose, additional, given knowledge of the application is used to early detect outliers or a wrong pose estimate. First, the FOV of the camera is known and thus approximately the size of the covered area on ground. This gives knowledge about the maximum distance of the matched GCPs on ground. So when choosing a subset of matches to compute a projection model, all selected GCPs must have a distance smaller than a certain threshold d_{max} to each other. Additionally, when a model is computed and applied to all other matches, again, only GCPs are taken into account that have an on ground distance smaller than d_{max} to all GCPs used to determine the model. Second, the orbit height is known approximately in advance. Thus, the norm of the estimated position vector can be restricted up to a certain range. So if the norm of the position vector of the computed projection model is not within the range $\left[h_{min}; h_{max}\right]$, the model is not considered further and a new subset is chosen. Additionally, as it could happen, that two features are matched to the same GCP, it is also checked for each new tested match, whether the GCP is already included in the set of inliers. Furthermore, to improve the runtime, as soon as it is detected that two GCPs have a distance larger than d_{max}, all possible subsets which includes these two GCPs are skipped. The detailed procedure of the algorithm is illustrated in algorithm 2. $\mathcal{F}$ denotes the set and n the number of all matches that remain after thresholding.

Algorithm 2 Modified P3P-RANSAC

$n_{in} \leftarrow 0$
$\mathcal{S}_{in} \leftarrow \{\}$
$[\boldsymbol{R}, \vec{t}] \leftarrow [eye(3,3), \vec{0}]$

for $i \leftarrow 0 : n-4$ **do**
    $\vec{f}_1 \leftarrow \mathcal{F}[i]$

    for $j \leftarrow i+1 : n-3$ **do**:
        $\vec{f}_2 \leftarrow \mathcal{F}[j]$
        if distance$(\vec{f}_1, \vec{f}_2) > d_{max}$ **then** continue

        for $k \leftarrow j+1 : n-2$ **do**:
            $\vec{f}_3 \leftarrow \mathcal{F}[k]$
            if $\big($distance$(\vec{f}_1, \vec{f}_3)$ **or** distance$(\vec{f}_2, \vec{f}_3)\big) > d_{max}$ **then** continue

            for $l \leftarrow k+1 : n-1$ **do**:
                $\vec{f}_4 \leftarrow \mathcal{F}[l]$
                if $\big($distance$(\vec{f}_1, \vec{f}_4)$ **or** distance$(\vec{f}_2, \vec{f}_4)$ **or** distance$(\vec{f}_3, \vec{f}_4)\big) > d_{max}$ **then**
                    continue
                $[\boldsymbol{R}_r, \vec{t}_r] =$ solveP3P$(\vec{f}_1, \vec{f}_2, \vec{f}_3, \vec{f}_4, \boldsymbol{K})$
                if norm$(\vec{t}_r) > h_{max}$ or norm$(\vec{t}_r) < h_{min}$ **then** continue
                $\mathcal{S}_{in,r} \leftarrow \{\vec{f}_1, \vec{f}_2, \vec{f}_3, \vec{f}_4\}$
                $n_r \leftarrow 4$

                for $\vec{f}_x$ in $\mathcal{F}$ with $x \neq i, j, k, l$ **do**:
                    $m \leftarrow \max($distance$(\vec{f}_1, \vec{f}_x)$,distance$(\vec{f}_2, \vec{f}_x)$, distance$(\vec{f}_3, \vec{f}_x)$,distance$(\vec{f}_4, \vec{f}_x))$
                    if $m > d_{max}$ **then** continue

                    $\boldsymbol{P} \leftarrow$ computeProjection$(\boldsymbol{K}, \boldsymbol{R}_r, \vec{t}_r)$
                    if $\vec{f}_x$ is inlier with respect to $\boldsymbol{P}$ and ρ **then**
                        $\mathcal{S}_{in,r} \leftarrow \mathcal{S}_{in,r} \cup \{\vec{f}_x\}$
                        $n_r \leftarrow n_r + 1$

                if $n_r > n_{in}$ **then**
                    /* *Better projection found* */
                    $n_{in} \leftarrow n_r$
                    $\boldsymbol{R} \leftarrow \boldsymbol{R}_r$
                    $\vec{t} \leftarrow \vec{t}_r$
                    $\mathcal{S}_{in} \leftarrow \mathcal{S}_{in,r}$

                if $n_r == n$ **then**
                    /* *Projection fits all correspondences* $\Rightarrow$ *all matches are correct* */
                    break

return $[\mathcal{S}_{in}, \mathbf{R}, \vec{t}]$

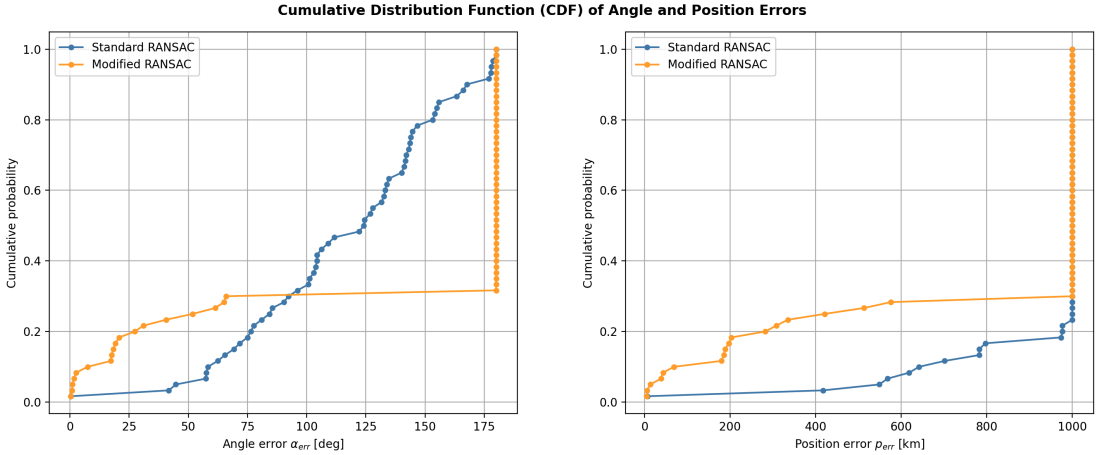


FIGURE 5.4: Cumulative distribution function of the position and angle error over all 60 test cases using the standard and the modified RANSAC.

As the swath width and height and height of the TOM camera are about 50 km and 40 km and the overflights are approximately with nadir pointing, the maximum distance of two GCPs on ground is set to $d_{max} = 60$ km. As the orbit height of the simulated data is about 500 km, the range of the allowed norm of the position vector is set to $[h_{min} = 200 \text{ km}; h_{max} = 800 \text{ km}]$. As can be seen in the right column of fig. 5.3, when applying the modified RANSAC, for the overflight 2 and 3 only correct matched features are used for pose estimation. For overflight 1 and 4, the number of outliers can be reduced as well. Unfortunately, some mismatches are still fed to the pose estimation step. The explanation for this is the following. If the set of matches given as input to the modified RANSAC contains no more than 3 correct matches, but happens to contain a subset that still passes all additional tests, this subset is used for pose estimation.
The resulting cumulative distribution of the position and angle error over all 60 test cases when using the modified RANSAC version of the pipeline is illustrated in fig. 5.4. For comparison, the curve of the standard RANSAC is added as well. It can be seen, that due to the large amount of data points at the maximum illustrated error, both for the angle and the position, that only for about 30% of the test cases a pose estimate is possible. But for these cases, the error of the estimate can be reduced significantly. Therefore, for all upcoming versions of the implementation, the standard RANSAC is replaced with the modified one. The reason why for 70% of the cases no estimation is possible is that too less correct features remain after thresholding, so no single projection model during the modified RANSAC can be found that passes all tests.

5.4 Thresholding vs. Taking Best n Matches

To circumvent this difficulty, the implementation is adapted. The step of thresholding the matched features is replaced by taking the best n_{best} matches with respect to the descriptor distance. In this way, it is ensured that a sufficient and constant number of matches is fed to the modified RANSAC algorithm. Another possibility would be to increase the threshold α. Unfortunately, it is difficult to find the best value for α for all

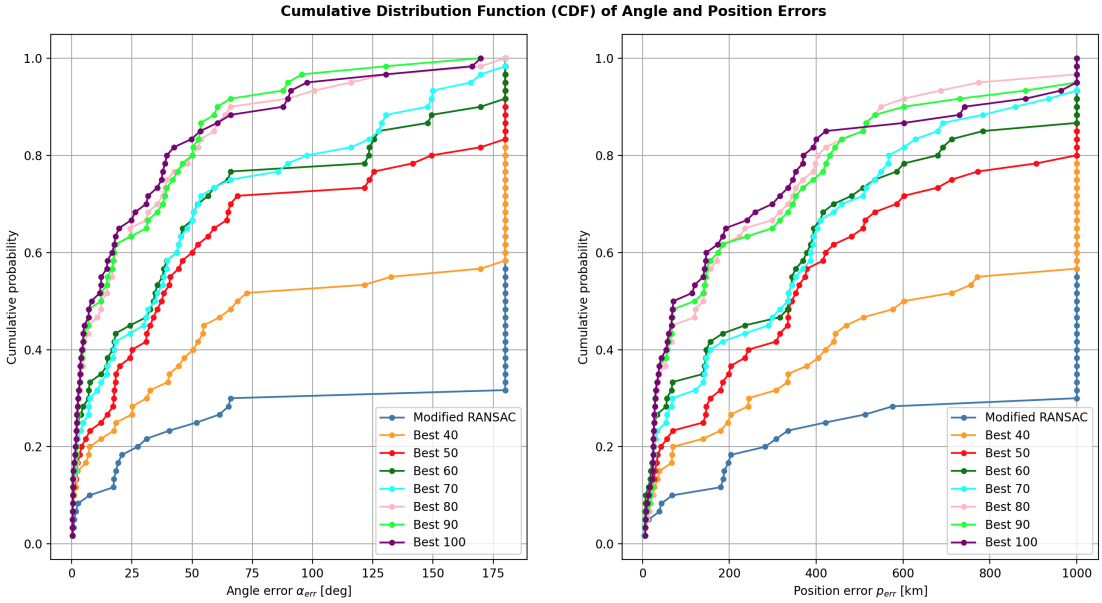


FIGURE 5.5: Cumulative distribution function of the position and angle error over all 60 test cases using the modified RANSAC with thresholding (curve *Modified RANSAC*) and using modifed RANSAC with selecting best n_{best} matches after matching (curves *Best* n_{best}).

possible scenarios, as even the descriptor distance of the same GCP to a correct matched feature varies greatly when different conditions of image acquisition occur.
As it is not obvious which value for n_{best} achieves the best performance regarding pose estimation accuracy, each onboard image of all overflights is tested with $n_{best} = 40, 50, 60, 70, 80, 90, 100$ in order to get a first impression of the order of magnitude. Of course the absolute best value again depends on several factors such as different conditions of image acquisition and cannot be determined in general. Fig. 5.5 and fig. 5.6 illustrate the resulting cumulative distribution of the position and angle error and of the number of inliers used for pose estimation respectively over all 60 test cases when using the modified RANSAC version of the pipeline combined with thresholding or with taking the best n_{best} matches. It can be seen that the accuracy of the pose estimate can be improved significantly when replacing the thresholding. Additionally, it can be observed that the errors can be reduced when n_{best} is increased, especially from $n_{best} = 40$ to $n_{best} = 80$. When further increasing n_{best}, the accuracy is improved only minimally. This behavior can be explained by the higher number of inliers that are fed to the final pose estimation when replacing the thresholding and increasing n_{best} As the worst case runtime of the modified RANSAC algorithm can be estimated to be proportional to n_{best}^4 due to the four for loops, n_{best} should be kept as small as possible. Thus, in the following $n_{best} = 80$ is considered as the best value trading-off accuracy and runtime.
Another important aspect can be observed when looking at the total number of inliers in fig. 5.6. In the work of Redelbach [41], it was stated out as questionable whether the matching would work if images are used for testing and creating the database that has been taken by different cameras. During the tests of [41], on average 28.5 matches remain after validation when using the ORB detector and descriptor and images of the Landsat 8 dataset for the database and for testing. When using the adapted pipeline

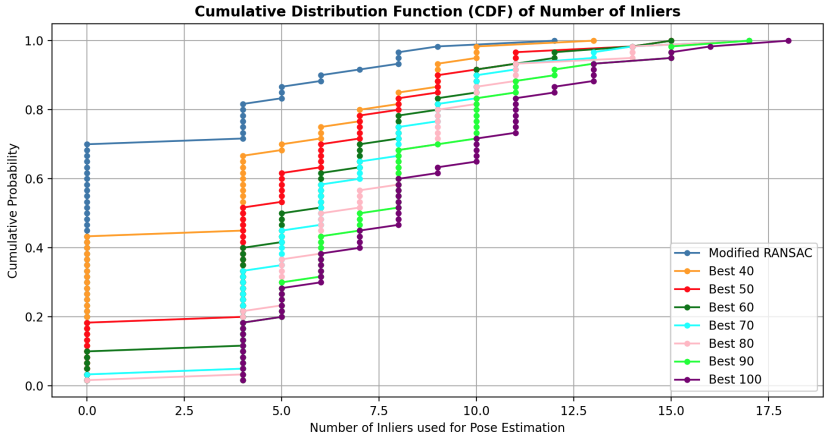


FIGURE 5.6: Cumulative distribution function of the number of inliers used for pose estimation over all 60 test cases using the modified RANSAC with thresholding (curve *Modified RANSAC*) and using modified RANSAC with selecting best n_{best} matches after matching (curves *Best* n_{best}).

and the EOS images for testing, it can be seen that the number of inliers is much less. However, enough matches for estimating the pose of the camera still remain so the approach of AVS still works when different image sources are used for creating the database and for the onboard pipeline.

5.5 Dividing Onboard Image for Feature Detection

As stated out in section 2.5.4, the accuracy of the pose estimate depends on the size of the spanned area by the used point correspondences. In order to improve the accuracy, the implementation is adapted by trying to force a larger distribution of the matched GCPs over the onboard image. This is done by dividing the onboard image into four

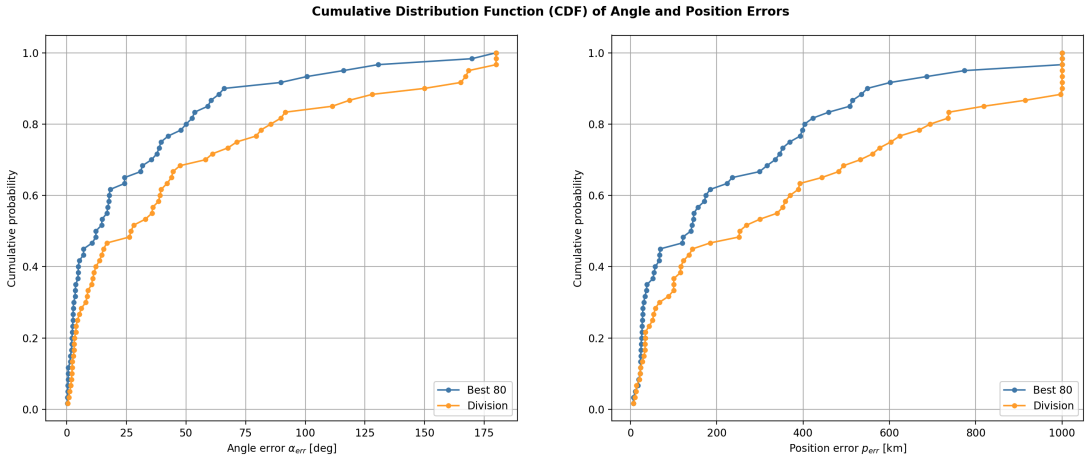


FIGURE 5.7: Cumulative distribution function of the position and angle error for pose estimation over all 60 test cases using the modified RANSAC with selecting best $n_{best} = 80$ matches (curve *Best* 80) and using the modified RANSAC with division of the feature detection (curve *Division*).

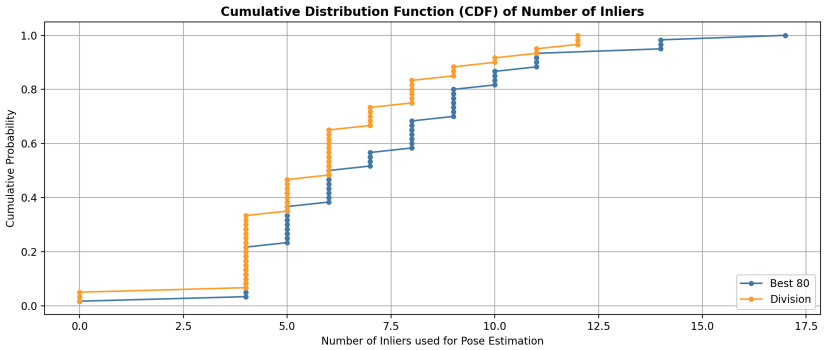


FIGURE 5.8: Cumulative distribution function of the number of inliers used for pose estimation over all 60 test cases using the modified RANSAC with selecting best $n_{best} = 80$ matches (curve *Best* 80) and using the modified RANSAC with division of the feature detection (curve *Division*).

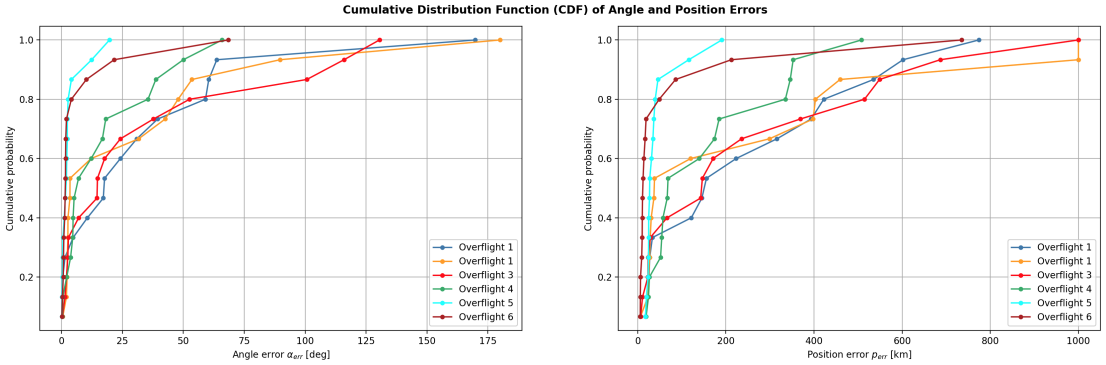
quadrants before the feature detection is done. Then, the features are detected and described separately within each of the four images. In order to get again a total number of 840 detected and described features, the best 210 features with respect to the Harris score are selected within each image. Each set of features is then matched to the GCP database and the best $n'_{best} = 20$ matches are chosen to obtain a total number of $n_{best} = 80$ matches. The procedure for all other steps remains unchanged.
Fig. 5.7 and fig. 5.8 illustrate the resulting cumulative distribution of the position and angle error as well as of the number of inliers used for pose estimation respectively over all 60 test cases when using the modified RANSAC version combined with taking the best $n_{best} = 80$ matches and with dividing the feature detection. It can be seen that when applying the division of the feature detection, the number of inliers that remain after validation and with it the accuracy of the estimate is reduced. This can be explained because in some areas of the image, there simply are no GCPs visible that are stored in the database. Thus, the detection and matching process within these areas of the image leads to mismatches that have a large descriptor distance. However, when 20 matches are selected, all of these mismatches are fed to the validation where they are then detected as outliers. In other areas of the image, there are a lot of GCPs visible. These are also correctly matched and have a small descriptor distance. But, when only selecting the best 20 matches in this area, it might happen that correct matches are discarded. The number of matches of each area that are taken into account for validation cannot be chosen arbitrarily large to ensure that all correct matches are included because of the resulting runtime issues of the validation. Therefore, the division of the feature detection does not represent a feasible improvement for the pipeline and is thus not pursued further.

5.6 Accuracy

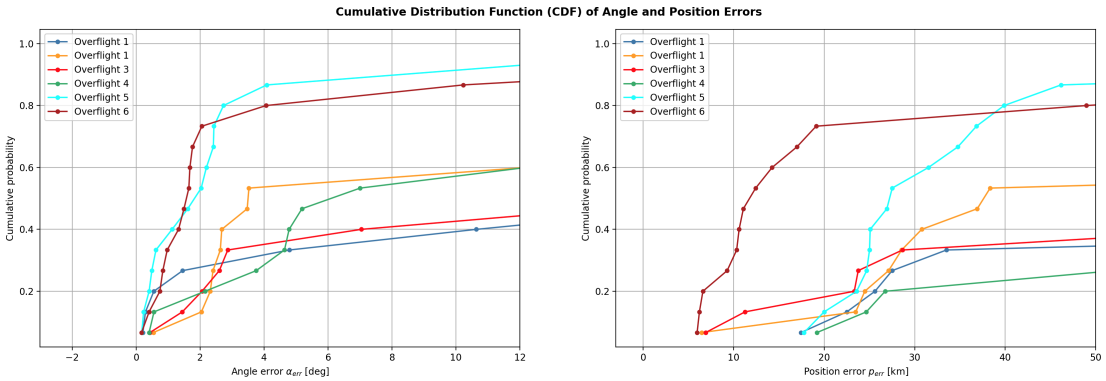
After improving the pipeline by qualitatively comparing the accuracy of the pose estimates of the different versions, in the following the accuracy is evaluated quantitatively. Taking the results of the previous performed tests into account, the final version of the

pipeline for this work is chosen to consist of using the modified RANSAC and replacing the thresholding validation by taking the best $n_{best} = 80$ matches. For this purpose, data for two additional overflights is created with the orbit and attitude simulator. The cumulative distribution of the angle and position error over each of the six overflights are plotted separately in fig. 5.9.

It can be observed that the accuracy of the pose estimate varies greatly during one overflight even though the consecutive images differ only slightly as can be seen in fig. 5.1. Thus, the same GCPs should be visible within all images of an overflight. Additionally, the accuracy of the pose estimate varies greatly between the different overflights as well, even though all of the images depicts approximately the same on ground area as the satellite is pointing to the same target during all overflights. So, the same GCPs should be visible during all overflights as well. Therefore, the current implementation is not stable over time during one overflight and not stable for different perspectives of various overflights. The difference in the performance between the overflights can be explained by the difference of the number of matches used as well as the number of mismatches included. To get an impression of these two values, fig. 5.10 presents the true and matched positions of the GCPs in the onboard images. For all overflights

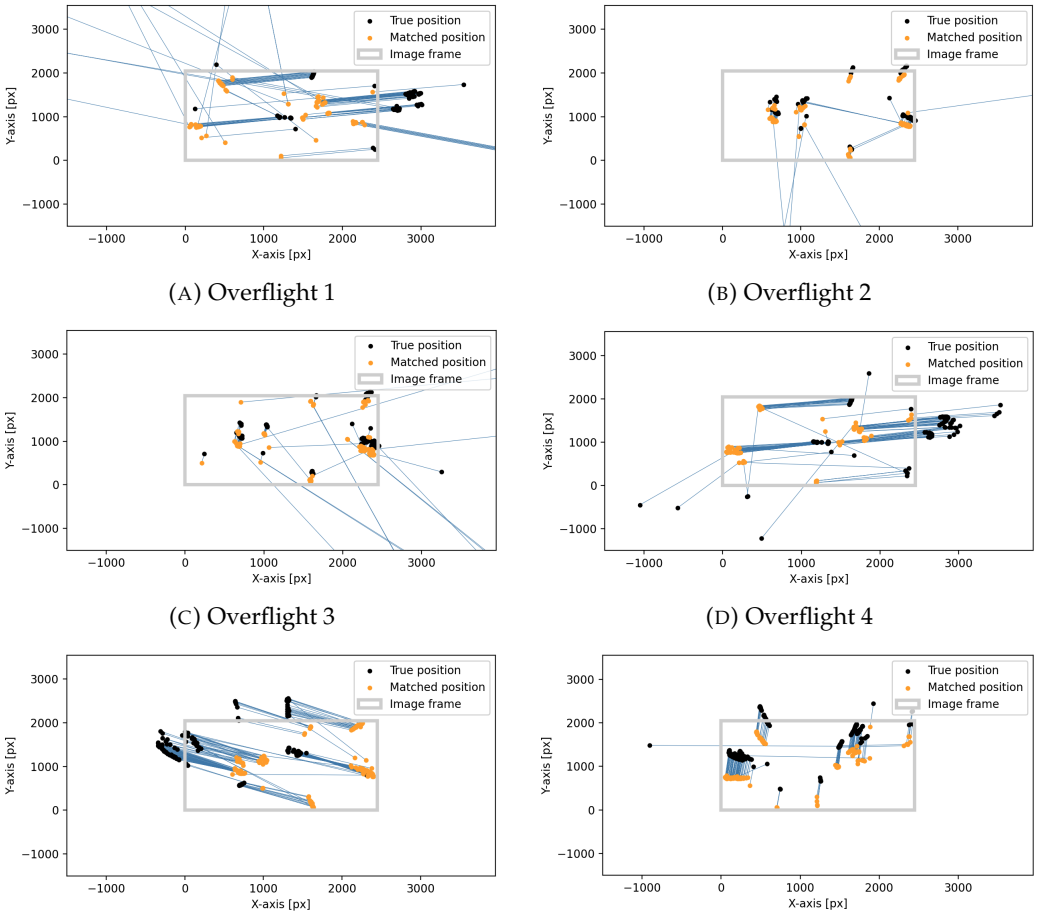


(A) CDF - Total



(B) CDF - Zoom

FIGURE 5.9: Cumulative distribution function of the position and angle error for pose estimation over six overflights separately with 15 images each using the modified RANSAC with selecting best $n_{best} = 80$ matches. Figure (A) presents the entire range of the angle and position errors whereby figure (B) gives a zoom to the range of about $0 - 10$ deg and $0 - 45$ km.



(A) Overflight 1

(B) Overflight 2

(C) Overflight 3

(D) Overflight 4

(E) Overflight 5

(F) Overflight 6

FIGURE 5.10: Plot of the image plane of the camera. The orange points mark the image positions of the features that are matched to GCPs and are used for pose estimation. The black points mark the image positions of the projected 3*D* coordinates with respect to the ECI-frame of the matched GCPs using the input position and attitude of the satellite, called true position. The light gray frame shows the size of the image frame. The blue lines connect the matched position with the true position of each GCP. Each image presents the results for all images of each of the six overflights using the final version of the pipeline.

the number of data points and thus the number of matched GCPs differs greatly. Furthermore, some overflights contain many mismatches whereby e.g. overflight 5 and 6 barely includes wrong matched GCPs. An explanation for this behavior cannot be found at the time of writing. Therefore, no general statement for the accuracy of the pose estimation for the implementation can be made.

For all overflights except of overflight 5, there is at least 1 image that results in an angle and position error of more than 50 deg and 400 km respectively. This let assume, that for these overflights during many test images mismatches are still included in the final pose estimation which can be confirmed by fig. 5.10. However, for overflight 5 and 6, the attitude is estimated with an angle error of less than 1 deg for 40% and with an angle error less than 4 deg for about 80% of the test cases. This shows that the proposed pipeline of image-based attitude estimation for a satellite could have real potential. For the position estimation, the error of about 80% of the test images of overflights 1-5 is larger than 20 km. Even though the position error of overflight 6 is less than 20 km for

about 70%, the remaining 30% still result in position errors of more than 5 km. Therefore, none of the position estimates would be accurate enough to provide added value for the onboard position determination.
When considering the presented estimations of the theoretical precision of section 2.5.4, not a single test run meets the accuracy of 1 km for the absolute position and of 0.1 deg for the attitude. As can be seen in fig. 5.10, the worse performance is not only due a tight distribution of the GCPs, but there must be other sources for the resulting larger error. As stated out before, it can be observed that the true and the matched image positions of the GCPs differ. This let assume that there is a systematic error when creating the images using the EOS. This assumption can be confirmed when looking at the test images, as the pointing target is not centered in the image, as can be seen in fig. 5.11. This could be the major source of the resulting large position and attitude error. It could be that the pose estimations actually have much smaller errors but the referenced true pose that is used for comparison does not match the tested image. This needs to be confirmed by using another dataset for test images in the future. But there are also other potential sources for errors in the estimate. The feature detection during the database creation as well as during onboard matching only provides a discrete pixel resolution, which leads to deviations. As the coordinates of the GCPs are based on the pixel locations, this could result in pose estimation errors. When creating the database of the GCPs, errors can result due to the deviations of the DEM to the true heights. Even though the EOS creates images by simulating an ideal camera without any distortions and accurately known camera parameters, this is obviously not the case, which also leads to errors.
Additionally, it has to be added that for this work the satellite is pointing to the same target during all overflights. Thus, the same area on ground, in detail the area around Wuerzburg, is depicted in all images. Therefore, the results cannot be taken as representatives for applying the method to images showing a completely different area. For sure, there are on ground areas which would result in better accuracy and others which would result in worse accuracy. This work only aims to analyze the feasibility of this approach in general and many more cases have to be analyzed in the future.

FIGURE 5.11: Example of a test image (first image of overflight 1) where the pointing target on ground is marked by an orange circle.

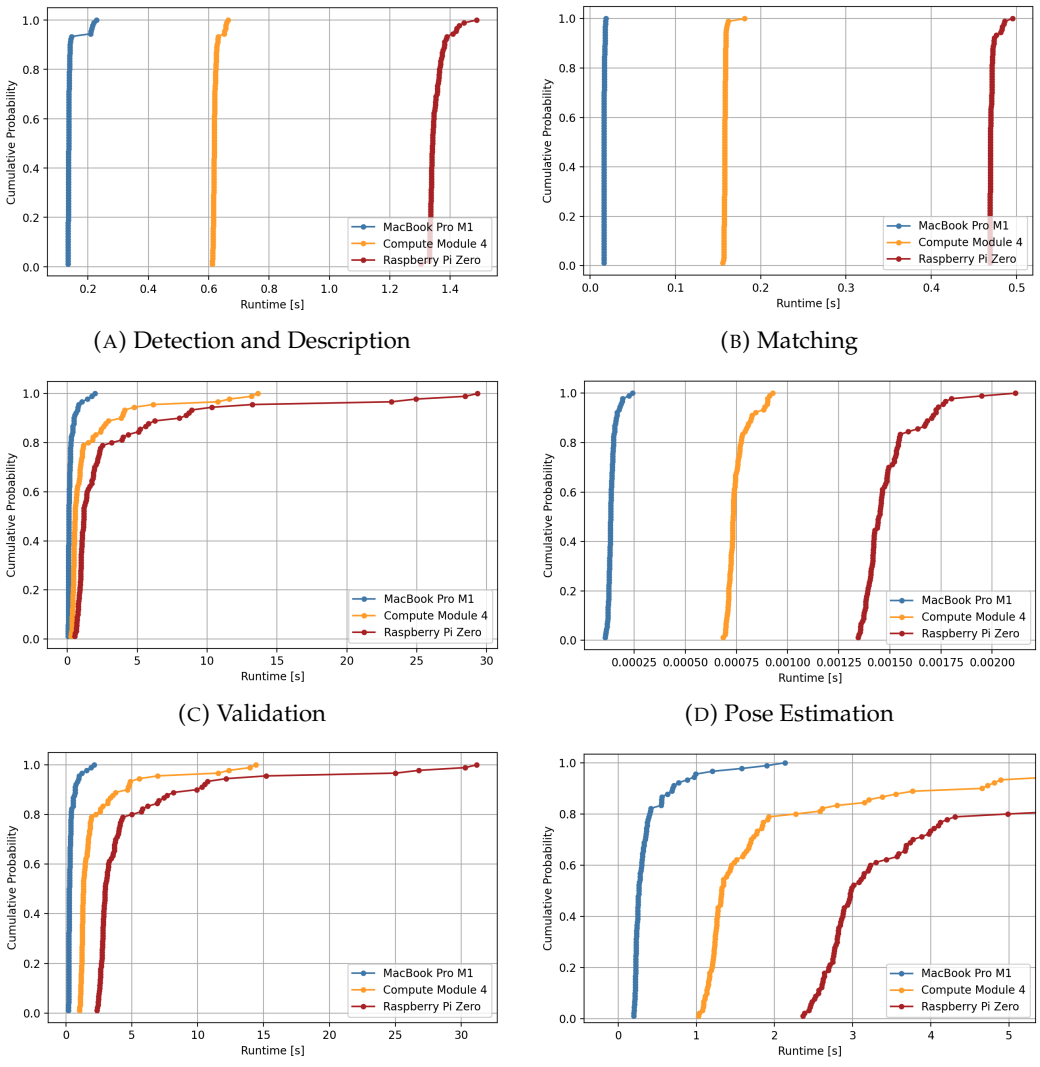
5.7 Runtime

Since the pipeline is executed onboard and the position and attitude need to be estimated in real-time to be usable for attitude and orbit control, the runtime is analyzed in the following. The developed RVS is planned to be implemented on a *Compute Module 4* which is integrated onboard the satellite. Thus, the *Compute Module 4* could be a realistic choice for AVS as well and is therefore mainly considered. In order to see if hardware with more or less computational power could also be an option for the application of AVS regarding runtime, the embedded pipeline is tested on three different devices: *Apple MacBook Pro* with an *Apple M1 Pro* chip, *Raspberry Pi Compute Module 4* and *Raspberry Pi Zero 2W*. Fig. 5.12 shows the cumulative distribution of the runtime for the different steps of the pipeline over all test images of the six overflights and table 5.1 lists the corresponding mean values and standard deviations. It can be seen, that for all devices the runtime of the detection and description as well as of the matching process is approximately constant for all test images. This observation is obvious as for all images the same amount of features is detected and matched with the database. The runtime of the validation step for all devices varies greatly especially for about 20% of the test cases. The variation can be explained, because the runtime of the validation step depends on the quality of the provided matches. If a lot of mismatches are fed to the modified RANSAC algorithm which can be detected early by the various implemented conditions, a lot of iterations are skipped and the execution is faster. But, if either the mismatches can pass the conditions or mainly correct features are passed to the validation, this step takes much longer. So, unfortunately, as a result, in the desired case where only correct matches are provided for the validation, the runtime is much longer. The runtime of the final pose estimation mainly depends on the quality of the initial guess and the number of used matches. As the number of used matches varies for the test images, the runtime of the pose estimation varies as well, but only slightly compared to the validation step. The plot of the total runtime looks similar to the one of the validation step. This gets obvious when considering the absolute values of the runtime for the different steps. The matching and the pose estimation can be executed much faster than the detection and description step and especially than the validation

Runtime	**Hardware Devices**		
	MacBook Pro M1	**Compute Module 4**	**Raspberry Pi Zero**
Detection + Description [s]	0.142 ± 0.020	0.621 ± 0.010	1.35 ± 0.03
Matching [s]	0.0165 ± 0.0005	0.158 ± 0.003	0.471 ± 0.004
Validation [s]	0.234 ± 0.337	1.52 ± 2.59	3.26 ± 5.59
Pose Estimation [s]	$1.40e\text{-}4 \pm 0.22e\text{-}4$	$7.51e\text{-}4 \pm 0.54e\text{-}4$	$1.49e\text{-}3 \pm 0.14e\text{-}3$
Total [s]	0.392 ± 0.340	2.30 ± 2.60	5.09 ± 5.60

TABLE 5.1: Mean value and standard deviation of the runtime of the tested hardware devices for different steps of the onboard pipeline: Feature detection and description, matching, validation as well as total runtime.

which takes about 1-14 s for the *Compute Module 4*. Thus, in order to improve the total runtime, the detection and description as well as the validation step need to be improved regarding computational effort. And when looking at the absolute values of the total runtime, it gets clear that the runtime either needs to be improved by adapting the pipeline in a more efficient way or by using an embedded hardware device with more computational power. Otherwise, the application of AVS at the current version would not be real-time applicable, neither using a *Raspberry Pi Zero* nor using a *Compute Module 4*. Looking at the total runtime when using the *MacBook Pro*, it gets clear that there are devices which could execute the pipeline in real-time, but hardware needs to be found that can also be integrated onboard the satellite. It has to be noted here, that the pipeline is implemented in *Python*. In order to improve the runtime, the implementation could be transferred e.g. in C++. However, it is not expected that the runtime



(A) Detection and Description

(B) Matching

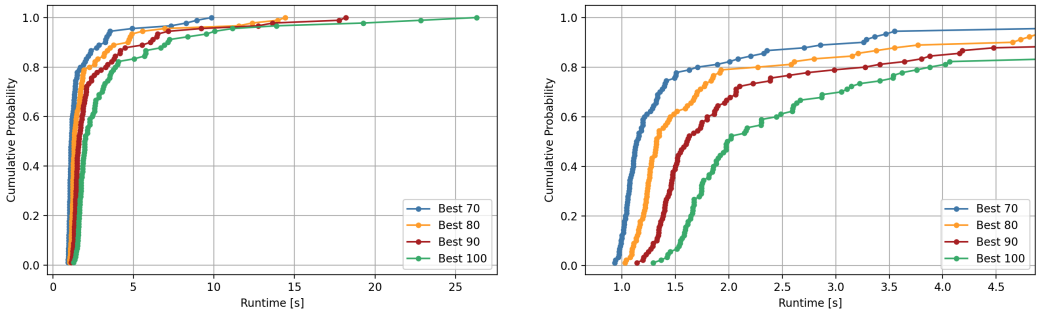
(C) Validation

(D) Pose Estimation

(E) Total - full range

(F) Total - zoomed

FIGURE 5.12: Cumulative distribution function of the runtime for all steps of the final version of the pipeline over all test images of the six overflights: (A) Detection and Description, (B) Matching, (C) Validation, (D) Pose Estimation, (E) and (F) Total Runtime. For each of the tested devices - MacBook Pro M1, Raspberry Compute Module 4 and Raspberry Pi Zero 2W - the results are depicted.



(A) Total - full range (B) Total - zoomed

FIGURE 5.13: Cumulative distribution function of the total runtime of the final version of the pipeline over all test images of the six overflights for different n_{best}. The tests are executed on the Raspberry Pi Compute Module 4.

will then be so much faster that the pipeline can be used in real time.
As a last aspect, which is tested within this work, the total runtime of the pipeline with respect to the number of matches n_{best} is analyzed. The cumulative distribution of the total runtime for $n_{best} = 70, 80, 90, 100$ is depicted in fig. 5.13. These values are chosen as lower ones do not achieve a satisfying accuracy as stated out in section 5.4. It can be seen that the runtime increases for higher values of n_{best}, but not as expected when considering the worst case runtime of the modified RANSAC algorithm that is proportional to n_{best}^4. This is because a large proportion of the matches that are taken into account additionally for higher n_{best} are mismatches. As stated out before, mismatches can be detected early during the modified RANSAC algorithm and so many iterations can be skipped. Thus, the runtime does not increase as much as expected. If the total runtime in general can be reduced, it might be worth considering using larger values for n_{best}, even if this only improves the accuracy of the pose estimate a little, as the runtime does not increase as drastically as expected.

Chapter 6

Outlook

In this chapter, ideas for improving the current version of the pipeline and for extending the approach are presented. Additionally, aspects that need to be investigated in the future are introduced.

6.1 Ideas for Improvements

As stated out in section 5.6, the accuracy of the DEM influences the accuracy of the GCP coordinates and thus the accuracy of the attitude and position estimate. As it cannot be found which specific dataset is used by the *Open-Elevation API* [31], it is hard to tell how accurate the provided heights are. Therefore, the accuracy of the estimates could be improved by trying out other APIs, e.g. *Google Earth Engine* [17] or *OpenTopoData* [37].
Even though a detailed investigation on the used feature detection, description and matching algorithms has been done by Redelbach [41], it is not ensured that the combination of the ORB detector and descriptor with a Brute-Force matching method achieves the best result for all applications. Thus, the accuracy of the estimate and the runtime of the pipeline should be evaluated again with respect to the used detection and description algorithm as well as with respect to their parameters when e.g. image data is available that was acquired by the onboard camera which is also used in future missions. Additionally, it might be useful to replace the Brute-Force method for matching by the Fast Library for Approximate Nearest Neighbors (FLANN). This would improve the runtime of the matching process. However, as the runtime of the matching process only contributes a small part to the total runtime of the current version and the reliability of the matches might decrease when using FLANN, this approach should be treated with caution. Major advantages regarding the total runtime can probably only be achieved when dealing with larger databases.
Not only the impact of the algorithms for detecting, describing and matching the features on the performance of the pipeline should be investigated again, but also of the used algorithm for solving the P3P problem. The current implementation uses the approach of Gao et al. [16], but there are many more, e.g. AP3P [24] or EPnP [29].
Furthermore, the validation step that is currently implemented by the developed modified RANSAC algorithm needs to be improved with respect to reliability of the selected

matches as well as with respect to its runtime. At the time of writing there are no specific ideas for improving this algorithm yet.

6.2 Ideas for Extensions

In the following, different ideas are presented how the current approach of AVS could be extended.

6.2.1 External Initial Estimate

During all tests, the used database covered only approximately the area of Germany and all test images displayed the region around Wuerzburg. This was done to demonstrate the general feasibility of the approach. However, in the desired application, AVS should be operational during the entire orbit. Thus, the database would contain many more features. This will lead to a much longer runtime of the matching process when using the current pipeline and therefore to a longer total runtime. Furthermore, it is also to be expected that the probability of a feature being matched to an incorrect GCP will increase because there is another GCP somewhere in the world that happens to have a smaller descriptor distance. These mismatches will than probably be detected during the validation applying the modified RANSAC algorithm but the number of remaining features for the final pose estimation might decrease significantly.
One way to counteract this circumstance is to include only the GCPs of the database that can theoretically be captured by the FOV of the onboard camera. This could be done by using an external initial estimate of the attitude and position of the satellite obtained by other sensors integrated on board, e.g. star trackers, sun sensors and GPS receivers. Knowing roughly the attitude and position of the satellite and the camera parameters, the camera's FOV can be projected to the Earth's surface adding some tolerances due to the potential error of the initial estimate. Based on the coordinates of the corners of the covered area, the database can be filtered by only selecting GCPs that lie within the FOV. However, how this can be implemented in detail still needs to be investigated. Particular attention must be paid to the runtime of the filtering process. If the filtering process takes longer than the time that is saved during the feature matching, no benefit can be derived from this approach regarding runtime. Such an initial estimate could not only be helpful for improving the runtime of the pipeline but also to improve the validation step. By including a rough estimate of the position and attitude, the developed modified RANSAC algorithm could be adapted, such that only models are further considered which are in line with the initial estimate. The image-based position estimation approach of [7] also uses such an database filtering process before matching. Unfortunately, at the time of writing no details on the implementation could be found.

6.2.2 Consecutive Images

During a satellite mission, AVS is not intended to be applied once for a single image and to output one single attitude and position estimate. AVS should be able to analyze incoming new image frames of the onboard camera at a certain rate. Therefore, the approach can be extended to not only analyze single images independently but to exploit correlations from consecutive images. The motivation for doing this is again to save computing effort and thus to improve the total runtime as well as to be able to get more information out of the images for a more accurate final attitude and position estimate. Three different possibilities of how this could be done are presented in the following. Which of the presented approaches is the most suitable one for future applications needs to be investigated in the future.

Internal Initial Estimate

Similar to the approach presented in section 6.2.1, the first idea is to use an internal initial estimate of the attitude and position of the satellite to be able to filter the GCPs of the database that could possibly lie in the FOV of the onboard camera. As the name lets suggest, this approach does not use other sensors to obtain an initial guess. For the first image that is to be analyzed, the pipeline as introduced in this work is applied and so the entire database is used for matching. A long runtime for this pose estimation is accepted as this estimate is not used for any real-time tasks of the ADCS but only as an initial estimate for the next image. This next image should be taken right after the first run of the pipeline has finished. The initial guess is used for filtering the database and thus the second pose estimate should be available in real-time and therefore can be used for attitude and orbit control. After the second image, every time an image is taken the pose estimate of the previous image is taken as an initial guess for filtering the database.

Feature Tracking

Another possibility for reducing the runtime is to not use an initial guess but to track the matched and validated features in consecutive images. In detail, the structure is as follows. For analyzing the first image, again the pipeline introduced in this work is applied, the long runtime is accepted and the pose estimate is not used for attitude and orbit control. Then, the image features, that correspond to the validated matches which are used for the pose estimation of the first image, are tracked in the next image and the matched $3D$ GCP positions are remembered. One of the most common feature tracking algorithms that could be used for this task is the Kanade-Lucas-Tomasi (KLT) tracker which is already implemented in several libraries e.g. in *OpenCV*. In this way, the runtime for detecting, describing, matching and validating the features of the next image frame can be saved. Furthermore, the results of this work make clear that the current implementation of the pipeline is not stable with respect to the number of features during one overflight. By tracking a set of features over the next image frames, this problem could be solved. Care must be taken that the runtime of the KLT tracker

is not longer than of the original pipeline, but this is not to be expected. After successfully tracking the features, a new set of 2D-3D point correspondences is obtained which then can be used to estimate the pose of the satellite. This attitude and position estimate should be available in real-time and usable for attitude and orbit control. The process of tracking these features is then repeated for the next images. It needs to be mentioned that some features will not be visible in the next image and therefore cannot be tracked as the satellite moves along the orbit and rotates. Thus, the set of matches used for pose estimation will decrease after some iterations. If the number of matches becomes too low a re-initialization step needs to be done. This step is similar to analyzing the first image, so the pipeline as introduced in this work is applied to obtain a set of new features which is then used for tracking in the next frames. But for matching the newly detected features, the previous attitude and position estimate can be used to filter the database and to make this step realizable in real-time as well. Additionally, the distribution of the matches within the image will become worse as the part of the image where no features are located becomes larger over time. As the distribution of the features influences the accuracy of the estimate (see section 2.5.4), it needs to be investigated for how many image frames this approach should be applied before a re-initialization should be performed again.

Relative Orientation

In case onboard the satellite more computational power is available for the application of AVS, the original pipeline can be extended by another parallel pipeline which computes a relative pose estimate. The relative pose estimate composes of the relative orientation and of the unit direction vector between the camera centers from which two consecutive images are taken. It is computed based on the image coordinates of the same features within two different images and on the camera parameters.
A block diagram of this extension is depicted in fig. 6.1. The procedure is as follows. During the first iteration ($n = 1$), only the already implemented, original pipeline is applied to the onboard image and for matching the entire database is used. Again, this works as an initialization step meaning that a longer runtime is accepted and the absolute pose estimate is not used for attitude and orbit control. Additionally, the inliers of the matched image features which remain after validation are remembered.
For the next iteration ($n = 2$), the absolute pose estimate of the original pipeline is used to filter the database in order to obtain only the GCPs that lie in the FOV. Therefore, the original pipeline should be able to compute an absolute pose estimate in real-time. In parallel to the known pipeline, the onboard image is forwarded to another pipeline, further referenced as the parallel pipeline. Here, the inliers of the matched image features from the original pipeline of the previous iteration are tracked in the new onboard image, which leads to 2D-2D image point correspondences. Based on these correspondences and basics of epipolar geometry, the essential matrix can be computed which encodes the relative pose between two calibrated cameras or in the case of AVS between two images taken by the same calibrated camera but from a different perspective. Details on the theoretical background of these computations can be read in [14]. Based on the essential matrix a rotation matrix from the first to the second camera perspective as

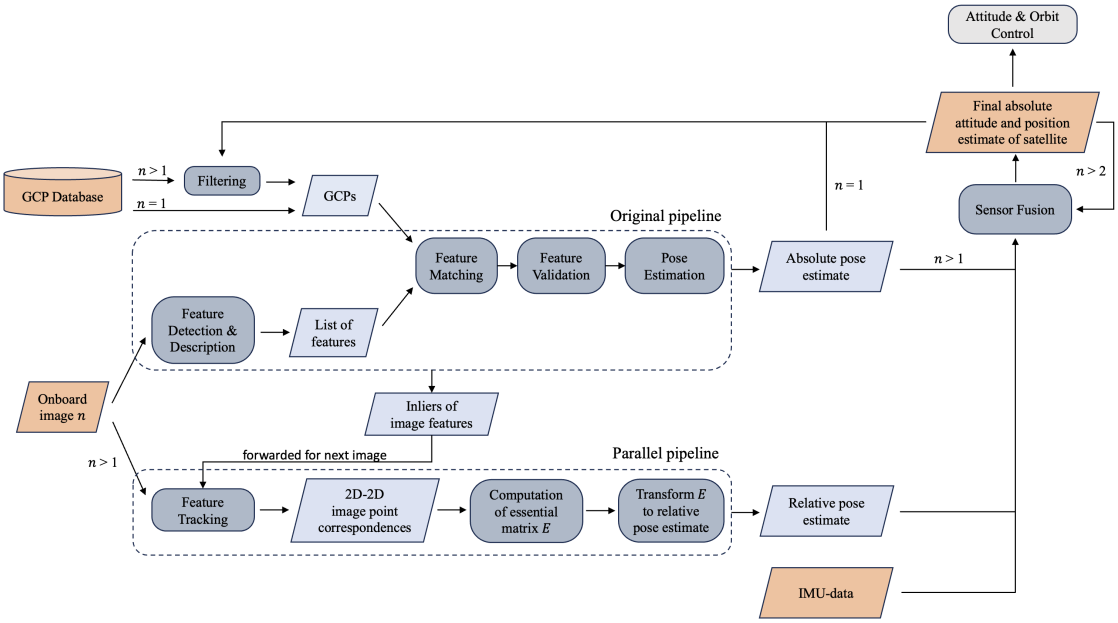


FIGURE 6.1: Block-diagram of the extension of the original pipeline of AVS by another parallel pipeline.

well as a unit direction vector from the first to the second camera center can be derived. An implementation for this can be found in *OpenCV* with the functions *findEssentialMatrix(...)* and *recoverPose(...)*.
The results, consisting of an absolute and a relative pose estimate can then be fused, optionally with data of an integrated Inertial Measurement Unit (IMU) in order to obtain an accurate final attitude and position estimate of the satellite which can then be used for attitude and orbit control. This procedure is repeated for all upcoming image frames with the difference that the database can then be filtered using the accurate final attitude and position estimate. Additionally, for the next iterations ($n > 2$) the previous final estimate can be included in sensor fusion as well.
As already mentioned this approach requires much more computational power than only the original pipeline. It needs to be investigated to what extend the additional information of the relative pose estimate can improve the accuracy of the final estimate and thus if it makes sense to put in the additional effort.

Sensor Fusion

How the sensor fusion should be implemented in detail needs to be investigated in future studies as well. The most common approach is to use a Kalman filter for this task as already described by Redelbach [41]. Including an image-based attitude and position estimate in a Kalman filter is not uncommon. However, implementing the fusion of the absolute and relative estimate and of other measurements e.g. from the IMU is not trivial and some theoretical considerations need to be done in advance. These depend on the specific application and derivation of the image-based estimate and so no other implementation can directly be used. But there are some other image-based attitude and position estimation approaches that combined their estimates with other sensor

data which could be useful for designing an own approach. Two examples are the work presented in [9] and in [32]. Dagvasumberel and Asami [9] presents how their image-based attitude estimate and gyroscope data are fused by using an Unscented Kalman filter. Mourikis et al. [32] combined the IMU-data with their image-based estimates in an Extended Kalman filter. For details one can refer to their publications.

6.3 Future Investigations

Besides the improvements and extensions, there are also many general aspects that need to be investigated in the future in order to make AVS applicable for satellite missions. First, it needs to be analyzed how large the error of the EOS test images contributes to the total estimation error to be able to evaluate the accuracy of the attitude and position estimates quantitatively again. Other image datasets with the required camera parameters and pose information may be published in the future that could be used for testing the pipeline and for comparing and evaluating the current results.
As mentioned earlier, the desired scenario is that AVS is applicable during the entire orbit, and thus the database will contain many more features than during the performed tests. Furthermore, the scene on ground that is captured by the satellite's onboard camera changes significantly. During the tests within this work, all test images showed approximately the same area on ground. Among others, this might lead to a much higher runtime and to a higher probability of mismatches. So future studies need to investigate how the size of the database as well as the different characteristics of the captured scene influence the performance of AVS especially with respect to runtime and accuracy of the estimates.
Additionally, all test images created by the EOS simulated approximately nadir pointing of the satellite. However, AVS should not only be applicable when the satellite is already pointing with an elevation of 90 deg. It is therefore necessary to investigate how the elevation angle of the satellite impacts the performance of the pipeline.
Lastly, the impact of seasonal changes of the vegetation and therefore of the changing characteristics of the captured scene on the performance needs to be analyzed in the future. This can simply be done by using test images that have been taken at a different time of the year than the images used for creating the database.

Chapter 7

Conclusion

This internship was intended to test the general applicability of the approach of AVS and to provide a first configuration of the onboard pipeline as well as of the database creation as a basis for the design and concrete implementation of the ADCS in future missions. Based on the work of [41], a pipeline for the onboard image-based attitude and position estimation was implemented and the approach for creating the database was adapted. This version of the pipeline was significantly improved several times with respect to the accuracy of the final pose estimate, among others by developing a new feature validation algorithm. However, still some mismatches were forwarded to the final pose estimation. An idea for how this could be improved has not yet been found and must be investigated in future studies. Furthermore, it is not sure yet, to what extend the systematic error of the used test images created by the EOS contribute to the error of the pose estimations of the test runs. But, the currently resulting attitude and position estimates are too inaccurate to be useful for the ADCS of real satellite missions. Additionally, the runtime of the current embedded version of the pipeline was tested mainly using a *Raspberry Pi Compute Module 4*. It became obvious, that the runtime of the pipeline needs to be improved in order to enable real-time applicability by either adjusting the implementation of the feature detection and validation or by using another hardware device with more computational power than the *Compute Module 4*. Thus, a lot of work still needs to be done in order to enable the application of AVS for future satellite missions especially with respect to accuracy and runtime. Furthermore, ideas of how the approach could be extended to improve the performance in the future are presented as well. Thus, in summary, this work provides meaningful considerations and a first implementation of the entire pipeline as a basis for all upcoming studies.

Appendix A

General Appendices

A.1 Coefficients of the solution for the P3P-problem

In the following, the coefficients of the polynomial for computing the distances from the observed points to the projection center for the solution of the spatial resection are listed:

$$A_4 = \left(\frac{a^2 - c^2}{b^2} - 1\right)^2 - \frac{4c^2}{b^2}\cos^2\alpha \tag{A.1}$$

$$\begin{aligned} A_3 = 4\Big(&\frac{a^2 - c^2}{b^2}\left(1 - \frac{a^2 - c^2}{b^2}\right)\cos\beta + 2\frac{c^2}{b^2}\cos^2\alpha\cos\beta \\ &- \left(1 - \frac{a^2 + c^2}{b^2}\right)\cos\alpha\cos\gamma\Big) \end{aligned} \tag{A.2}$$

$$\begin{aligned} A_2 = 2\Big(&\left(\frac{a^2 - c^2}{b^2}\right)^2 + 2\cdot\left(\frac{a^2 - c^2}{b^2}\right)^2\cos^2\beta + 2\left(\frac{b^2 - c^2}{b^2}\right)\cos^2\alpha \\ &+ 2\left(\frac{b^2 - a^2}{b^2}\right)\cos^2\gamma - 4\left(\frac{a^2 + c^2}{b^2}\right)\cos\alpha\cos\beta\cos\gamma - 1\Big) \end{aligned} \tag{A.3}$$

$$\begin{aligned} A_1 = 4\Big(&-\left(\frac{a^2 - c^2}{b^2}\right)\left(1 + \frac{a^2 - c^2}{b^2}\right)\cos\beta \\ &+ \frac{2a^2}{b^2}\cos^2\gamma\cos\beta - \left(1 - \left(\frac{a^2 + c^2}{b^2}\right)\right)\cos\alpha\cos\gamma\Big) \end{aligned} \tag{A.4}$$

$$A_0 = \left(1 + \frac{a^2 - c^2}{b^2}\right)^2 - \frac{4a^2}{b^2}\cos^2\gamma \tag{A.5}$$

A.2 Landsat 8 Data

A.2.1 References of Used Landsat 8 Data

Each data package of the Landsat 8 data has a unique ID that includes all necessary information. It is structured as follows [49]:

LXSS_LLLL_PPPRRR_YYYYMMDD_yyyymmdd_CC_TX

- **L**: Landsat
- **X**: Sensor (C = Combined OLI/TIRS, T = TIRS-only (if Landsat 8 or higher), T = TM (if Landsat 4-5), O = OLI-only, E = ETM, M = MSS)
- **SS**: Satellite (09 = Landsat 9, 08 = Landsat 8, 07 = Landsat 7, ... 01 = Landsat 1)
- **LLLL**: Processing Correction Level (L1TP = precision and terrain, L1GT = systematic terrain, L1GS = systematic)
- **PPP**: WRS Path
- **RRR**: WRS Row
- **YYYYMMDD**: Acquisition Date expressed in Year, Month, Day
- **yyyymmdd**: Processing Date expressed in Year, Month, Day
- **CC**: Collection Number (02)
- **TX**: Collection Category (RT = Real Time, T1 = Tier-1, T2 = Tier-2)

Therefore, each data package that is used within this work is referenced in the following listing the corresponding IDs.

A.2.2 Data Used for Database

Table A.1 lists all the data used for creating the GCP database, that was used during all performed tests of chapter 5.

Description	Landsat 8 image data ID
Database Scene 1/1	*LC08_L1TP_195026_20200507_20200820_02_T1*
Database Scene 1/2	*LC08_L1TP_195026_20190419_20200828_02_T1*
Database Scene 2/1	*LC08_L1TP_195025_20200507_20200820_02_T1*
Database Scene 2/2	*LC08_L1TP_195025_20190419_20200828_02_T1*
Database Scene 3/1	*LC08_L1TP_195024_20200421_20200822_02_T1*
Database Scene 3/2	*LC08_L1TP_195024_20190419_20200828_02_T1*
Database Scene 4/1	*LC08_L1TP_195023_20200421_20200822_02_T1*
Database Scene 4/2	*LC08_L1TP_195023_20190419_20200828_02_T1*
Database Scene 5/1	*LC08_L1TP_196026_20200327_20200822_02_T1*
Database Scene 5/2	*LC08_L1TP_196026_20190629_20200827_02_T1*
Database Scene 6/1	*LC08_L1TP_196025_20220418_20220427_02_T1*
Database Scene 6/2	*LC08_L1TP_196025_20210602_20210608_02_T1*
Database Scene 7/1	*LC08_L1TP_196024_20220418_20220427_02_T1*
Database Scene 7/2	*LC08_L1TP_196024_20190629_20200827_02_T1*
Database Scene 8/1	*LC08_L1TP_196023_20220418_20220427_02_T1*
Database Scene 8/2	*LC08_L1TP_196023_20190629_20200827_02_T1*
Database Scene 9/1	*LC08_L1TP_194026_20220420_20220427_02_T1*
Database Scene 9/2	*LC08_L1TP_194026_20160318_20200907_02_T1*
Database Scene 10/1	*LC08_L1TP_194025_20220303_20220309_02_T1*
Database Scene 10/2	*LC08_L1TP_194025_20160318_20200907_02_T1*
Database Scene 11/1	*LC08_L1TP_194024_20220623_20220705_02_T1*
Database Scene 11/2	*LC08_L1TP_194024_20200601_20200825_02_T1*
Database Scene 12/1	*LC08_L1TP_194023_20220623_20220705_02_T1*
Database Scene 12/2	*LC08_L1TP_194023_20200601_20200824_02_T1*

TABLE A.1: Landsat 8 data references for the database.

A.3 Parameters of the Algorithms

Feature Detection, Description and Matching
An overview as well as the default and used values of the different adjustable parameters for the used feature detection and description algorithm ORB are listed in the following table A.2. For thresholding the matched features, a descriptor distance of $\alpha = 30$ is chosen. This value is taken from the work of [41].

Parameter	Description	Default value	Used value
Fast threshold	Fast threshold value used for keypoint detection	20	35
Nr. features	Maximum number of retained features	500	3500
Scale factor	Pyramid decimation ratio	1.2	1.2
Nr. levels	Number of pyramid levels	8	4
First level	Level of pyramid the source image is put to; Previous layers are filled with upscaled versions of the source image	0	0
Edge Threshold	Size of the border where the features are not detected	31	35
WTA_K	Number of points that produce each element of the oriented BRIEF descriptor	2	2
Score type	Algorithm used to rank features in order to retain best features	*HARRIS*	*HARRIS*
Patch size	Size of the patch used by the oriented BRIEF descriptor	31	35

TABLE A.2: Parameters for ORB [36].

Bibliography

[1] Alahi, Alexandre, Ortiz, Raphael, and Vandergheynst, Pierre. "Freak: Fast retina keypoint". In: *2012 IEEE conference on computer vision and pattern recognition*. Ieee. 2012, pp. 510–517.

[2] Alhwarin, Faraj. "Fast and robust image feature matching methods for computer vision applications". PhD thesis. Bremen, Universität Bremen, Diss., 2011, 2011.

[3] Banno, Atsuhiko. "A P3P problem solver representing all parameters as a linear combination". In: *Image and Vision Computing* 70 (2018), pp. 55–62.

[4] Bay, Herbert, Tuytelaars, Tinne, and Van Gool, Luc. "Surf: Speeded up robust features". In: *Lecture notes in computer science* 3951 (2006), pp. 404–417.

[5] Cai, Caixia. "6d visual servoing for industrial manipulators applied to human-robot interaction scenarios". In: (2017).

[6] Calonder, Michael et al. "Brief: Binary robust independent elementary features". In: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer. 2010, pp. 778–792.

[7] Campagna, Gianluca et al. "Please Spacecraft, know where you stand: Enabling visual navigation in LEO through DL-based Earth surface feature detection and matching." In: (2024).

[8] Chaumette, Francois and Hutchinson, Seth. "Visual servo control. I. Basic approaches". In: *IEEE Robotics Automation Magazine* 13.4 (2006), pp. 82–90.

[9] Dagvasumberel, Amartuvshin and Asami, Kenichi. "A Visual-Inertial attitude propagation for resource constrained small satellites". In: *Journal of Aeronautics and Space Technologies* 12.1 (2019), pp. 65–74.

[10] Dauner, Johannes et al. "Visual servoing for coordinated precise attitude control in the TOM small satellite formation". In: *Acta Astronautica* 202 (2022), pp. 760–771.

[11] Derpanis, Konstantinos G. "Overview of the RANSAC Algorithm". In: *Image Rochester NY* 4.1 (2010), pp. 2–3.

[12] Elsner, Lisa. "Development and Integration of Cooperative Vision-Based Attitude Control for the TOM Mission". MA thesis. University of Würzburg, 2022.

[13] Fischler, Martin A and Bolles, Robert C. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[14] Förstner, Wolfgang and Wrobel, Bernhard P. *Photogrammetric computer vision*. Springer, 2016.

[15] Friesen, Davis. "Development of an Earth Observation Simulator". MA thesis. University of Würzburg, 2023.

[16] Gao, Xiao-Shan et al. "Complete solution classification for the perspective-three-point problem". In: *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), pp. 930–943.

[17] *Google Earth Engine*. (online; accessed 1-August-2024).

[18] Grunert, Johann August. "Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodasie". In: *Grunerts Archiv fur Mathematik und Physik* (1841), pp. 238–248.

[19] Harris, Chris, Stephens, Mike, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[20] Hassaballah, Mahmoud, Abdelmgeid, Aly Amin, and Alshazly, Hammam A. "Image features detection, description and matching". In: *Image Feature Detectors and Descriptors: Foundations and Applications* (2016), pp. 11–45.

[21] Hladký, Maroš. "Vision Based Attitude Control". MA thesis. University of Würzburg, 2018.

[22] Hutchinson, S., Hager, G.D., and Corke, P.I. "A tutorial on visual servo control". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670.

[23] Johnson, Andrew E et al. "Implementation of a Map Relative Localization System for Planetary Landing". In: *Journal of Guidance, Control, and Dynamics* 46.4 (2023), pp. 618–637.

[24] Ke, Tong and Roumeliotis, Stergios I. "An efficient algebraic solution to the perspective-three-point problem". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7225–7233.

[25] Klančar, Gregor et al. "Image-based attitude control of a remote sensing satellite". In: *Journal of intelligent & robotic systems* 66 (2012), pp. 343–357.

[26] Kneip, Laurent, Scaramuzza, Davide, and Siegwart, Roland. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *CVPR 2011*. IEEE. 2011, pp. 2969–2976.

[27] Koizumi, Shoei et al. "Development of Attitude Sensor using Deep Learning". In: 2018.

[28] Kouyama, Toru et al. "Satellite attitude determination and map projection based on robust image matching". In: *Remote Sensing* 9.1 (2017), p. 90.

[29] Lepetit, Vincent, Moreno-Noguer, Francesc, and Fua, Pascal. "EPnP: An accurate O (n) solution to the PnP problem". In: *International journal of computer vision* 81 (2009), pp. 155–166.

[30] Leutenegger, Stefan, Chli, Margarita, and Siegwart, Roland Y. "BRISK: Binary robust invariant scalable keypoints". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2548–2555.

[31] Lourenço, João Ricardo. *Open-Elevation API*. (online; accessed 1-July-2024).

[32] Mourikis, Anastasios I et al. "Vision-aided inertial navigation for spacecraft entry, descent, and landing". In: *IEEE Transactions on Robotics* 25.2 (2009), pp. 264–280.

[33] Nakano, Gaku. "A Simple Direct Solution to the Perspective-Three-Point Problem." In: *BMVC*. 2019, p. 26.

[34] NASA. *A Royal View of Denmark*. (online; accessed 26-May-2023). 2017.
[35] NASA. *Landsat Science*. (online; accessed 12-July-2023).
[36] OpenCV. *ORB Class Reference*. (online; accessed 1-August-2023).
[37] *OpenTopoData*. (online; accessed 1-August-2024).
[38] *Orekit*. (online; accessed 1-July-2024).
[39] Persson, Mikael and Nordberg, Klas. "Lambda twist: An accurate fast robust perspective three point (p3p) solver". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 318–332.
[40] *PyProj*. (online; accessed 1-July-2024).
[41] Redelbach, Joshua. "Analysis of Feature Detection and Matching Algorithms for Absolute Visual Servoing". Bachelor's thesis. University of Würzburg, 2023.
[42] Rosten, Edward, Reitmayr, Gerhard, and Drummond, Tom. "Real-time video annotations for augmented reality". In: *Advances in Visual Computing: First International Symposium, ISVC 2005, Lake Tahoe, NV, USA, December 5-7, 2005. Proceedings 1*. Springer. 2005, pp. 294–302.
[43] Rublee, Ethan et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.
[44] SEO-Analyse. *Google Earth*. (online; accessed 13-August-2023).
[45] Tuytelaars, Tinne, Mikolajczyk, Krystian, et al. "Local invariant feature detectors: a survey". In: *Foundations and trends® in computer graphics and vision* 3.3 (2008), pp. 177–280.
[46] USGS, United States Geological Survey. *Landsat 8*. (online; accessed 12-July-2023).
[47] USGS, United States Geological Survey. *Landsat Collection-1*. (online; accessed 13-July-2023).
[48] USGS, United States Geological Survey. *Landsat Collection-2*. (online; accessed 13-July-2023).
[49] USGS, United States Geological Survey. *Landsat Collection 2 Data Dictionary*. (online; accessed 13-July-2023).
[50] USGS, United States Geological Survey. *Landsat Level-1*. (online; accessed 13-July-2023).
[51] USGS, United States Geological Survey. *Landsat Level-2*. (online; accessed 13-July-2023).
[52] USGS, United States Geological Survey. *Landsat Missions*. (online; accessed 12-July-2023).