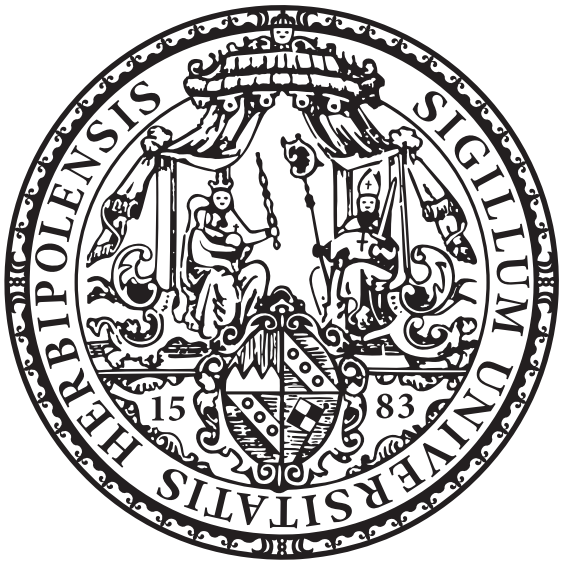
JULIUS-MAXIMILIANS-UNIVERSITÄT WÜRZBURG

IN COOPERATION WITH THE

ZENTRUM FÜR TELEMATIK E.V.



BACHELOR THESIS

Analysis of Feature Detection and Matching Algorithms for Absolute Visual Servoing

JOSHUA REDELBACH

August 30, 2023

Supervisor:
Prof. Dr. Marco Schmidt

Advisor:
M.Sc. Johannes Dauner
M.Sc. Lisa Elsner

Declaration of Authorship

I, Joshua Redelbach, declare that the work in this thesis 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 thesis work are those of the author.

Signed:

Date:

Abstract

One key application of small Low Earth Orbit (LEO) satellites is Earth observation using visual cameras onboard. To obtain high-quality image data, the satellite must be precisely aligned with the observation target during the overflight which leads to high requirements on the Attitude Determination and Control System (ADCS). Due to limited size and power of these small satellites, alternative approaches to the common used bulky and highly power consuming sensors such as star trackers are needed. One obvious approach is to exploit the image data from the payload camera for this purpose. Therefore, the Zentrum für Telematik e.V. (ZfT) develops an approach called Absolute Visual Servoing (AVS) that enables precise pointing of a satellite towards a specific location on Earth by determining the absolute attitude and position based on the image data of an onboard camera. Two important procedures during AVS are the feature detection, description and matching performed onboard the satellite, as well as the creation of a database containing Ground Control Points (GCPs) in advance. Within this thesis, state-of-the-art algorithms for feature detection, description and matching are analyzed and the applicability of these processes regarding AVS is investigated. It is shown that there are big differences in the performance of the different algorithms regarding AVS and that several combinations offer certain advantages. When the results are considered in total, the ORB detector combined with the ORB descriptor is suggested as the best combination. Based on the results of this thesis, the most suitable combination of algorithms for a specific application of AVS can be selected. Furthermore, many problems that limit the applicability of AVS have arisen during the tests, too. Therefore, it is also presented what needs to be investigated in future work, and how to adapt the application of AVS due to the limitation.

Acknowledgements

First of all, I would like to express my appreciation to my examiner Prof. Dr. Marco Schmidt, who gave me the opportunity to write my thesis at the Zentrum für Telematik e.V. (ZfT).

Furthermore, I am very grateful to my supervisors, especially to Johannes Dauner, for giving me the chance to work on such an interesting topic, and for guiding and supporting me throughout the work on this thesis.

Finally, I would like to thank my family and friends for the support and constant encouragement throughout my thesis.

Thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

VS	Visual Servoing
LEO	Low Earth Orbit
ADCS	Attitude Determination and Control System
GCP	Ground Control Point
ZfT	Zentrum für Telematik e.V.
TOM	Telematics Earth Observation Mission
RVS	Relative Visual Servoing
AVS	Absolute Visual Servoing
ECEF	Earth Centered Earth Fixed
ECI	Earth Centered Inertial
FOV	Field of View
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Features
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
ORB	Oriented FAST and Rotated BRIEF
FREAK	Fast Retina Keypoint
BRISK	Binary Robust Invariant Scalable Keypoints
LoG	Laplacian of Gaussian
DoG	Difference of Gaussian
AT	Absolute Thresholding
TNN	Thresholded Nearest Neighbor
NNDR	Nearest Neighbor Distance Ratio
ANN	Approximate Nearest Neighbor
RANSAC	Random Sample Consensus
NASA	National Aeronautics and Space Administration
USGS	United States Geological Survey
WRS-2	Worldwide Reference System-2
UTM	Universal Transverse Mercator
WGS-84	World Geodetic System-84
DN	Digital Numbers
TOA	Top of Atmosphere
SR	Surface Reflectance
OpenCV	Open Source Computer Vision Library

Chapter 1

Introduction

1.1 Motivation

In recent years, the relevance of small Low Earth Orbit (LEO) satellites in the space market has significantly grown, especially of pico, nano and micro satellites weighing less than 1, 10 and 100 kg respectively. These smaller satellites offer several advantages over their larger and more complex counterparts. They are more cost-effective to construct and launch into Earth's orbit, and their development time is considerably shorter, ranging from 1 to 2 years, in comparison to highly capable, big and complex satellites. These developments also enable smaller entities such as universities to acquire space technology, eliminating exclusivity to big corporations [39]. One key function of these small satellites is Earth observation, primarily by utilizing visual cameras. In order to obtain high-quality image data during satellite missions for Earth observation applications, the satellite must be very precisely aligned with the observation target during the overflight. Therefore, these missions make high demands on the Attitude Determination and Control System (ADCS). The task of precise attitude determination is usually performed by sensors such as sun sensors, magnetometers, gyroscopes and star trackers [60]. Due to the limited size and power in small satellites, alternative approaches to bulky and highly power consuming sensors are required. An obvious approach for Earth observation missions is to exploit the image data from the payload camera for this purpose. Even these small onboard cameras are now able to capture high-resolution pictures thanks to the rapid development of imaging technology. Furthermore, the current microprocessor technology enables the computationally heavy analysis of these images in real-time. In this way, space and mass could be saved compared to using conventional sensors, and additional energy would only be required for data processing.

The approach of determining the attitude of a satellite based on image data provided by an onboard camera has already been investigated in multiple studies (see section 2.1). The derived attitude from the image data can then be used to calculate the control input for attitude-controlling actuators such as reaction wheels. Such vision-based control concepts are also known as Visual Servoing (VS). The Telematics Earth Observation Mission (TOM) of the Zentrum für Telematik e.V. (ZfT) is a satellite formation mission, whose aim is to observe ash clouds during a volcanic eruption. In order to achieve the joint tracking of the same target area by three satellites and thus to meet the already

mentioned high requirements on the ADCS, a Relative Visual Servoing (RVS) approach was developed [11, 14]. This concept of RVS is now enhanced to Absolute Visual Servoing (AVS). AVS enables precise pointing of a single satellite towards a specific location on Earth's surface by determining the absolute attitude and position based on the image data of an onboard camera (more details in section 2.2).

1.2 Scope of Thesis

An important procedure during AVS is the detection of features in onboard satellite images, the description of these features and matching the detected features with Ground Control Point (GCP)s stored in an onboard database created in advance. There exist several algorithms and methods to implement each of the three parts as well as to create the database. To meet the high requirements for AVS, e.g. real-time capability and memory consumption of the database, the main objective of this thesis is to analyze the processes of feature detection, description and matching during the creation of the database as well as during the onboard matching and to compare state-of-the-art algorithms in this area. This thesis thus intends to show the applicability of the process of feature detection, description and matching regarding AVS and to provide one possible configuration of algorithms as a basis for the design and concrete implementation of the ADCS in future missions.

1.3 Approach

In order to achieve this objective, a selection of promising state-of-the-art algorithms for feature detection, description and matching is investigated. To enable a meaningful comparison between the different algorithms for each part and to find a well performing configuration of these algorithms regarding AVS, a test application is implemented. This test application comprises the creation of the database as well as the simulation of the onboard matching. To create the database, a research for suitable satellite image data that enables the extraction of GCPs is done. In addition, these satellite images then are also used to simulate the process of onboard feature detection, description and matching performed by the satellite by using the different algorithms. To find the best configuration of the algorithms the performance of this process is analyzed based on different metrics.

1.4 Structure of Thesis

The thesis is structured as follows: Chapter 2 presents the theoretical background. At first, the state of the art of visual servoing onboard satellites is described and the concept of AVS is introduced in detail. Afterwards, the concepts of feature detection, description, matching and validation as well as concrete algorithms are introduced. Subsequently, GCPs in the traditional sense and in the context of AVS are described. Chapter 3 focuses on the test application. It presents the used satellite image data as well

as the concept and setup of the test application. The different test runs and the corresponding experimental results are described and evaluated in chapter 4. In addition, constraints of the approach of AVS as well as aspects that needs to be investigated in future work are addressed. Afterwards, an outlook is given on how AVS could be applied in the future. Finally, the thesis is concluded in chapter 5 by a summary of the contributions.

Chapter 2

Theoretical Background

The following chapter provides the necessary background knowledge to fully understand, first, the motivation and conceptual idea of AVS, second, the concept of feature detection, description, matching, as well as validation, and third, the meaning of GCPs in the context of AVS.

2.1 State of the Art of Visual Servoing Onboard Satellites

VS is a method to control the motion of a robot based on image data. The technique originally came from the field of robotics and was first introduced in the early 1980s. The primary objective of VS is to minimize the difference between the current and the desired state, which is determined through image measurements, to perform positioning or tracking tasks [9, 21, 7, 18]. To adapt the concept of VS to control the attitude of a satellite, the current attitude of the spacecraft has to be derived from the image data of an onboard camera. Based on the difference between the current and desired attitude, the control input is calculated for the actuators such as reaction wheels.
Determining the attitude of a satellite based on image data of an onboard camera has already been addressed in several studies. A few of them are described in the following, but many more can be found in literature. The approach of Koizumi et al. [24] initially detects the edge of the Earth in the captured image and determines the 2-axis attitude (nadir direction) based on it. Subsequently, the 3-axis attitude determination is accomplished by matching land patterns using deep learning. Due to the computationally intense template matching, a real-time application is questionable. In addition, a catalog of earth images is needed onboard the satellite, which requires large memory space. Dagvasumberel and Asami [10] use feature detection and matching algorithms to find same patterns in successive images and to estimate the relative attitude afterwards. The study propagates the relative attitude with gyroscope measurements in an unscented Kalman filter and thus depends on the accuracy and drift of the gyroscope. In the study of Kouyama et al. [25], the attitude of the satellite is derived from its observation images and known satellite position by matching land features from an observed image and from well-registered base-map images. As well as in the study of Koizumi et al. [24], reference images with known geographical information needs to be stored onboard the satellite, requiring large memory space.

As mentioned above, the derived attitude from the image data can then be used to calculate the control input. The concept of VS has been adapted to the application of satellites, i.e. by Hladký [20], Klančar et al. [23], Dauner et al. [11], and Elsner [14]. The studies of Dauner et al. [11] and Elsner [14] have been conducted within TOM. TOM is a satellite formation mission of the ZfT, whose objective is to observe ash clouds during a volcanic eruption. In order to fulfill the high demands on the ADCS, a RVS approach was developed to achieve the joint tracking of the same target area by three satellites. To implement this concept highly distinctive features are detected in the images of each satellite and the control input is then calculated from the movement of the features between consecutive images. In order to achieve the joint tracking of a target area by the different satellites, the detected features and their positions in the image are exchanged between the satellites. This procedure enables the alignment of the satellites relative to the target object. It is important to mention that RVS only works if the images of the different satellites overlap at least to some extent at the beginning. To ensure this, the satellites must first be aligned to an accuracy of about 2 deg using conventional attitude sensors such as sun sensors or magnetometers.

2.2 Absolute Visual Servoing

To extend the applicability of image-based attitude control algorithms to further Earth observation missions, the relative approach is enhanced to an absolute one. This should enable the precise pointing of a single satellite towards a specific location on Earth's surface by determining the absolute attitude and position of the satellite based on images taken onboard. In AVS, features are detected in the incoming new image frames, which are than be compared and matched with known features. These known features, so called GCPs, are structures on the surface of the Earth with unique appearance in satellite imagery which are stored in an onboard database mapped with their absolute position on Earth. The position of these GCPs on ground are indicated with respect to the Earth Centered Earth Fixed (ECEF) coordinate system. Based on the appearance and pixel position of these GCPs in the current image, the absolute attitude and position of the satellite with respect to the Earth Centered Inertial (ECI) coordinate system

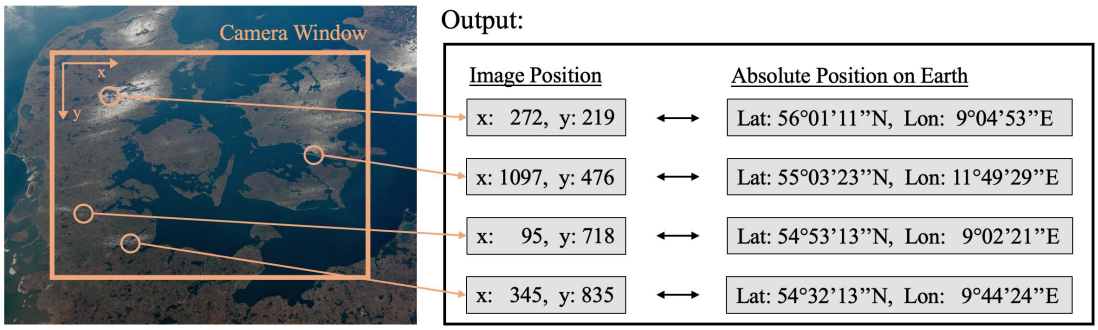


FIGURE 2.1: Illustration of a scenario in which the image position (x- and y-coordinates) of detected features are successfully matched with the absolute position on Earth (latitude and longitude in ECEF frame) of known features. Image credit: NASA [32]

is determined. Subsequently, it is used to calculate the control input to achieve the required precise pointing towards the desired target. The desired scenario, in which the image position of detected features are successfully matched with their absolute position on Earth, is illustrated in fig. 2.1. Compared to RVS, this approach not only has the advantage that a single satellite can be precisely aligned to a target, but also that it is not necessary to point the satellite with a certain accuracy in advance.
In conclusion, the main goal of AVS is to enable a reliable position and attitude determination during the entire orbit by detecting and matching a sufficient number of features with the satellite's camera that are stored in an onboard database. Therefore, certain requirements must be met. First, the GCP features of the database have to be equally distributed around the globe, so that it is guaranteed that the onboard camera can detect a minimum number of features within its Field of View (FOV). Second, in order to be able to determine the exact position and attitude, the matched features must not contain any false matches. To ensure a reliable feature matching, the GCP features must have a unique appearance in satellite images and have to be described with a unique signature, so that these features are detected and matched correctly onboard, even though a different camera as the one to create the database is used.
This thesis focuses on the process of creating the GCP database and matching the features detected in the onboard image with the features stored in the database. The exact process of how the attitude and position is determined afterwards is not investigated in detail. However, the procedure for this is briefly described in the following section.

2.3 Attitude and Position Determination

In general, determining the attitude and position of a satellite means to compute how the satellite's coordinate system (body-frame; $\vec{x}^B, \vec{y}^B, \vec{z}^B$) is shifted ($\vec{O}^{I\to B}$) and rotated ($\theta^{I\to B}, \phi^{I\to B}, \psi^{I\to B}$) with respect to an inertial reference frame ($\vec{x}^I, \vec{y}^I, \vec{z}^I$), in this case the ECI frame. In order to calculate these six parameters ($\vec{O} = [x, y, z]^T, \theta, \phi, \psi)^{I\to B}$ the following three steps need to be performed:

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

Figure 2.2 illustrates the different coordinate frames referred to in this section. The position and attitude of the onboard camera with respect to the ECEF frame can be computed using the collinearity equations:

$$x^C = -k\frac{(x^E - x^{E\to C})\cdot r_{11} + (y^E - y^{E\to C})\cdot r_{12} + (z^E - z^{E\to C})\cdot r_{13}}{(x^E - x^{E\to C})\cdot r_{31} + (y^E - y^{E\to C})\cdot r_{32} + (z^E - z^{E\to C})\cdot r_{33}}\,, \tag{2.1}$$

$$y^C = -k\frac{(x^E - x^{E\to C})\cdot r_{21} + (y^E - y^{E\to C})\cdot r_{22} + (z^E - z^{E\to C})\cdot r_{23}}{(x^E - x^{E\to C})\cdot r_{31} + (y^E - y^{E\to C})\cdot r_{32} + (z^E - z^{E\to C})\cdot r_{33}} \tag{2.2}$$

with

$$\boldsymbol{R}^{E\to C} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \tag{2.3}$$

$$\begin{bmatrix} \cos\phi\cos\psi & -\cos\phi\sin\psi & \sin\phi \\ \cos\theta\sin\psi + \sin\theta\sin\phi\cos\psi & \cos\theta\cos\psi - \sin\theta\sin\phi\sin\psi & -\sin\theta\cos\phi \\ \sin\theta\sin\psi - \cos\theta\sin\phi\cos\psi & \sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi & \cos\theta\cos\phi \end{bmatrix}, \tag{2.4}$$

where k is a constant regarding the used camera. These equations link the pixel positions (camera-frame; x^C,y^C) of the matched GCP features with their ground position (ECEF frame; x^E, y^E, z^E). The components of the vector $[x^{E\to C}, y^{E\to C}, z^{E\to C}]^T$ represents the shift of the camera-frame with respect to the ECEF frame. For a detailed deviation of these equation, one refer to [47]. To find the attitude and position of the camera $(x, y, z, \theta, \phi, \psi)^{E\to C}$, for each onboard matched feature the two collinearity equations are set up. Six equations are required to be able to compute all six parameters, leading to the need of at least three matched features. Therefore, in the rest of this thesis, a sufficient number of matches is meant to be greater than or equal to 3. If more than 3 features are given, the system is overdetermined and an approximate, best solution can be computed which results in a more accurate attitude and position.
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 which is a routine task for the ADCS and thus is not described in detail within this thesis.

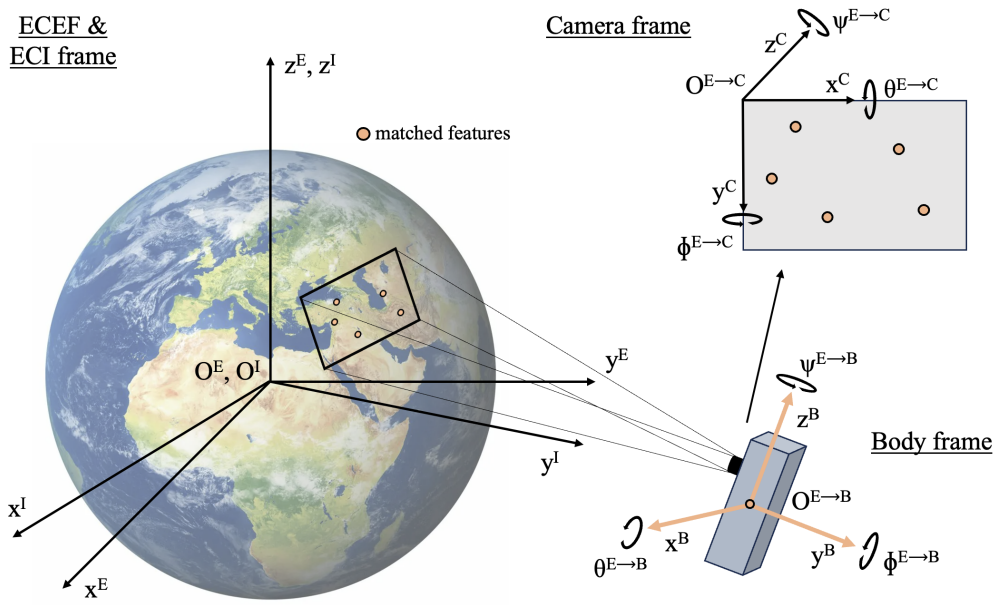


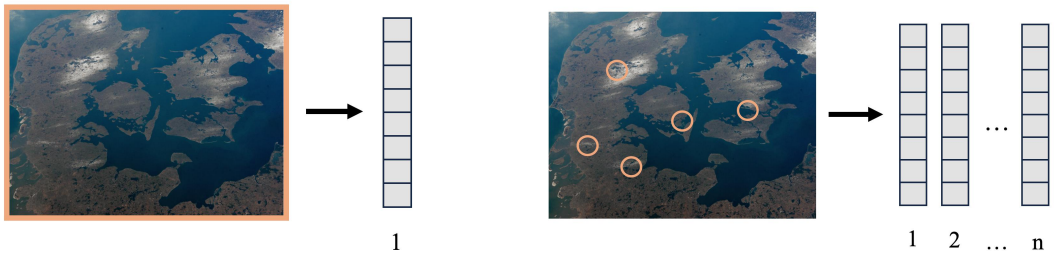
FIGURE 2.2: Illustration of the different reference frames used to determine the attitude and position of the satellite based on the onboard matched features. Vector arrows are omitted. Credits for image of Earth: SEO-Analyse [48]

2.4 Features

As the detected features form the basis of the control law, their selection influences the performance of AVS significantly. Therefore, it is essential to first clarify what a feature is and how it is defined. Unfortunately, there is no exact definition of features. Generally, there are two types of image representation: global and local features [19]. By using global features, a single multi-dimensional feature vector is created that describes all pixels of the image as a whole. This vector represents information about various attributes of the image, such as color, texture, and shape. The primary objective of local feature representation is to describe the image by a set of feature vectors, which captures the local structures of the image. Each vector represents a specific region of interest in the image, called keypoint (see fig. 2.3). These feature vectors are also named descriptors. Which type of image representation should be used depends on the application. For visual servoing in satellite applications, local features are used because they are better suited to detect different instances of the same object in multiple images. Furthermore, they are more distinctive in smooth regions and are real-time applicable [19, 14].
Local features can be considered as image patterns which differ from its immediate neighborhood [49]. It commonly refers to a highly distinctive point or an unique piece of data in an image that can be detected repeatedly in different images showing the same scenario. To ensure that, a feature should have the following properties [19]:

- **Robustness**: The same feature locations should be detected independent of scaling, rotation, shifting, photometric deformations, compression artifacts and noise.
- **Repeatability**: The same features of the same scene or object should be detected repeatedly under different viewing conditions.
- **Accuracy**: The features should be localized accurately, especially for image matching tasks, where precise correspondence is required.
- **Efficiency**: The features should be detected quickly to support real-time applications.

In summary, features can represent different parts or properties in an image, such as different shapes, segments, textures, or regions of interest. In this work, a feature is



(A) Global feature representation. (B) Local feature representation.

FIGURE 2.3: Types of feature representation [32, 19].

defined as a small unique region in the image represented by a pixel location and a descriptor vector. As already mentioned, the performance of AVS significantly depends on the selection of GCP features for the database as well as on the detection, description, matching and validation of the features onboard the satellite. Thus, the upcoming sections 2.5, 2.6, 2.8 and 2.9 focus on these concepts.

2.5 Feature Detection

The main objective of a feature detection algorithm is to determine local features in an image. Therefore, the algorithm searches for specific regions of interest and identifies their exact position in the image. In order to locate these interest regions, the algorithms usually search for areas with significant structural information and specific shape, such as edges, corners or blobs. This is done by defining a specific salience measure and looking for local extrema across the image pixels [3].
One of the earliest local feature detector algorithms and most common corner detector algorithms was the Harris Corner Detector [17], which was an improvement of the one developed by Moravec [30]. The basic concept of the corner detector of Moravec [30] as well as of the improvement of Harris, Stephens, et al. [17] is as follows: A local window around a pixel in the image is shifted by small amount (u, v) in different directions and the average changes of image intensity $E(u, v)$ is computed. Then three cases can occur:

- A smooth and flat region is detected if the value of $E(u, v)$ is small.
- An edge is detected if $E(u, v)$ changes only in one direction.
- A corner is detected if $E(u, v)$ greatly changes in all directions.

The mathematical form of this idea was defined by Harris, Stephens, et al. [17] in the local auto-correlation function, that can be interpreted as the cornerness value:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x,y) - I(x+u, y+v)]^2. \tag{2.5}$$

The image intensity $I(x, y)$ is computed based on the gray-level values of the pixels. The window function $w(x, y)$ specifies the local image window. It is either a rectangular window or a Gaussian window which gives weights to pixels underneath. All possible shifts are obtained by performing an analytic expansion about the shift origin in order to provide isotropy. Therefore, $I(x + u, y + u)$ can be approximated by a Taylor expansion:

$$I(x+u, y+v) \approx I(x,y) + I_x \cdot u + I_y \cdot v, \tag{2.6}$$

where $I_x = \frac{\partial I(x,y)}{\partial x}$ and $I_y = \frac{\partial I(x,y)}{\partial y}$ are the partial derivatives of $I(x, y)$. Combining equation 2.5 and 2.6 results in:

$$E(u,v) = \sum_{x,y} w(x,y)[I_x(x,y) \cdot u - I_y(x,y) \cdot v]^2. \tag{2.7}$$

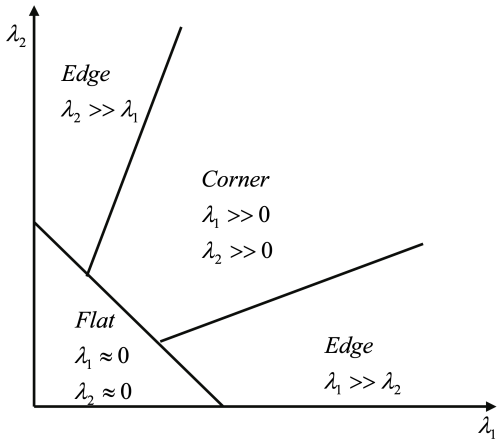


FIGURE 2.4: The Harris Corner Detector: Interpretation of the eigenvalues [3].

Equation 2.7 can be rewritten in matrix form:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \cdot \boldsymbol{M} \cdot \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.8}$$

where M is a symmetric matrix and defined as:

$$\boldsymbol{M} = \sum_{x,y} w(x,y) \cdot \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_x^2 \end{bmatrix}. \tag{2.9}$$

As mentioned earlier, $E(u,v)$ is closely related to the local auto-correlation function, with $\boldsymbol{M}$ describing its shape at the origin. Therefore, the eigenvalues λ_1 and λ_2 of $\boldsymbol{M}$ are proportional to the principal curvatures of the local auto-correlation function and form a rotational invariant description of $\boldsymbol{M}$. As above, three cases can be considered (see fig. 2.4):

- If both eigenvalues and thus both curvatures are small, so that the local auto-correlation function $E(u,v)$ is flat and has only little changes in any direction, then the windowed image has approximately a constant intensity distribution. The region is smooth and flat.
- If only one eigenvalue and thus one curvature is high, then the $E(u,v)$ is ridge shaped and has significant changes only in one direction. The region contains an edge.
- If both eigenvalues and thus both curvatures are high, so $E(u,v)$ is sharply peaked and has large changes in any direction. The region contains a corner.

To avoid the computationally expensive determination of the eigenvalues λ_1 and λ_2, Harris, Stephens, et al. [17] suggest to compute the response R with the following equations:

$$R = det(\boldsymbol{M}) - \kappa \cdot (trace(\boldsymbol{M}))^2, \tag{2.10}$$

$$det(\boldsymbol{M}) = \lambda_1 \cdot \lambda_2, \tag{2.11}$$

$$trace(\boldsymbol{M}) = \lambda_1 + \lambda_2. \tag{2.12}$$

The interpretation of R can be read in Moravec [30]. The parameter κ needs to be determined empirically, but in the literature values in the range of $0.04 - 0.15$ are commonly used [3]. The simple concept of the Harris Corner Detector provides a good entry point to understand the approach of feature detectors. In section 2.7 more feature detection algorithms are described which offer various advantages compared to the Harris Corner Detector and are thus more suitable for the application of AVS.

2.6 Feature Description

After detecting local features in the image, these keypoints have to be described quantitatively in a feature descriptor. This is done by analyzing the neighborhood around a keypoint located at (x, y) in an image. That neighborhood contains a certain amount of adjacent pixels. Feature descriptors encode interesting information of these pixels into a vector and act as a numerical fingerprint that can be used to differentiate as well as to match two features. These feature vectors need to be robust to noise and to changes in viewpoint and in photometric imaging conditions [3]. In addition, they have to be distinctive while at the same time the length of the vector is limited. This is because otherwise the description and the matching process would not be real-time applicable. Over the last years, a lot of feature descriptor methods have been proposed. They can be divided into three main groups depending on the analyzed characteristics of the neighboring pixels: texture, shape and color descriptors [3]. All of the used algorithms are based on texture description. The term texture in this context refers to visual characteristics of surfaces, such as their roughness or smoothness. Accurately determining texture descriptors provides a powerful tool for similarity matching. Several specific feature description algorithms are presented in more detail in section 2.7.

2.7 Algorithms

In the following, concrete feature detection and description algorithms are presented. From these algorithms, a selection is then made for the upcoming test application in chapter 3. The main focus of this thesis is not on the exact functionality of all these algorithms but on testing their applicability regarding AVS. Therefore, each algorithm is summarized shortly. Further details can be found in the corresponding publications referenced in the respective section.

2.7.1 SIFT

In section 2.5 the Harris Corner Detector is introduced. One key advantage of the detector is its rotation invariance. This implies that the algorithm is able to identify identical corners across different images, even if one of the images is rotated. Nevertheless, it should be noted that the features detected by the Harris Corner Detector are not invariant to changes in scale. This means that a corner present in a low-scaled image may not be recognized as a corner in a high-resolution image (see fig. 2.5). However, in many

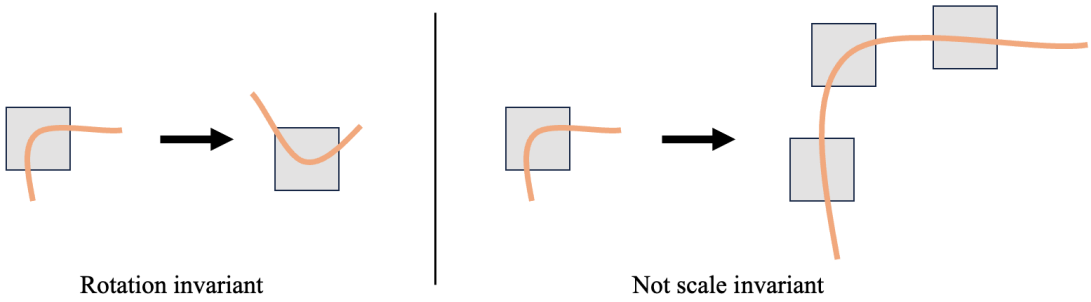


FIGURE 2.5: Dependency of a corner detector on rotation (left) and scale (right).

applications it is mandatory to detect the same features in different scaled images of the same scenario. Therefore, Lowe [27] developed a scale-invariant feature detection and description algorithm, called Scale Invariant Feature Transform (SIFT), which is widely used in computer vision. The algorithm consists of four steps:

1. **Scale-space extrema detection**: Finding scale invariant keypoints can be done by searching for stable features across all possible scales. Therefore, the scale-space of the image is created. It consists of different octaves, where the size of the image is down-sampled by a factor of 2. Each octave includes a series of smoothed images. This smoothing process is accomplished by applying Gaussian blurring at different values of σ, which is mathematically referred to as the convolution of the Gaussian operator and the image $I(x, y)$:

$$K(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.13}$$

where $*$ is the convolution operation in (x, y) and G is the Gaussian operator:

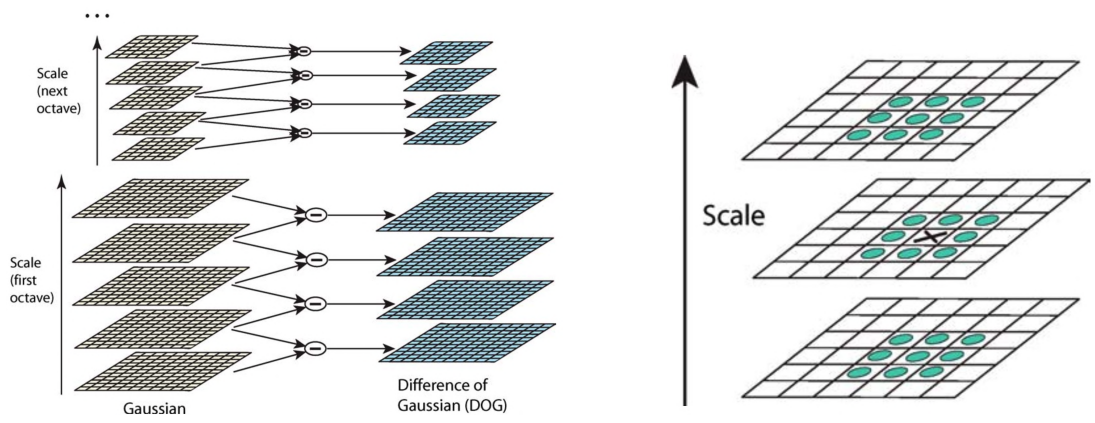
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \cdot e^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{2.14}$$

Mikolajczyk [29] pointed out, that the maxima and minima of the scale-normalized Laplacian of Gaussian (LoG) $\sigma^2\nabla^2 G$ produce the most stable image features compared to a range of others, such as gradients or corners. The LoG can be approximated as:

$$\sigma\nabla^2 G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \Rightarrow G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{2.15}$$

Therefore, the LoG can be estimated by the Difference of Gaussian (DoG) and can be computed by subtracting adjacent Gaussian blurred images of the scale-space (see fig. 2.6 (A)). In order to detect potential keypoints in the DoG images, each pixel is compared with its eight neighbors as well as with the nine pixels in the previous and next scale. If the value of the pixel is the highest or the lowest among its neighbors, it is a local extrema and thus a potential keypoint (see fig. 2.6 (B)).

2. **Keypoint localization**: To improve the localization accuracy the algorithm performs a detailed localization step. They use the method of Brown and Lowe [6]



(A) Creating the scale-space. (B) Finding extrema.

FIGURE 2.6: Illustration of the processes of creating the scale-space and finding extrema in the scale-space during SIFT [27].

for fitting a 3D quadratic function to the local sample points to determine their interpolated precise location and scale. In order to eliminate unstable keypoints, candidates with low contrast or poorly defined edges are discarded to ensure that only robust and distinctive features are retained.

3. **Orientation assignment**: SIFT assigns a dominant orientation to each keypoint to achieve invariance to rotation. Therefore, a local histogram with 36 bins covering 360 deg of gradient orientations is created based on the neighboring pixels around each keypoint, and the peaks in the histogram represent the dominant orientations. The keypoint is then assigned to one or more orientations based on these peaks.

4. **Keypoint descriptor**: In this step, a descriptor is generated for each keypoint to describe its appearance and capture its local neighborhood information. The descriptor is computed based on the gradient magnitude and orientation within a region around the keypoint. This region is first rotated through the orientation angle computed in the step before to provide rotation invariance. Then, an area around the keypoint is selected and divided into 16 square sub-regions. The size of the area is related to the keypoint's scale. Based on the pixel within each sub-region, an 8-bin orientation histogram is calculated. All these 16 8-bin orientation histograms are transformed into a 128D vector. In order to achieve invariance to illumination changes, the vector is normalized. Each of the 128D feature vectors results in a floating point vector which requires 512 byte of storage.

2.7.2 SURF

In order to enable a robust as well as fast feature detection and description process that is suitable for real-time applications, Bay, Tuytelaars, and Van Gool [4] developed the Speeded-Up Robust Features (SURF) algorithm. It is based on the SIFT algorithm with a few modifications to speed up the process. Similar to the SIFT algorithm, SURF detects keypoints at different scales in order to obtain scale-invariant features. Therefore, the

feature detector relies on a scale invariant blob detector based on the determinant of the Hessian matrix

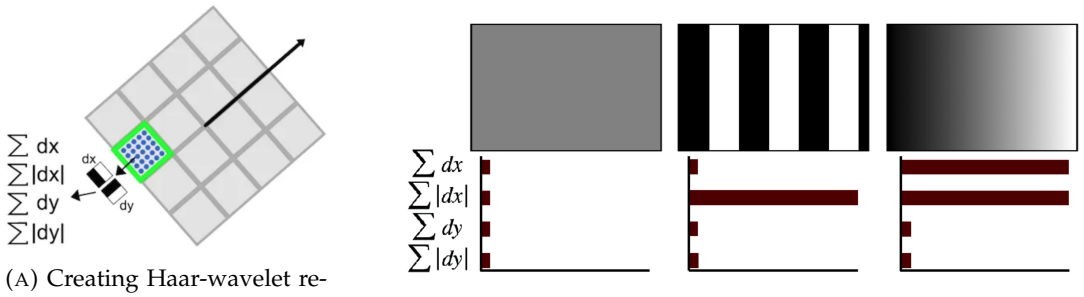
$$\boldsymbol{H}(p,\sigma) = \begin{bmatrix} G_{xx}(x,y,\sigma) & G_{xy}(x,y,\sigma) \\ G_{xy}(x,y,\sigma) & G_{yy}(x,y,\sigma) \end{bmatrix}, \tag{2.16}$$

where (x,y) is the location of the current pixel p in the image I, σ identifies the scale and $G_{xx}(x,y,\sigma)$, $G_{xy}(x,y,\sigma)$ and $G_{yy}x(x,y,\sigma)$ are the convolutions of the image with the second derivatives of the Gaussian function (also referred to as LoG). The entries of the Hessian matrix can be interpreted as the image intensity slope of the pixel located at p in horizontal (G_{xx}), vertical (G_{yy}) and diagonal (G_{xy}) direction. All directions are combined in the determinant of the Hessian matrix which expresses therefore the local intensity change around the pixel p in a single value. Pixels with the highest determinant are most contrasting and are thus chosen as keypoint candidates [4, 14]. To speed up the computational expensive calculation of the second derivatives of the Gaussian function, the entries of the Hessian matrix are approximated by box filters. The convolution with the box filter is efficiently implemented using integral images which saves a high amount of computational cost. The approximation is denoted as follows:

$$\boldsymbol{H}_{approx}(p,\sigma) = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad \Rightarrow \quad det(\boldsymbol{H}_{approx}) = D_{xx}D_{yy} - (0.9 \cdot D_{xy})^2 \tag{2.17}$$

In order to achieve scale invariance, keypoints need to be extracted at different scales. But instead of resizing the image to get various versions of the same image at different scales and apply the same box filters to these images, SURF uses different sizes of the box filters to the original image in parallel. By this procedure computational effort is saved. Afterwards, a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied to localize interest points in the image and over scales.

After finding the keypoints, a descriptor vector is determined that describes the neighboring pixels of each keypoint. Similar to the SIFT algorithm, first the orientation of the point of interest is calculated to achieve rotational invariance. Therefore, the Haar-wavelet responses in both x- and y-directions in a circular neighborhood around the keypoint are computed. The Haar-wavelet response is a simple rectangular filter that efficiently captures horizontal and vertical intensity differences within the analyzed sub-region. By using again integral images, only six operations are needed to compute the response in x- or y-direction at any scale [4]. Then, the horizontal and vertical wavelet responses within each scanning area of $\frac{\pi}{3}$ are summed up, which yields in six new vectors. The one with the highest magnitude represents the orientation of the keypoint. This orientation is then used to extract the feature descriptor vector. Therefore, a square region centered around the keypoint and aligned along its orientation is constructed. This area is split up into 16 smaller 4×4 square sub-regions. For each sub-region the Haar-wavelet responses in x- and y-direction, called d_x and d_y respectively, are summed up and form a first set of entries of the descriptor vector. In order to get information about the polarity of the intensity changes, the sum of the absolute values of the responses, $|d_x|$ and $|d_x|$, are extracted as well. Thus, each of the 16 sub-regions provides a 4D vector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ which results in a 64D descriptor



(A) Creating Haar-wavelet responses for each sub-region [28].

(B) Quantitative examples of generating the 4D vector, extracted during the SURF feature description, on sample images [4].

FIGURE 2.7: Illustration of the SURF feature description process.

vector (see fig. 2.7) that requires 256 byte of storage. After turning the descriptor vector into a unit vector, SURF ensures in addition to invariance to rotation and scaling also invariance to contrast and illumination.

2.7.3 FAST

Even though SURF and other feature detectors show high performance regarding real-time applicability, some applications, such as SLAM (Simultaneous Localization and Mapping), require an even faster feature detection process. Therefore, Rosten and Drummond [44] developed the Features from Accelerated Segment Test (FAST) algorithm which is a corner detection approach.

In order to find a corner candidate a segment test is performed for each pixel. Let p be the pixel to be analyzed, $I(p)$ its intensity and t a threshold value that is selected at the beginning. The segment test classifies a pixel p as a corner, if at least n contiguous pixels along a circle of 16 pixels around p (see fig. 2.8) are brighter than $I(p) + t$ or darker than $I(p) - t$. In order to speed up the algorithm the original detector [43, 45] first performs a high-speed test. It examines only the four pixels at location 1, 5, 9 and 13 (see fig. 2.8). The pixel p can only be a corner if at least three of these pixels are brighter than $I(p) + t$ or darker than $I(p) - t$. To enable this high-speed test n needs to be greater than or equal to 12. In the paper of Rosten and Drummond [44] $n = 12$ was chosen. If the high-speed test is passed the full segment test is applied to the pixel

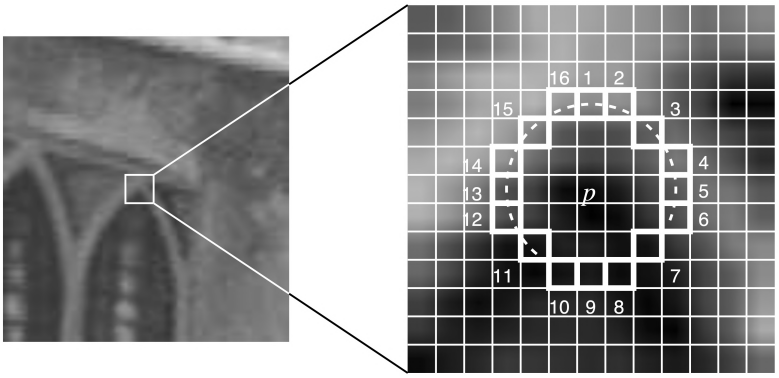


FIGURE 2.8: Illustration of the analyzed neighborhood of the considered pixel p during FAST [44].

p. Even though the original approach achieves high performance, there exist several limitations:

1. The high-speed test does not generalize well for $n < 12$.
2. The speed of the algorithm depends on the order in which the 16 pixels of the circle are queried.
3. The knowledge of the high-speed test is discarded afterwards.
4. Multiple interest points are detected in adjacent locations.

To circumvent these difficulties the original FAST corner detector was revisited by adding two concepts [44]. The first three points are addressed with a machine learning approach. Last one is addressed using non-maximal suppression.

- **Machine learning approach**: The ID3 algorithm [40], which is a decision tree classifier, is trained with a set of various images. The exact procedure can be studied in detail in [44]. The resulting decision tree is used to query the adjacent pixel more efficiently.
- **Non-maximal suppression**: In order to solve the problem of detecting multiple interest points in adjacent locations, a score function V is computed for all detected feature points. To compute the score function the sum of absolute differences between $I(p)$ and the intensity of the pixels q of the circle that are brighter (S_{bright}) as well as for the pixels of the circle that are darker (S_{dark}) than $I(p)$ are calculated. V is then defined as the maximum of these two sums:

$$V = \max \Big(\sum_{q \in S_{bright}} |I(q) - I(p)| - t, \quad \sum_{q \in S_{dark}} |I(p) - I(q)| - t \Big) \tag{2.18}$$

  If the detector has found two adjacent keypoints, the one with a lower V value is discarded.

2.7.4 BRIEF

As already mentioned, some applications need to run the process of feature detection, description and matching in real-time with limited computational resources. Therefore, it is not only important to improve the detection of the features, but also to implement a feature description process that is more efficient in terms of computational effort and memory space in order to enable fast feature matching. Therefore, Calonder et al. [8] developed a binary feature descriptor called Binary Robust Independent Elementary Features (BRIEF) that describes a keypoint by creating a binary string vector based on its neighborhood. The neighborhood is a square patch of size $S \times S$ pixels around the keypoint centered in the patch. The advantage of creating binary descriptors instead of common descriptors, such as the ones of SIFT or SURF, is that they can be compared using Hamming distance instead of Euclidean distance. The Hamming distance can be computed very efficiently with a bitwise XOR operation which speeds up the matching process significantly. The general approach of BRIEF is very simple. A binary test τ is

performed that compares the intensity values I of two pixels p and q in the patch:

$$\tau(p,q) = \begin{cases} 1 & \text{if } I(p) < I(q) \\ 0 & \text{otherwise} \end{cases} . \tag{2.19}$$

A set of k (p,q)-pixel-pairs are chosen to uniquely define a set of binary test results. The BRIEF descriptor is then the k-dimensional bit-string

$$f_k = \sum_{1 \le i \le k} 2^{i-1} \, \tau(p_i, q_i). \tag{2.20}$$

In the paper of Calonder et al. [8] $k = 128$, 256 and 512 were considered and it was pointed out that $k = 256$ provides a good trade-off between speed and storage efficiency. That leads to a 256D bit-string vector which requires only 32 byte of storage When creating such descriptors, two choices have to be made in advance:

- **Smoothing kernels**: Due to the fact that the method only takes the information of single pixels into account, it is very noise-sensitive. In order to reduce this sensitivity and thus increase stability as well as repeatability of the descriptor, the patch needs to get pre-smoothed. This is done by applying a Gaussian kernel. Calonder et al. [8] showed that $\sigma = 2$ with a kernel window of a size of 9×9 pixels is sufficient.
- **Spatial Arrangement of the Binary Tests**: Generating a k bit long vector leaves many options for selecting the k test pixel pairs (p_i, q_i) in the patch. Different sampling geometries can be chosen. The best results were achieved by drawing both pixels from a Gaussian distribution of $(0, \frac{1}{25}S^2)$ around the keypoint [8].

2.7.5 ORB

Rublee et al. [46] propose a feature detection and description algorithm that builds on the FAST keypoint detector (see section 2.7.3) and the BRIEF feature descriptor (see section 2.7.4) and is thus called Oriented FAST and Rotated BRIEF (ORB). As stated out earlier in the respective sections, both of these algorithms are very attractive because of their advantages regarding computational speed. However, FAST as well as BRIEF contain several limitations which are addressed and improved in the ORB algorithm to provide a robust feature detection and description algorithm for real-time applications.

Oriented FAST (oFAST)
The authors of [46] noticed that FAST has large response along edges as well. Therefore, they employ the Harris cornerness value, a measure of cornerness, that is introduced earlier in section 2.5 [17]. For a desired number of N keypoints, the threshold value t is set low enough to obtain more than N keypoints. Then, the Harris cornerness value is calculated for these points and they are ordered respectively. Afterwards, the top N points with the best score are picked as the resulting features. In order to circumvent the sensitivity to scale of FAST this process is repeated for the same image at different scales.

After the keypoints are selected, the orientation is determined for each of them by calculating the intensity centroid [42]. The intensity centroid C assumes that a corner's intensity is shifted from its center, and the vector $\vec{OC}$, from the corner's center O to the centroid, is used to compute an orientation. Therefore, the moments of a patch around the keypoint are determined as

$$m_{ij} = \sum_p x^i y^j I(p), \tag{2.21}$$

where $i, y \in \{0, 1\}$, p are the pixels of the patch, (x, y) are their locations and $I(p)$ is the Intensity of p. The centroid can then be the computed as

$$C = \Big(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\Big). \tag{2.22}$$

Assumed that the center of the corner O lies in the center of the local coordinate system of the patch, the vector $\vec{OC}$ can be easily determined and the orientation of the patch is therefore:

$$\theta = \text{atan2}(m_{01}, m_{10}). \tag{2.23}$$

Rotated BRIEF (rBRIEF)
In order to provide invariance to rotation the selected pixel pairs (p, q) for the binary test τ are rotated according to the orientation of the patch θ. This is done by defining a $2 \times k$ matrix $\boldsymbol{S}$ based on the pixel pairs and applying the rotation matrix $\boldsymbol{R}$ based on θ:

$$\boldsymbol{S} = \begin{bmatrix} p_1 & p_2 & ... & p_k \\ q_1 & q_2 & ... & q_k \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \tag{2.24}$$

$$\boldsymbol{S_\theta} = \boldsymbol{R} \cdot \boldsymbol{S} \tag{2.25}$$

Additionally, there are two properties these pixel pairs should have. First, there should exist a high variance among these points which leads to more distinctive feature descriptors. Second, uncorrelated pixel pairs used in the binary tests are desired, as each test then contributes to the result. In order to achieve these properties, ORB performs a greedy search among all possible pixel pairs to select those that are both uncorrelated and of high variance. These are rotated as described before and used to calculate the descriptor vector (see equation 2.19 and 2.20). More details on rBRIEF can be found in [46]. The length of the resulting bit-string is typically 256 bit which require 32 byte of storage.

2.7.6 BRISK

Leutenegger, Chli, and Siegwart [26] developed another feature detection and description method called Binary Robust Invariant Scalable Keypoints (BRISK) whose detection part is also based on the FAST algorithm. With the goal of real-time and high quality feature matching, a novel bit-string descriptor based on intensity comparisons by dedicated sampling of each keypoint neighborhood was developed.

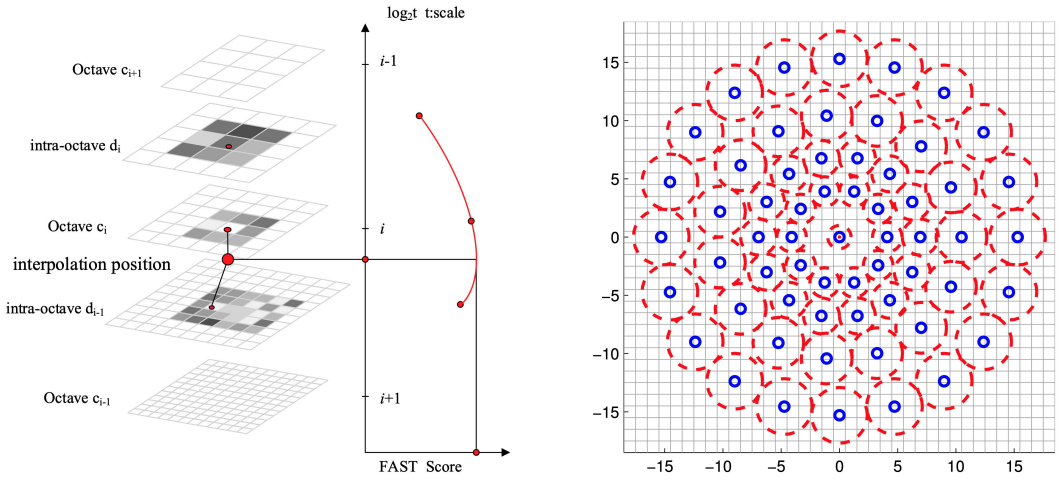
Scale-Space Keypoint Detection

As well as ORB, BRISK aims to extract scale invariant features. Therefore, a scale-space is created with typically 4 octaves and intra-octaves in which the original image is down-sampled each time. Initially, the FAST detector is applied on each octave and intra-octave separately in order to identify potential keypoints. Then, these keypoints are subjected to a non-maximum suppression in scale-space. The FAST score s based on the score function V (see section 2.7.3) is computed for the keypoint, its eight adjacent pixels in the same octave and for the the adjacent pixels in the layer above and below. All scores of these neighboring pixels have to be lower than the score of the keypoint. Afterwards, by fitting a 2D quadratic function and interpolating between the scale octaves, the precise location and scale of each keypoint are determined (see fig. 2.9 (A); more details in [26]).

Keypoint Description

Just like many other binary feature description algorithms, BRISK also creates the descriptor vector based on intensity comparisons of pixel pairs in a certain neighborhood around the keypoint. Contrary to BRIEF using a set of randomly selected pixel pairs for these binary tests, BRISK selects a fixed neighborhood sampling pattern around the keypoint. Therefore, four concentric circles are built in a patch of size 40×40 pixels centered on the point of interest. On these four circles, $N = 60$ points (blue) that build our sampling locations are evenly distributed (see fig. 2.9 (B)). Gaussian smoothing with standard deviation σ_p proportional to the distance between the points on the respective circle is applied. Next, in order to enable invariance to rotation the orientation of the keypoint is determined. Therefore, in general the local gradient between two pixel points p and q is estimated as follows:

$$g(p,q) = (p,q) \cdot \frac{I(q,\sigma_q) - I(p,\sigma_p)}{\|q-p\|}, \tag{2.26}$$



(A) Scale-space (B) Sampling pattern

FIGURE 2.9: Illustration of the precise location and scale determination in the scale-space using interpolation (A) and the sampling pattern (B) for the BRISK algorithm [26].

where $I(p, \sigma_p)$ is the intensity value of sampling point p after Gaussian smoothing with σ_p. A set of point pairs of all sampling points is defined as $\mathcal{A}$. Additionally, a subset of short-distance pairs $\mathcal{S}$ and of long-distance pairs $\mathcal{L}$ is defined:

$$\begin{aligned} \mathcal{S} &= \{(p,q) \in \mathcal{A} \mid \|q-p\| < \delta_{max}\} \subseteq \mathcal{A} \\ \mathcal{L} &= \{(p,q) \in \mathcal{A} \mid \|q-p\| > \delta_{min}\} \subseteq \mathcal{A} \end{aligned}. \tag{2.27}$$

Iterating through the point pairs in $\mathcal{L}$, the overall characteristic pattern direction of the keypoint is estimated:

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \frac{1}{L} \cdot \sum_{(p,q) \in \mathcal{L}} g(p,q), \tag{2.28}$$

where L is the number of pairs in $\mathcal{L}$. The rotation is then computed by:

$$\theta = \text{atan2}(g_y, g_x). \tag{2.29}$$

After the long-distance subset is used to estimate the direction of the keypoint, all short-distance sampling points are used to build the binary descriptor vector. In order to obtain rotation invariance, the sampling points are rotated by the angle θ. The descriptor bit-string is determined as follows:

$$\tau(p^\theta, q^\theta) = \begin{cases} 1 & \text{if } I(q^\theta, \sigma_q) > I(p^\theta, \sigma_p) \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \tag{2.30}$$

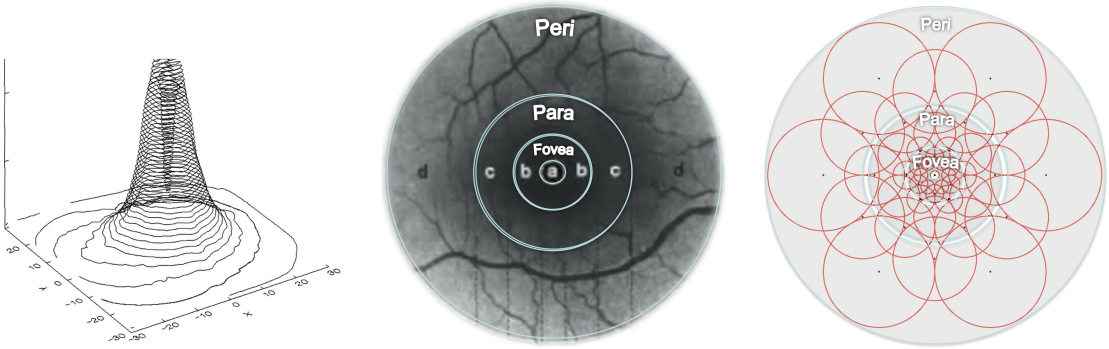
$$f = \sum_{1 \le i \le k} 2^{i-1} \, \tau(p_i^\theta, q_i^\theta) \quad , \tag{2.31}$$

where (p_i^θ, q_i^θ) are the rotated sampling point pairs corresponding to $(p_i, q_i) \in \mathcal{S}$. Typically, the bit-string has a length of $k = 512$ bit and thus needs 64 byte of storage.

2.7.7 FREAK

Alahi, Ortiz, and Vandergheynst [2] developed another feature description algorithm called Fast Retina Keypoint (FREAK) inspired by the human visual system. Similar to BRISK, FREAK selects a limited number of points in a specific sampling pattern as a set of possible sampling pairs. As well as the approach of ORB, FREAK uses a machine learning technique to learn the optimal set of sampling pairs. The method can be structured in three parts:

1. **Retina sampling pattern**: Neuroscience has discovered that the topology of the human's retina plays an important role during the extracting of details from images. The retina consists of multiple ganglion cells and each cell is influenced by multiple photoreceptors. The region where light influences the response of a ganglion cell is called receptive field. With increasing radial distance from the center of the retina, the size of the fields increases as well and the density of ganglion cells decreases (see fig. 2.10 (A)). The ganglion cells are segmented into four regions: foveal, fovea, parafoveal and perifoveal (see fig. 2.10 (B)). The pattern



(A) Density of ganglion cells (B) Retina areas (C) Sampling pattern

FIGURE 2.10: Illustration of the analogy between the human visual system and the sampling pattern in FREAK. (A) shows the distribution of ganglion cells over the retina. The density is clustered in four areas (a: foveola, b: fovea, c: parafoveal, d: perifoveal) shown in (B). (C) illustrates the distribution of sampling points inspired by the ganglion cells [2].

of the ganglion cells is transferred to choose a set of sampling points in order to describe the keypoint representing the center of the retina. Similar to BRISK a circular sampling grid is used with the difference of a higher density of points near the center. The density of points decreases exponentially with increasing distance to the keypoint (see fig. 2.10 (C)). The sampling points are marked black representing the ganglion cells of the retina. In order to be less sensitive to noise, each sampling point is smoothed with a Gaussian kernel where the radius of the circle (marked red) illustrates the size of the standard deviation of the kernel.

2. **Coarse-to-fine descriptor**: The binary descriptor is constructed by comparing pairs of sampling points of the retinal sampling grid according to equation 2.19 and 2.20. In order to find the most descriptive sampling pairs among thousands of possible pairs, Alahi, Ortiz, and Vandergheynst [2] developed an algorithm to learn the best pairs from a huge training data. Surprisingly, a symmetric structure in the selected pairs can be observed. The first pairs that should be used to determine the descriptor are mainly peripheral pairs whereas the last ones implicate highly centered points. This indicates that a coarse-to-fine ordering of the pairs is automatically preferred. In addition, the study shows that the first 512 pairs are the most relevant ones and by adding more pairs the performance does not increase which leads to an bit-string of size 512 bit requiring 64 byte of storage.

3. **Orientation**: In order to circumvent sensitivity to rotation, the rotation of the keypoint is determined by summing the estimated local gradients over a selection of pairs similar to BRISK. But instead of using thousand of pairs with a long distance, FREAK select 45 pairs with symmetric receptive fields with respect to the keypoint. The orientation then can be computed with equation 2.28 and 2.29.

2.7.8 Summary of Algorithms

In summary, five feature detection algorithms (SIFT, SURF, FAST, ORB and BRISK) and six feature description algorithms (SIFT, SURF, BRIEF, ORB, BRISK and FREAK)

are introduced in section 2.7.
The SIFT feature detection and description algorithm is known for its robustness in various computer vision applications due to its invariance to scale and rotation. However, it should be noted that SIFT is computationally intense and may not be suitable for real-time applications on resource-constrained devices such as satellites. Additionally, by creating a 128D floating point descriptor vector, the comparison of different features during the feature matching process and the memory space needed for storing multiple features in a database might cause difficulties. Nevertheless, its fundamental concepts have inspired the development of other feature detection and description methods such as SURF.
The SURF algorithm has gained popularity due to its efficient improvements to the SIFT approach. The smaller size of the feature vector (64D floating point vector) leads to a faster determination and comparison in general. According to the developers [4], due to the smaller descriptor as well as all other improvements and approximations, SURF is three times faster than SIFT with a comparable performance which makes the process of feature matching using SURF suitable for many simple real-time applications.
FAST is an extreme fast corner detection algorithm. Due to the fact that the detection is based on analyzing the neighboring pixels on a circle and a circle is invariant to rotation, the method detects the same features in rotated images and thus eliminates one key downside of typical corner detectors. However, the algorithm is not robust to high levels of noise and it is sensitive to scale. Therefore, the algorithm is adapted and improved in the ORB algorithm (see section 2.7.5) as well as in the BRISK algorithm (see section 2.7.6).
By creating a 256D bit-string vector, the binary feature description algorithm BRIEF provides a descriptor that requires only 32 byte of storage. As already mentioned, comparing strings can be done by computing the Hamming distance. This can be performed extremely fast by modern CPUs that often provide a specific instruction to perform *XOR* operation. However, BRIEF is not designed to be invariant to rotation and is thus very sensitive to in-plane rotation.
The ORB feature detection and description algorithm offers several advantages, including computational speed and robustness in terms of invariance to rotation and scale, by combining and improving FAST and BRIEF. It is important to mention that even though these improvements lead to more robustness, a higher computational effort is required for their implementation. However, during experiments of [46], ORB was two times faster than SIFT, while performing just as well in many scenarios, which does it make a promising candidate for real-time applications.
BRISK is a feature detection and description algorithm with scale and rotation invariance. Even though its binary descriptor vector is twice as long as the one of BRIEF or ORB, the algorithm is suitable for many real-time applications. The experiments of [26] show BRISK's high quality performance compared to other state-of-the-art algorithms at a dramatically lower computational cost, e.g. an order of magnitude faster than SURF.
The FREAK algorithm is a robust and fast feature descriptor that computes a 512 bit binary feature vector as well as BRISK. The experiment of the developers shows, that

in general FREAK is faster in computation with lower memory space and also more robust than SIFT, SURF or BRISK. FREAK is thus another competitive alternative for embedded real-time applications.

2.8 Feature Matching

After describing the detected features, the image is represented by a set of local features. The main objective of feature matching is to recognize certain parts of an image by comparing the set of detected feature descriptors with another one. After the matching is done, only features that occur in both sets should remain. In most applications, two images are to be matched. Therefore, the second set of feature vectors belongs to another image. In the case of AVS, the second set consists of the feature vectors of the GCPs stored in a database but the concept remains the same. The process of feature matching consists of three parts [3]:

1. **Similarity Measure**: In order to evaluate in mathematical sense how similar two descriptors are, the distance between them is calculated. There are many types of how the measurement could be implemented. The most popular ones are determining the Euclidean or the Hamming distance. The Euclidean distance d_E between two n-dimensional vectors $\vec{v} = (v_1, v_2, ..., v_n)$ and $\vec{w} = (w_1, w_2, ..., w_n)$ is defined as follows:

$$d_E(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}. \tag{2.32}$$

   It is well suited for floating point feature vectors such as the ones generated by SIFT or SURF. The Hamming distance is used to compare binary feature vectors and is defined as the number of positions at which the corresponding elements of two vectors $\vec{v}$ and $\vec{w}$ with same length n are different:

$$d_H(\vec{v}, \vec{w}) = |\{j \in \{1, 2, ..., n\} \mid v_j \neq w_j\}|. \tag{2.33}$$

2. **Matching Strategy**: After determination of the distance of two vectors, a decision has to be made whether two features are similar and thus are matching or not according to the matching strategy. There exist three common approaches: Absolute Thresholding (AT), Thresholded Nearest Neighbor (TNN) and Nearest Neighbor Distance Ratio (NNDR). Using AT, two features are considered as a correct match if their distance is less than a certain threshold. Therefore, each feature from the first descriptor set may match to more than one feature from the second feature set. The concept of TNN matches each feature from the first set to its nearest neighbor feature from the second set, if the absolute distance between them is less than a threshold. In this way, only some features of the first set may match and, if this is the case, the match is unambiguous. NNDR computes the distances of a feature from the first set to its nearest and second nearest neighbors of the second set. If the ratio between these distances is less than a threshold, the feature is matched to its nearest neighbor.

3. **Searching Technique**: There are several possibilities how to iterate over two sets and e.g. find the pair of vectors with smallest distance to each other. The simplest way is a Brute-Force approach, where each feature of the first set is compared with all features of the second set. This search leads to high accuracy but also to high computational effort. In order to speed up the process, many methods have been proposed for Approximate Nearest Neighbor (ANN) search. ANN trades accuracy for efficiency by querying the second set in a more efficient way but not guaranteeing to return the true nearest neighbor. The two main techniques of ANN are hierarchical space partition-based and hash-based methods.

In order to find the best feature matching algorithm for a specific application, one can simply choose a similarity measure, a matching strategy and a searching technique based on the requirements of the application.

2.9 Feature Validation

Even though a suitable combination of similarity measure, matching strategy and searching technique is chosen for a certain application, it always happens that some features are matched wrongly. To ensure a reliable process of feature matching the last task is to validate the matches. During this process, mismatches should be found and discarded. This can be done in many ways leading to different outlier removal algorithms. In related work, the process done during feature validation is listed as an additional point under the term feature matching. As the validation of the matched features plays an important role during AVS, in this work the concepts of matching and validation are considered separately.

2.9.1 RANSAC

One algorithm that finds outliers and separates the matched features into an inlier and outlier feature set is Random Sample Consensus (RANSAC) [15]. The goal of RANSAC is to find a model that describes how the two sets of datapoints (in this case image features) are correlated. After finding this model, datapoints that do not fit the model are detected as outliers. Unlike many other sampling techniques, that use as much of the data as possible to obtain the model, RANSAC takes the smallest possible set to determine the underlying model and continues to enlarge this set with datapoints that fit the model as well. The algorithm consists of the following steps [12]:

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.

2.9.2 Affine Transformation

As mentioned, the goal of RANSAC is to determine the best fitting model for the analyzed datapoints. Which model is used depends on the structure and characteristics of the datapoints. For image feature matching, a geometrical transformation model such as affine transformation is often used as the underlying model. Consider two images, each containing a set of points. Each point $\vec{p_1}$ of image 1 corresponds to a certain point $\vec{p_2}$ of image 2 representing matched features of two images. An affine transformation describes how a triangle spanned by three points of image 1 has to be rotated, translated and scaled to obtain the triangle spanned by the corresponding points of image 2 (see fig. 2.11). After finding this transformation, it can be applied to each single point. If the transformed point differs from the corresponding point of image 2, it does not fit the determined model [3].
Mathematically, the affine transformation can be described as follows. Consider two corresponding points of image 1 $\vec{p_1} = (x, y)^T$ and of image 2 $\vec{p_2} = (u, v)^T$. Then, the transformation can be expressed by the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \text{ or} \tag{2.34}$$

$$\vec{p_2} = \boldsymbol{M} \cdot \vec{p_1} + \vec{t}, \tag{2.35}$$

where the matrix $\boldsymbol{M}$ represents the scale and rotation and the vector $\vec{t}$ the translational part. In order to find an affine transformation, the six model parameters have to be determined. By using at least three points of each set, the equation can be rearranged

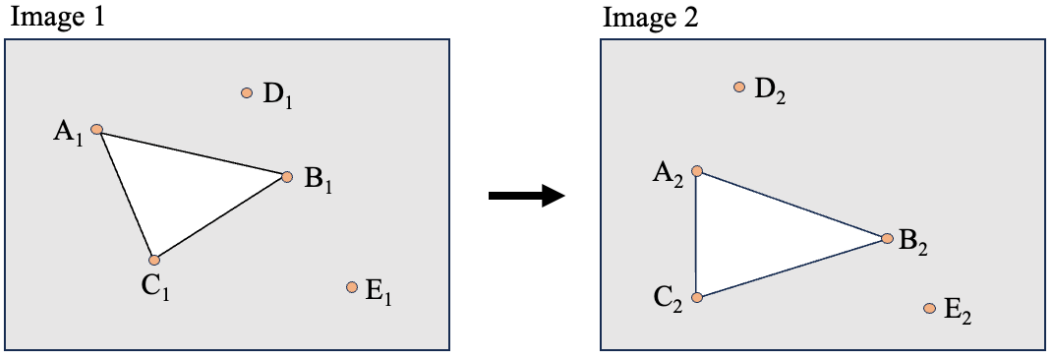


FIGURE 2.11: Illustration of the affine transformation: Left and right image show corresponding datapoints of image 1 and image 2 respectively and the triangle spanned by three corresponding datapoints.

so that the model parameters are summarized in a column vector $\vec{x}$:

$$\underbrace{\begin{bmatrix} u_1 & v_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & u_1 & v_1 & 0 & 1 \\ u_2 & v_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & u_2 & v_2 & 0 & 1 \\ u_2 & v_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & u_2 & v_2 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \\ t_x \\ t_y \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix}}_{\vec{b}}, \tag{2.36}$$

This system of linear equations can then be solved as follows:

$$\vec{x} = (\boldsymbol{A}^T \cdot \boldsymbol{A})^{-1} \cdot \boldsymbol{A}^T \cdot \vec{b}, \tag{2.37}$$

where $(\boldsymbol{A}^T \cdot \boldsymbol{A})^{-1} \cdot \boldsymbol{A}^T$ is the pseudo inverse of matrix $\boldsymbol{A}$ [3].

2.10 Ground Control Points

In the traditional sense, GCPs are defined as photographically identifiable points on the Earth's surface whose geographic location is precisely known. They are used in remote sensing and geospatial mapping where they serve as reference points to accurately match and align aerial or satellite imagery with real-world coordinates. Typically, GCPs are physical markers or targets that can be easily identified in both the imagery and on ground. Examples of GCPs include artificial targets such as specially designed ground markers, painted crosses, or natural features like road intersections or prominent landmarks. These points should be stable and permanent, allowing a reliable identification and measurement over time. Their geographic locations are measured on ground using high-precision surveying techniques [1, 51].
In the context of AVS, the term GCP has a slightly different meaning. Here, GCPs are also defined as photographically identifiable points on the Earth's surface with a known geographic location. The difference is that these points are not selected manually by human beings but by feature detection algorithms such as the ones described in section 2.7. These algorithms are applied to satellite imagery in order to extract locations on Earth's surface with unique appearance in satellite images. These points can neither be easily interpreted nor found on ground. Thus, their geographic locations cannot be determined by measurements on ground. In order to be able to determine their Earth's position anyway, the feature detection algorithms 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 detecting features in the image, their pixel position in combination with the metadata can then be used to determine their exact ground position. If this process is successfully done, the detected keypoints get a unique signature by a feature description algorithm. Afterwards, each GCP in the context of AVS consists of an unique description mapped with its absolute ground position with respect to the ECEF coordinate system and can be stored in a database.

In this way, GCPs should be easily found in other satellite imagery by simply detecting features in this image and matching the detected keypoints with the GCP database.

Chapter 3

Test Application

As already mentioned, the main goal of this thesis is to analyze the process of feature detection, description, matching and validation during AVS. Therefore, a test application is developed. The general concept regarding AVS, the used satellite image data as well as the concrete setup and implementation are described in the following chapter.

3.1 Concept

In general, the entire process regarding feature detection and matching in AVS can be divided into two main parts: Creation of the GCP database on ground and the onboard feature matching (see fig. 3.1).

Creation of the Database
The following steps are performed on ground in advance of the launch of the satellite. First, a dataset of high-quality satellite image data with corresponding detailed metadata is required. The metadata has to provide precise information about the geographic location of the image. The satellite images are then analyzed by a feature detection algorithm that extracts unique keypoints. Afterwards, the position on Earth's surface is determined for each keypoint based on its pixel position and the given metadata. Each keypoint gets a signature by a feature description algorithm. The pair of feature description and absolute position on Earth are stored in a database which is integrated into the satellite.

Onboard Feature Matching
After creating and integrating the GCP database into the satellite, the onboard procedure is quite intuitive. The integrated onboard camera provides live images of the Earth's surface. A feature detection algorithm extracts keypoints which are described by a feature description algorithm afterwards. These descriptors are then matched with the GCP descriptors stored in the database. Optionally, a validation method can be applied to find and discard false matches. As a result, a list of image positions mapped with their geographic location on Earth are provided which can then be used to determine the absolute attitude and position of the satellite.

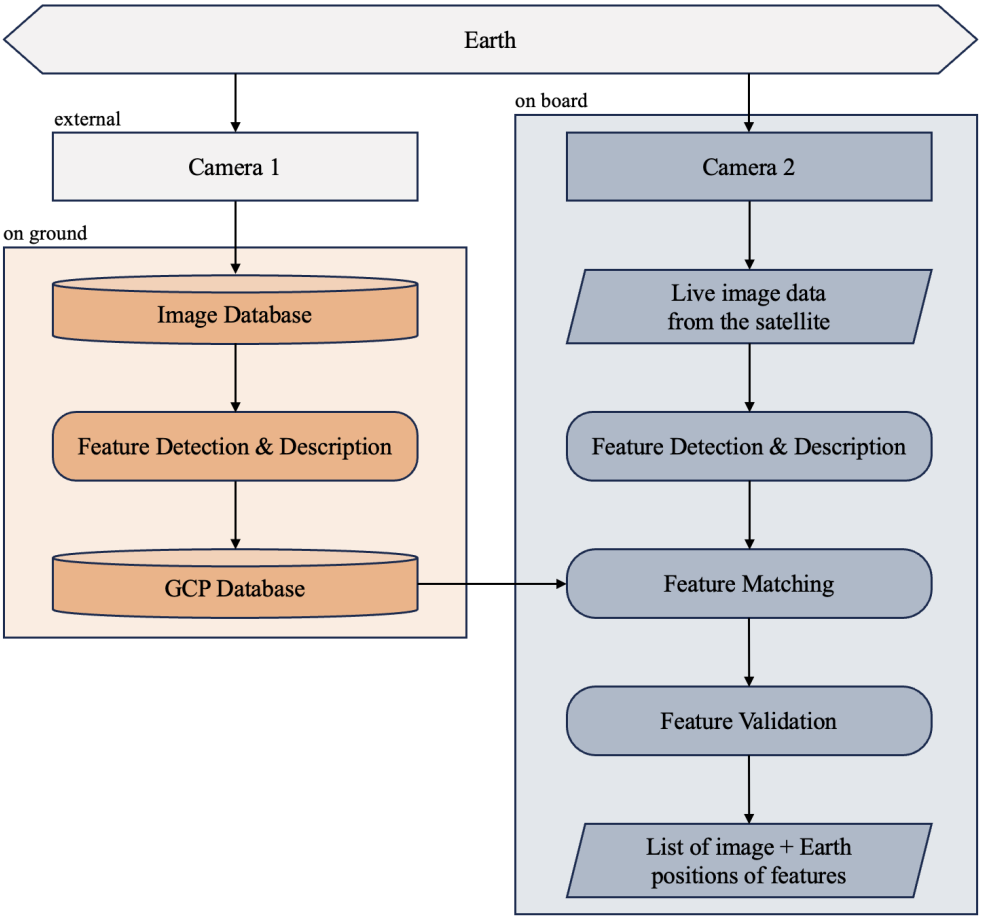


FIGURE 3.1: Overview of the processes of feature detection, description and matching in AVS.

3.2 Satellite Image Data

To be able to create a database with feature descriptors and their geographic locations a suitable satellite image dataset is required. ‹Suitable› in this context means to meet the following requirements:

- The dataset provides RGB-images similar to those the onboard camera of the satellite takes.
- 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.

3.2.1 Landsat 8 Dataset

One dataset that meets all the mentioned requirements is offered 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 [33, 58]. The most recent satellite of the program is the Landsat 9 that was launched recently on 27 September 2021. Thus, the data provided by Landsat 9 is limited. Its predecessor 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) [52]. The characteristics of the bands are summarized in table 3.1. Each imagery data acquired by the two sensors of the satellite represents one scene of the Worldwide Reference System-2 (WRS-2) which is illustrated in appendix A.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. One of these data portals used in this thesis is EarthExplorer, a website provided by USGS [50].
The Landsat 8 data is divided into different Collections, Levels and Tiers resulting in many different datasets. In order to get the most suitable data for a certain application, it is necessary to understand the characteristics of each category. There exist two versions of Landsat datasets called Collections. Each Collection represents a set of processing methods for the Landsat raw images. Within one Collection, all data share the same radiometric and geometric parameters, calibration methods, etc. The data provided by Collection-2 has several improvements in comparison to Collection-1 as it offers e.g. improved geographic location accuracy and radiometric calibration as well as a more detailed and consistent metadata-file. Important to mention is that Landsat

Spectral Band	Description	Wavelength	Resolution
Band 1	Coastal Aerosol	0.43 µm–0.45 µm	30 m px^{-1}
Band 2	Blue	0.45 µm–0.51 µm	30 m px^{-1}
Band 3	Green	0.53 µm–0.59 µm	30 m px^{-1}
Band 4	Red	0.64 µm–0.67 µm	30 m px^{-1}
Band 5	Near-Infrared	0.85 µm–0.88 µm	30 m px^{-1}
Band 6	SWIR 1	1.57 µm–1.65 µm	30 m px^{-1}
Band 7	SWIR 2	2.11 µm–2.29 µm	30 m px^{-1}
Band 8	Panchromatic	0.5 µm–0.68 µm	15 m px^{-1}
Band 9	Cirrus	1.36 µm–1.38 µm	30 m px^{-1}
Band 10	Thermal infrared 1	10.60 µm–11.19 µm	100 m px^{-1}
Band 11	Thermal infrared 2	11.50 µm–12.51 µm	100 m px^{-1}

TABLE 3.1: Parameters of the acquired spectral bands of Landsat 8.

Collection-1 data is no longer available to download since 30 December 2022 [53, 54]. Additionally, for each Collection the data is evaluated and categorized in one of three Tiers based on their data quality and level of processing. Immediately after acquisition, the data is stored into the Real-Time Tier. Even though the Landsat 8 imagery data is very accurate, additional processing is needed. After this is done, the data is categorized into Tier 1 or Tier 2 based on their data quality whereby data with the highest available quality is placed into Tier 1. Among other factors, data quality can be affected by significant cloud cover or insufficient GCPs [53, 54].
Lastly, there exist two data categories called Levels. Level-1 products include quantized and calibrated scaled Digital Numbers (DN) representing the multi-spectral image data. The DNs of the reflective bands (Band 1-9) can be converted to Top of Atmosphere (TOA) reflectance using the corresponding rescaling coefficients provided by the metadata-file. TOA reflectance is the measured reflectance by a space-based sensor flying higher than the Earth's atmosphere. These values include effects from clouds and atmospheric aerosols and gases. In order to eliminate these effects on the image data, Level-1 data is corrected by several methods. The resulting image data represents the Surface Reflectance (SR). SR represents an estimate of the Earth's surface spectral reflectance as it would be measured on ground without atmospheric effects [56, 57].

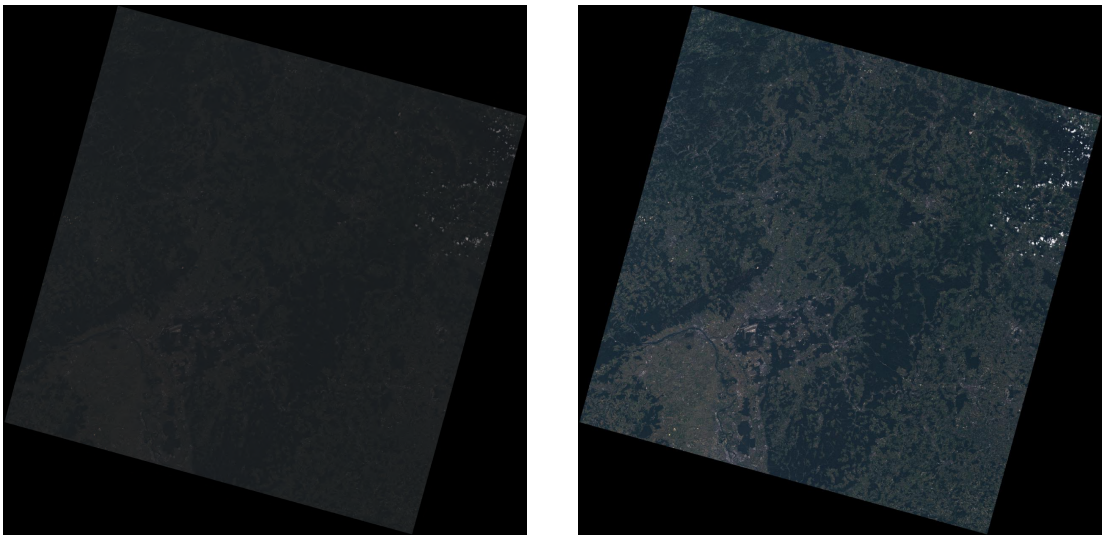
3.2.2 Generating RGB-Images

In order to extract GCP features for the onboard database which the satellite is able to detect, RGB-images have to be generated that are similar to the onboard data. Due to the fact that the Collection 1 data is no longer available to download, the Collection 2 data is chosen. This Collection would be taken anyway due to the several improvements mentioned earlier. The pictures taken by the onboard camera won't be preprocessed to eliminate effects of the atmosphere but represent the 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. In conclusion, the Landsat 8 Collection-2 Level-1 Tier-1 data is used to generate RGB-images.
RGB-images consist of three color channels: Red, Green and Blue. In order to generate a RGB-image, the three single color channels can be added together. Thus, the spectral bands 2, 3 and 4 of the Landsat 8 satellite are simply placed on top of each other. As mentioned in section 3.2.1, to get the TOA reflectance data from the Tier-1 data, first the DN values of each band must be converted using the corresponding coefficients provided by the metadata-file as follows [56]:

$$\rho'_\lambda = m_\rho \cdot \mathrm{DN} + a_\rho, \tag{3.1}$$

where ρ' is the TOA reflectance and m_ρ and m_ρ are the band-specific multiplicative and additive rescaling factors from the metadata respectively. Additionally, a correction for the sun angle can be determined by taking the local sun elevation angle θ_{SE} into account that is provided by the metadata as well [56]:

$$\rho_\lambda = \frac{\rho'_\lambda}{\sin \theta_{SE}}. \tag{3.2}$$



(A) Shifting and scaling (B) Google Earth Engine visualization parameters

FIGURE 3.2: Example of generated RGB-images based on the provided Landsat 8 data using simple shifting to zero and scaling (A) or adjusting the reflectance values based on Google Earth Engine visualization parameters (B).

Then, these values have to be adapted to the pixel range of RGB-images from 0-255. But by simply shifting the minimum value to zero, scaling the other values of each band to 255 and adding the three bands together, the resulting RGB-image appears very dark and with low contrast (see fig. 3.2 (A)). That is because after converting the DNs to TOA reflectance values, the data is still representing the precise amount of reflected light and not an image format. In order to get a realistic true color RGB-image such as the ones of the onboard RGB-camera, the reflectance values must be scaled differently. For this work, the visualization parameters Google Earth Engine is suggesting are adopted [16]:

$$\rho_{px} = \begin{cases} 0.4 & \text{if } \rho_\lambda \geq 0.4 \\ 0.0 & \text{if } \rho_\lambda \leq 0.0 \\ \rho_\lambda \cdot \frac{255}{0.4} & \text{otherwise} \end{cases} \tag{3.3}$$

After applying all these conversions a realistic RGB-image is generated (see fig. 3.2 (B)). 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).

Note: The used Landsat 8 data for all images shown in the figures of this thesis and used for the different test runs and experiments are referenced in appendix A.

3.3 Setup of the Test Application

The following section focuses on the actual setup of the test application. First, the procedure of creating the database for the GCPs is explained before the process of the onboard feature matching is described. Whenever feature detection and description are referred to in the following section, all algorithms selected in section 3.5 are meant.

3.3.1 Ground Control Points Database

As explained in section 2.2, the main objective of the GCP database is to provide features that are, first, equally distributed over the globe and, second, have a unique appearance in satellite imagery so they can be found by the onboard camera of a satellite. To ensure these two requirements, the first approach of creating the database is implemented as follows:

Feature Detection and Description
At first, a scene of the WRS-2 is chosen that should be covered by the database. For this scene, two Landsat 8 data packages of Landsat 8 Collection-2 Level-1 Tier-1 are selected and the RGB-images are generated. During selection, care is taken that the scenes have 0% cloud cover to ensure that the entire area on ground is visible and only unique appearances of Earth's surface are extracted. In addition, because it has not yet been investigated how seasonal changes could effect the recognition of a feature, the two selected data packages should have been recorded approximately at integer annual interval within the same month (± 1). For the following tests in chapter 4, all images are taken from April, May or June. If no such data with 0% cloud cover exist, than images from March and July are involved as well. Furthermore, the more recent data package must be from no later than 2020.
Then, features have to be extracted from each of the two RGB-images. It is important to notice that due to the FOV of the satellite's camera, the onboard images cover a smaller area on ground than the RGB-images generated from the Landsat 8 data. To ensure that the onboard camera is able to see a certain amount of GCP features within its FOV, the GCPs have to be equally distributed and thus each image is divided into smaller regions. The feature detection algorithm is applied to each region and extracts features.

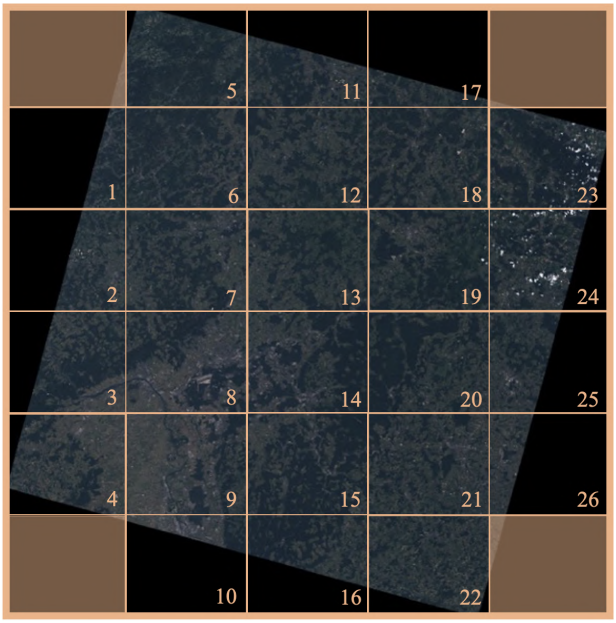


FIGURE 3.3: Division of the Landsat 8 RGB-images into smaller regions for feature detection regarding the TOM camera parameters.

Then, the description method creates the descriptor vectors for these keypoints. Afterwards, all features of a RGB-image are collected resulting in a list of features equally spaced over the RGB-image. How the image needs to be divided depends on the FOV of the onboard camera and the associated swath width and height. For this work, the parameters of TOM are used whose camera has a swath width of about 50 km and a swath height of about 40 km. As the RGB-images cover an area of about 240 km $\times$ 240 km it is divided in $\frac{240\,\text{km}}{50\,\text{km}} = 4.8 \approx 5$ columns and $\frac{240\,\text{km}}{40\,\text{km}} = 6$ rows as can be seen in fig. 3.3. As the regions in the four corners hardly depict any Earth's surface, these regions are neglected leading to a total number of 26 regions for each scene.

Feature Matching
The next step consists of matching the detected features of the two images showing the same scene. The resulting matches should then include features that are constant over time, can be detected in different images and have a unique signature so they are recognized during the matching process. All these properties lead to a high probability that these features are detected and correctly matched onboard as well. For the matching process the following parameters are used:

- Similarity Measure: Hamming distance for all binary feature description algorithms and Euclidean distance otherwise.
- Matching Strategy: Finding for each feature of the first image its nearest neighbor of the second image regarding descriptor distance.
- Searching Technique: Because the database is created in advance of the mission, the required computational effort does not matter and high accuracy is needed. Thus, the Brute-Force approach is used comparing each feature of the first set with all features of the second set.

Feature Validation
In order to guarantee that only features of correct matches are included in the database an additional validation step is performed. This is done by determining and comparing the geographic positions of the matched features on ground. First, a coordinate system must be chosen to describe the ground position. As the metadata-file of each image includes the coordinates of the four corners in the WGS-84- as well as in the UTM format, one of these reference systems have to be chosen. WGS-84 is a spherical coordinate system which complicates the computation of the ground position of any pixel in the image plane. The UTM system is a cartesian coordinate system representing the transverse Mercator projecting of the Earth surface into a two-dimensional plane. Due to the fact that the UTM system lies parallel to the image plane coordinate system of each Landsat 8 image, the UTM position of any pixel (p_x^{utm}, p_y^{utm}) can be easily determined by knowing its pixel coordinates (p_x^{px}, p_y^{px}), the UTM coordinates of the upper left (u_x^{utm}, u_y^{utm}) and lower right (l_x^{utm}, l_y^{utm}) image corner as well as the pixel width w and height h (see fig. 3.4):

$$p_x^{utm} = u_x^{utm} + (l_x^{utm} - u_x^{utm}) \cdot \frac{p_x^{px}}{w} \tag{3.4}$$

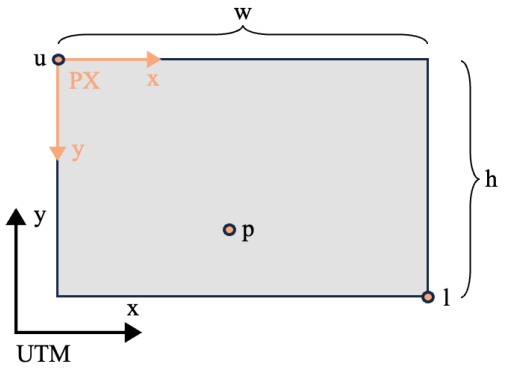


FIGURE 3.4: Illustration of determining the UTM position of a pixel.

$$p_y^{utm} = l_y^{utm} + (u_y^{utm} - l_y^{utm}) \cdot (1 - \frac{p_y^{px}}{h}). \tag{3.5}$$

In order to evaluate whether two features f and g are correctly matched, the following conditions have to be satisfied:

$$\left|f_x^{utm} - g_x^{utm}\right| \leq \epsilon \quad \text{and} \tag{3.6}$$

$$\left|f_y^{utm} - g_y^{utm}\right| \leq \epsilon. \tag{3.7}$$

In this test application, the threshold ϵ is set to 60 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. If both conditions are satisfied, the feature is identified as a candidate for the GCP database. As the area covered by a Landsat 8 image does not exactly represent a scene of the WRS-2 (see fig. 3.5), data from adjacent rows or paths contain an overlapping area. So that the same feature is not added to the database twice, it must first be checked whether the database already contains a descriptor with the same UTM position as the one of the current feature. If this is not the case, the descriptor vector and the UTM position of the matched feature extracted from the most recent RGB-image are stored into the database. The process of creating the database is illustrated in a block diagram shown in fig. 3.6. The steps are performed for each scene that should be covered by the database.

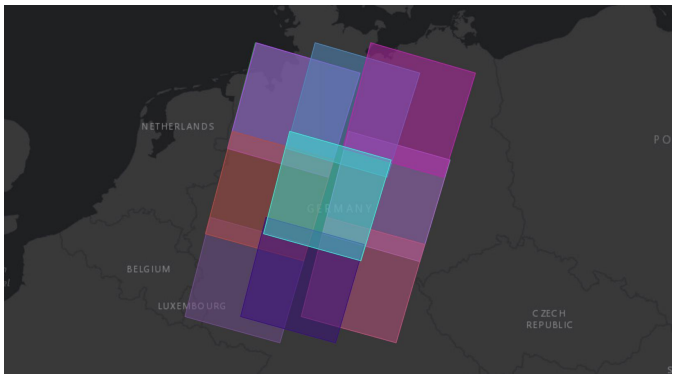


FIGURE 3.5: Illustration of the overlapping of the Landsat 8 image data from different scenes. Image taken on Earth Explorer [50].

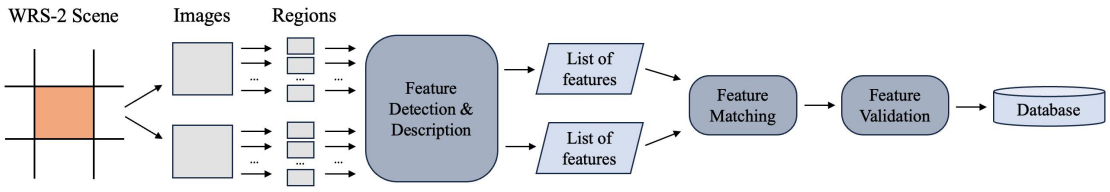


FIGURE 3.6: Block-diagram of creating the GCP database.

3.3.2 Onboard Feature Matching

After the GCP database is created for the selected scenes, the test application should then simulate the process of the onboard feature matching. Here, the main goal is to provide a reliable feature matching meaning that no false matches are selected and passed on to determine the satellite's attitude and position.

Feature Detection and Description
The first step is to obtain an image from the onboard camera. As well as for creating the database, RGB-images generated from the Landsat 8 data are used for this purpose. As the onboard camera captures a smaller area on ground due to its FOV than the one covered by a WRS-2 scene, a smaller region is cut out of the RGB-image. Again, the parameters of TOM are used leading to the following shape of the cut out region:

$$\text{width} = \frac{\text{swath width}}{\text{resolution}} = \frac{50\,\text{km}}{0.03\,\text{km}\,\text{px}^{-1}} \approx 1666\,\text{px}, \tag{3.8}$$

$$\text{height} = \frac{\text{swath height}}{\text{resolution}} = \frac{40\,\text{km}}{0.03\,\text{km}\,\text{px}^{-1}} \approx 1333\,\text{px}. \tag{3.9}$$

For the resulting RGB-image keypoints are extracted and described by the feature detection and description algorithm respectively.

Feature Matching
Next, these detected features are matched with the ones stored in the database. Following parameters are used for the matching process:

- Similarity Measure: Hamming distance for all binary feature description algorithms and Euclidean distance otherwise.
- Matching Strategy: TNN is used meaning that for each feature of the test image its nearest neighbor of the GCP features regarding descriptor distance is found. Additionally, the distance must not exceed a threshold value α that depends on the used description algorithm.

$$\left|\vec{d}_1 - \vec{d}_2\right| \leq \alpha \tag{3.10}$$

- Searching Technique: As high accuracy is needed and false matches could be fatal for attitude and position determination, the Brute-Force approach is used.

The resulting matches consists of a descriptor distance, a pixel position in the test image and an Earth position expressed in UTM coordinates.

Feature Validation
The overall goal of AVS is to cover the entire globe with GCPs. Thus, a huge amount of features have to be included in the database. Even though the feature detection algorithms should only extract unique appearances of the Earth's surface, covering a wide area 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 during the onboard feature matching, could cause false matches being included. In order to circumvent this difficulty an additional validation method is applied. It should guarantee that the resulting list includes all correctly selected matches but ignores all false ones.
The developed validation method is based on the RANSAC algorithm with the affine transformation as the underlying model described in section 2.9.1 and 2.9.2 respectively. Usually, RANSAC checks if the pixel positions of the matched keypoints detected in two different images correlate with each other. In AVS, the matching process does not find corresponding features of two images. But it matches the features extracted from the onboard image with the features of the database that only consists of a descriptor and a UTM position. Therefore, the RANSAC algorithm needs to be modified to be applicable for AVS. This is done by not checking whether the pixel positions of the matched features correlate, but instead by searching for the best affine transformation between the pixel positions and the corresponding UTM positions. If an affine transformation fits for more than 3 points, all these corresponding features have the same relative location in the pixel and the UTM coordinate system which indicates a high probability that these features are matched correctly. In detail, the algorithm comprises of the steps described in algorithm 1. $\vec{f}$, M and n denote a match consisting of a pixel position and a corresponding UTM position, the number of combinations consisting of three different matches and the number of matches in total respectively.
The runtime of this algorithm depends on the number of matches n to be analyzed. If after tresholding (see equation 3.10) the selected list of features contains any mismatches the abort criterion of line 15 never gets true and the algorithm computes and checks every possible affine transformation leading to $M = \binom{n}{3}$ iterations. For a fixed k the binomial coefficient increases exponentially with increasing n (see table 3.2). This could cause difficulties in terms of real-time applicability. Therefore, if after tresholding the number of selected matches exceeds 50, only the best 50 matches regarding descriptor distance are chosen.

k	**n**	$\binom{\mathbf{n}}{\mathbf{k}}$
3	25	2300
3	50	19600
3	75	67525
3	100	161700

TABLE 3.2: Illustration of increasing computational effort of proposed validation method: Examples for number of iterations for a fixed k and increasing n.

Algorithm 1 Modified RANSAC

$n_{inliers} \leftarrow 0$
$[\boldsymbol{R}, \vec{t}] \leftarrow [eye(2,2), \vec{0}]$
for $i \leftarrow 1 : M$ **do**
    Select systematically the next tuple of three matches $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$
    $[\boldsymbol{R_r}, \vec{t}_r] =$ computeAffineTransformation$(\vec{f}_1, \vec{f}_2, \vec{f}_3)$
    Compute the number of inliers n_r with respect to $\boldsymbol{R_r}$ and $\vec{t}_r$

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

    if $n_r == n$ **then**
        /* *Affine transformation fits to all matches* $\Rightarrow$ *all matches are correct* */
        break
    end if

end for

Unfortunately, the method has one main disadvantage. An affine transformation is computed based on 3 points and therefore always results in at least 3 inliers. If the best affine transformation returned by the algorithm only fits for 3 features, no statement can be made about whether the features are correct matches or not. In this case, for the application of AVS all inliers are discarded to ensure that no false matches are passed on to determining the satellite's attitude and position. In any other case, the feature validation results in a list of pixel and UTM position pairs. Additionally, this means that if the modified RANSAC algorithm is applicable and results in a set of matched features, the attitude and position can be surely determined as the set includes at least 4 features. Because the UTM system is not helpful for further computations regarding the ADCS, the Earth's position of the feature needs to be transformed to the ECEF coordinate system. As this work focuses on the procedure of feature analysis, one can refer for more details on this transformation to [62].
The entire procedure of feature detection and description, matching as well as validation described in this section is applied to each test image. An overview of this process is given in fig. 3.7.

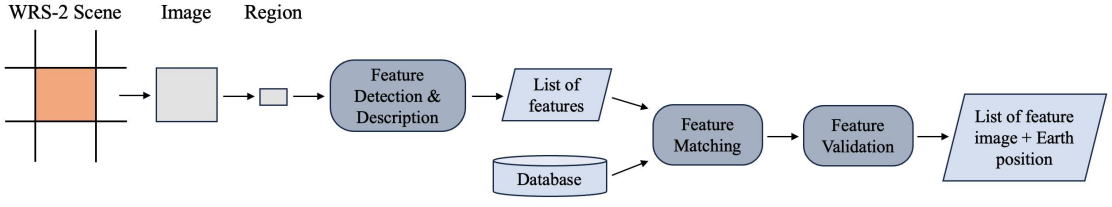


FIGURE 3.7: Block-diagram of onboard feature matching.

3.4 Implementation

For implementing the test application, *Visual Studio Code 1.80* is used as Integrated Development Environment with the programming language *Python 3.11*. As it is sufficient for the purposes of this application the database is implemented as an *.csv*-file for simplicity's sake. The software runs on a *Apple MacBook Pro 14" 2021* with a *Apple M1 Pro* chip and 16 GB unified memory. Additionally, *Open Source Computer Vision Library (OpenCV)* is integrated to get access to the implementations of the algorithms that are to be analyzed. 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*.

3.5 Selection of Algorithms

The choice of a well performing feature detection as well as description algorithm is crucial for the performance of AVS. The combination of these two algorithms should enable a fast process of distinct feature detection and unique feature description as a basis for the following matching. Unfortunately, no detailed studies on the comparison of different feature detection and description algorithms on satellite images could be found. As there exists a wide range of differences among the algorithms of how they extract and describe features, a broad investigation on a variety of combinations of algorithms has to be made.
The introduced algorithms in section 2.7 are among the current state-of-the-art methods. They have been compared in different combinations in multiple studies (such as [5, 31, 34]) and the results point out that each of them has some pros and cons depending on the requirements of the application in which they should be used. As SURF is an improved version of SIFT and ORB is a combination of FAST and BRIEF, only SURF, ORB and BRISK are further analyzed. Furthermore, an investigation of every possible combination would go beyond the scope of this thesis. Therefore, for each detection algorithm the combination with the included descriptor as well as the FREAK algorithm is analyzed. This leads to the combinations of detectors and descriptors shown in table 3.3 that are investigated in chapter 4. As the same features should be extracted from the onboard images and the image data used for creating the database, the same combination of algorithms is used for both processes. An implementation of each algorithm is accessible via the library *OpenCV*.

Detector	Descriptor
SURF	SURF
SURF	FREAK
ORB	ORB
ORB	FREAK
BRISK	BRISK
BRISK	FREAK

TABLE 3.3: Combinations of algorithms analyzed in the test application.

Chapter 4

Tests and Evaluation

As there exists a variety of different requirements for the process of AVS, several tests need to be carried out in order to find a well performing configuration of algorithms as well as to investigate the applicability of AVS in general. This chapter summarizes the executed tests as well as their results. Based on all findings, one certain combination of algorithms is suggested. Last, the constraints of AVS are considered and an outlook on future work is given.

4.1 Tests

Except the first one and section 4.1.7, each of the following subsections deals with a certain requirement for the process of feature detection, description, matching or validation regarding AVS and thus has the same structure. First, the investigated requirement is described. Then, the experimental setup is introduced. Afterwards, the metrics are described measuring the performance regarding the analyzed requirement. Last, the results are presented and evaluated.
As mentioned earlier, the main objective of AVS is a reliable onboard attitude and position determination during the entire orbit. To analyze the feasibility of this process for the entire globe, first the following tests are carried out on smaller areas of the Earth. Based on the performance and problems emerged, assumptions and constraints can be derived for the feasibility of AVS during the entire orbit. It needs to be mentioned that within the work on this thesis, only a limited number of scenarios can be tested during each test. Thus, all results are to be interpreted under this restriction. However, obvious trends regarding the performance can be seen that probably would be amplified by increasing the number of different test scenarios.

4.1.1 Adjustment of the Parameters

Each of the algorithms for feature detection and description includes several adjustable parameters which can influence their performance. The optimal selection of the parameters depends on the structure of the analyzed images as well as on the requirements of the application. Within this thesis, it is not feasible to carry out all the following tests with every possible combination of parameters of the different algorithms. Thus, the initial parameters are set in advance and get fine-tuned during the following tests.

To find well working parameters for satellite image data the basic structure of the test application described in chapter 3 is used. However, the onboard feature matching is designed a bit differently. This is because the modified version of the RANSAC algorithm applied during feature validation depends on the threshold value α used during feature matching. α depends on the selected algorithms whose performance in turn depends on their parameters. Therefore, after the onboard feature detection and description, the nearest neighbor for each detected feature is found without applying any additional matching or validation methods. The used database is created based on only one scene. The onboard test images are taken from another image of the same scene by using three different regions. The adjustment of the parameters for each algorithm is done by an iterative process starting with the default values provided by *OpenCV*. The iterative setting of the parameters is performed manually not using any distinct optimization method.
In order to find a strategy how to evaluate a particular parameter setting and to decide which parameter needs to be adjusted, it is first important to understand what the parameters affect in detail. A short description of each parameter is given in appendix B and for more context refer to chapter 2. In general, there exist three groups of parameters. The first one includes parameters that specify the desired uniqueness of the features and associated with it the number of detected keypoints. The more unique the detected features should be, the fewer keypoints may be returned. The parameters of the second group specify the size and structure of the keypoints. The last group contains parameters that personalize the proceeding of the descriptor algorithms, e.g. by specifying the patch size.
Regarding AVS, on the one hand, it is important to get a sufficient number of correctly matched features after the onboard matching. On the other hand, the database should contain as few features as possible, as the memory space and computational effort onboard the satellite increase with the number of stored features. Along with this, it is important to only store very unique feature descriptors so that the percentage of mismatches decreases. Therefore, throughout the iterative process of adjusting the parameters, attention is paid to the number of features stored in the database, to the number of detected features in the test images, to the number of correct matched features with the database as well as to the percentage of correct matched features from the number of detected features in the test image. Based on these four metrics, the parameters are optimized.
The resulting parameters are listed in appendix B. In summary, those of the second and third group mainly affect the percentage of correct matched features and thus can be optimized respectively. In contrast to this, the parameters of the first group influence the values for all four metrics as expected. Therefore, a suitable choice of these parameters is crucial for obtaining a sufficient number of correctly matched features onboard the satellite. However, as the thresholding and the modified RANSAC validation during the onboard feature matching yet cannot be included within this setup but affect the number of the resulting onboard matched features, no statement can be made about the best values for these parameters at this point. Therefore, these parameters have to be fine-tuned later as described in section 4.1.3. It is important to mention that it is not

guaranteed that the chosen parameters form the best configuration for all scenarios, but the best in this experiment.

4.1.2 Onboard Feature Validation

In the following test, the proposed method based on the RANSAC algorithm for validating the onboard matched features based on their image and UTM position is evaluated.

Requirement
One of the most important tasks during AVS is to reliably match the onboard detected features so that no mismatches are passed on to the attitude and position determination. To ensure this, as described in section 3.3.2, after matching each detected keypoint of the onboard image with its nearest neighbor of the database in terms of descriptor distance, two additional steps are performed: thresholding and validation by the proposed modified version of the RANSAC algorithm. In a desired scenario, these two methods only discard the mismatches and keep all the correctly selected features.

Experimental Setup
For analyzing the performance of the two methods, two experiments are executed. For the first experiment, three databases are created, each based on only one scene with size of approximately 240 km$\times$240 km. Afterwards, for each database another Landsat 8 image of the same scene is used to extract four onboard test images (regions 12-15, see fig. 3.3) resulting in 12 test scenarios. For each scenario, the onboard feature matching, as described in section 3.3.2, is performed by use of the corresponding database. To see how the performance of the methods might be influenced by the size of the covered area on ground of the database, a second experiment is executed. For this purpose, one database covering 12 different scenes (approx. 550 km $\times$ 700 km) is created and the same 12 test scenarios are run through. The used Landsat 8 data is referenced in appendix A.
As the performance of the proposed validation method does not depend on the applied detection and description algorithm, the two experiments are executed using ORB as detector and descriptor. As its parameters of the first group are not yet accurately adjusted, the following values are used in the upcoming tests: *fast threshold* = 50 and *Nr. features* = 3500. The descriptor threshold value α is set to 30.

Metrics
The performance of thresholding and of the proposed validation method is evaluated based on the following metrics:

- Number of features stored in the database located within the FOV of the test region (possible matches).
- Number of matches after standard matching, thresholding and modified RANSAC.
- Number and percentage of correctly selected matches after standard matching, thresholding and modified RANSAC.

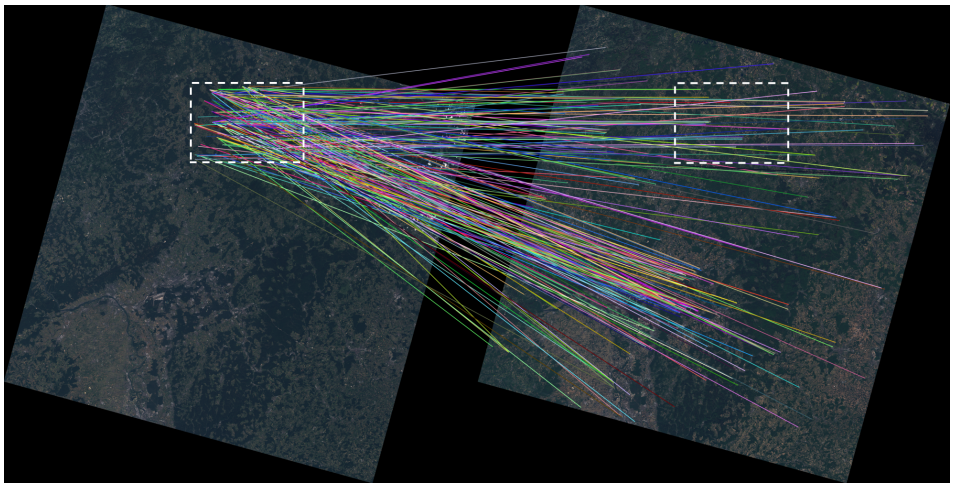
Image / region	Possible matches	After matching	Correct	After thresholding	Correct	After validation	Correct
1/12	93	1665	55	30	23	21	21
		3.3 %		76.7 %		100.0 %	
1/13	124	1268	44	15	13	13	13
		3.5 %		86.7 %		100.0 %	
1/14	1226	3268	908	50	50	50	50
		27.8 %		100.0 %		100.0 %	
1/15	343	2072	230	50	50	48	48
		11.1 %		100.0		100.0 %	
2/12	610	3500	489	50	50	49	49
		14.0 %		100.0 %		100.0 %	
2/13	185	2950	134	50	49	46	46
		4.5 %		98.0 %		100.0 %	
2/14	292	3123	246	50	50	49	49
		7.9 %		100.0 %		100.0 %	
2/15	633	3242	546	50	50	50	50
		16.8 %		100.0 %		100.0 %	
3/12	88	1238	25	7	3	0	0
		2.0 %		42.9 %		0.0 %	
3/13	86	1416	43	18	11	10	10
		3.0 %		61.1 %		100.0 %	
3/14	130	1236	61	24	23	21	21
		4.9 %		95.8 %		100.0 %	
3/15	149	1831	74	30	24	23	23
		4.0 %		80.0 %		100.0 %	

TABLE 4.1: Summary of the onboard matching results using a single scene for creating the database comparing standard matching, thresholding and validation via the modified RANSAC algorithm.

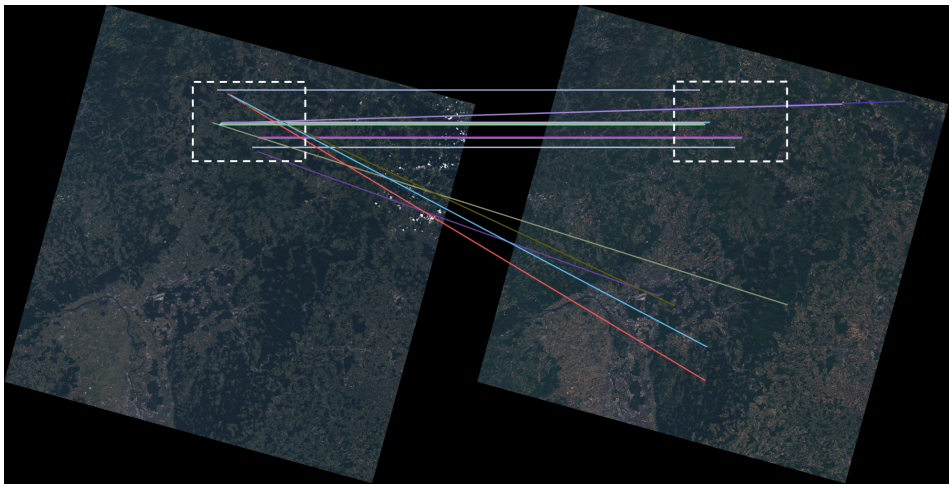
Results and Evaluation of Experiment 1

The resulting values for the metrics of the first experiment are presented in table 4.1. Each test run is referenced by its test image and region number. The databases for the test images 1, 2 and 3 contains 8694, 7222 and 3650 features respectively. By looking at the percentage of correctly matched features after the standard matching (avg. 8.6%) it gets clear that additional validation methods are necessary. After thresholding, this percentage can be greatly improved (avg. 86.8%) but is still not satisfying regarding a reliable feature matching. However, after validating the matches, in 11 of 12 test scenarios only correct matches are selected. Unfortunately, in 8 of these cases the validation method not only discards the mismatches but also correct ones. By an average rate of 95.9% of selecting all correct matches remaining after thresholding, the performance is still satisfying. Furthermore, by adjusting the approach in order to select all remaining correct features, it is more likely that the method then would also indicate mismatches as inliers. As this case has definitely to be avoided, it is acceptable that not all correct matches can be filtered out.

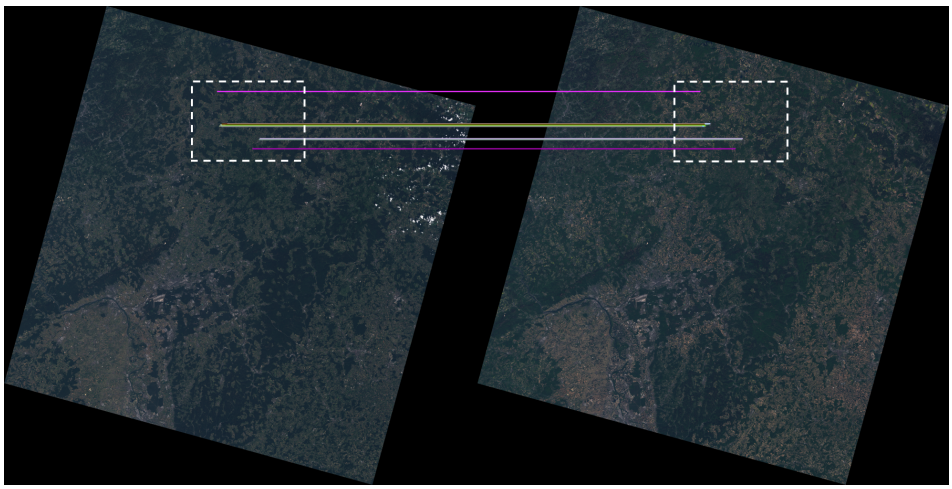
In the only test run which does not achieve 100% correctly selected matches (3/12), the application of the modified RANSAC algorithm results in an affine transformation which fits for three points. Even though these three inliers are exactly the correct



(A) Best 250 matches regarding descriptor distance after standard matching (55 of 1665 correct).



(B) Matches after thresholding (23 of 30 correct).



(C) Matches after validation (21 of 21 correct).

FIGURE 4.1: Illustration of the onboard matching results for test image 1, region 12. The colored lines connect the matched keypoints of the test image (left) and the image used for the database (right).

matches remaining after thresholding, no reliable statement can be made in this case, as described in section 3.3.2. Thus, no matches are selected.
In addition, it is noticeable that the number of remaining matches after validation strongly depends on both the number of detected features in the onboard image and the number of stored features in the database. In regions that contain many features stored in the database as well as detected in the onboard image, thresholding already achieves a high number and accuracy of correctly selected matches. In these cases, additional validation would not be necessary at all. In regions that contain only few features, however, the error rate after thresholding is significantly higher and can only be improved by validation. The problem associated with these regions gets even more obvious in the second experiment, where it is described in detail. These first results confirm that the application of an additional validation step is necessary and that the proposed algorithm ensures a reliable onboard matching process using a single scene database. Figure 4.1 illustrates the process for test run 1/12.

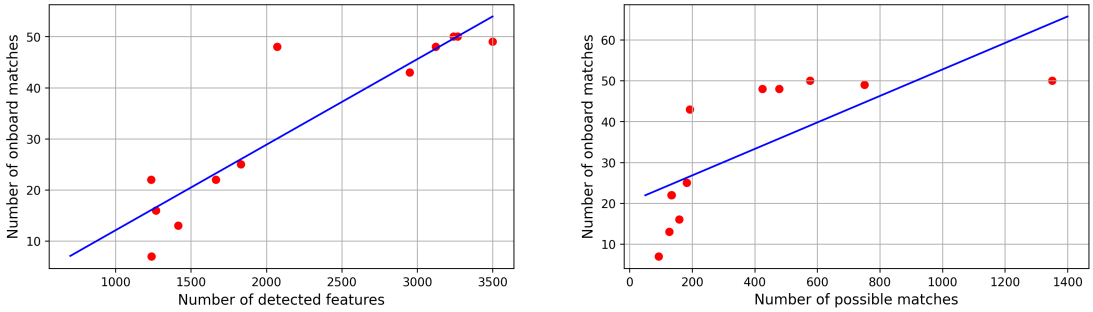
Results and Evaluation of Experiment 2
Table 4.2 presents the results of the second experiment using a database created based on 12 scenes. The database contains 57641 features. It is notable that the number of possible matches increases for each region which can be explained by the overlapping of the scenes. After the standard matching, the percentage of correctly selected matches is similarly low as in the first experiment (avg. 8.1 %). Applying thresholding increases this percentage (avg. 71.3 %), but it is significantly worse than in experiment 1. In general, it can be seen that the number of correctly selected matches after thresholding is similar or slightly increased compared to the first experiment. The increase is explained by the larger amount of possible matches. But now additionally, more wrong matches are included as well. This occurs primarily in regions where there are generally both fewer features stored in the database as well as detected onboard. This indicates an ground region with only few distinctive appearances in satellite imagery. These less distinctive features are more likely to be mismatched. Thus, with a larger database, only thresholding provides an increasingly larger rate of mismatches in these regions and an additional validation method is required. After validating the remaining features by applying the proposed method, in all cases only correct matches are selected. The higher rate of remaining mismatches does not seem to cause any difficulties. Similar to the first experiment, some correct matches get discarded as well. However, selecting on an average rate of 95.9 % of all remaining correct matches is satisfying.
In general, it also can be noticed that the number of matches after validation also increases compared to the first experiment due to the overlapping of the scenes. However, this number is still significantly lower in some regions than in others (see test image 2 vs. 3). This is particularly the case in regions where generally few unique features are detected onboard as well as stored in the database (see fig. 4.2). This is simply because some regions only have few unique appearances in satellite images. Therefore, fewer features can be detected and thus included in the database as well as matched onboard. Unfortunately, this circumstance cannot be resolved. As it has not yet been investigated how the number of resulting matches effects the accuracy of the attitude and position

Image / region	Possible matches	After matching	Correct	After thresholding	Correct	After validation	Correct
1/12	133	1665	56	42	25	22	22
			3.4 %		59.5 %		100.0 %
1/13	158	1268	49	26	16	16	16
			3.9 %		61.5 %		100.0 %
1/14	1351	3268	843	50	50	50	50
			25.8 %		100.0 %		100.0 %
1/15	478	2072	233	50	50	48	48
			11.2 %		100.0 %		100.0 %
2/12	751	3500	469	50	50	49	49
			13.4 %		100.0 %		100.0 %
2/13	192	2950	120	50	47	43	43
			4.1 %		94.0 %		100.0 %
2/14	424	3123	274	50	50	48	48
			8.8 %		100.0 %		100.0 %
2/15	577	3242	467	50	50	50	50
			14.4 %		100.0 %		100.0 %
3/12	93	1238	22	41	7	7	7
			1.8 %		17.1 %		100.0 %
3/13	127	1416	38	50	14	13	13
			2.7 %		24.0 %		100.0 %
3/14	134	1236	52	50	23	22	22
			4.2 %		46.0 %		100.0 %
3/15	182	1831	64	50	27	25	25
			3.5 %		54.0 %		100.0 %

TABLE 4.2: Summary of the onboard matching results using 12 scenes for creating the database comparing standard matching, thresholding and validation via the modified RANSAC algorithm.

determination in detail, it is difficult to predict whether these regions might cause difficulties.

It is also noticeable that only a small proportion of possible matches are found after standard matching (experiment 1 avg. 61.1 % , experiment 2 avg. 48.5 %). So many features located within a certain region are stored in the database, but apparently are not unique enough to be recognized onboard. Therefore, the idea is to only store the most unique features in the database which are more likely to be recognized by the standard matching, in order to avoid unnecessarily stored features and thus to keep the required computing time for the matching as well as the needed memory storage low. This is investigated in the following section. In summary, during this test it is shown that the proposed validation method is essential in order to ensure a reliable onboard matching process. The performance of the modified RANSAC algorithm regarding the number of selected matches and accuracy convinces.



(A) Number of detected features. (B) Number of possible matches.

FIGURE 4.2: Dependency between number of detected features (A) as well as possible matches (B) and number of onboard matches. The red points represent data of table 4.2. The blue line illustrates linear regression for datapoints.

4.1.3 Uniqueness of the Features

In the following test, the impact of manipulating the uniqueness of detected features, and therefore the impact of the number of stored features in the database, on the performance of the different combinations of detection and description algorithms is analyzed.

Requirement

In addition to the task of a reliable onboard feature detection, it is important to store as few features as possible in the database, due to limited resources onboard the satellite in terms of memory space and processing power. However, in order to still match a sufficient number of features onboard, the idea is to store only the features that appear to be most unique during the detection process of the database creation. These features are most likely to be detected in the onboard images and subsequently matched, too.

As described in section 4.1.1, there exists one parameter for each of the detection algorithms that sets the threshold at which score a pixel is identified as a feature. A first idea would be to increase the threshold to select only the most unique features. However, it could happen that regions which contain few to no features with a score above the threshold are then not covered by the database at all. Thus, onboard matching in these regions would be ruled out in advance. However, as the goal is to ensure that there are features stored in the database over the entire desired area, this approach could cause problems.

Another idea is to set a low value for the threshold and to select the best n features for each region after the detection process during the creation of the database. This ensures that a certain amount of features are detected from each region and thus that the entire area is covered during feature detection. However, this does not also ensure that the entire area is covered in the database, as these detected features also have to be matched with the features of the second image of the same scene. During the onboard matching the best $1.2 \cdot n$ features are selected. If a feature that is stored in the database is detected in the onboard image, but with a lower score, the additional $20\,\% \cdot n$ selected features should enable a matching anyway. This idea is implemented and tested in the following.

Experimental Setup
To analyze the impact of the number of selected features n on the performance of the six different combinations of algorithms, six different values for n are chosen: 100, 400, 700, 1000, 1300 and 1600. Therefore, for each combination of algorithms six databases are created resulting in 36 databases in total. Each database is based on 12 scenes. Afterwards, for each database the onboard feature matching is executed for six test scenarios (region 13 and 14 of three test images). The used Landsat 8 data is the same as in the test of 4.1.2 (see appendix A.1). The thresholds as well as all other parameters of the detection and description algorithms are summarized in appendix B. The parameter alpha of the different descriptor algorithms needed for thresholding during the onboard matching process, is fine-tuned in advance. Its values can be found in appendix B.

Metrics
The metrics to evaluate the performance of the different algorithms for the values of n are the following:

- Number of stored features in the database.
- Number of selected matches after onboard matching.
- Percentage of correct selected matches after onboard matching.

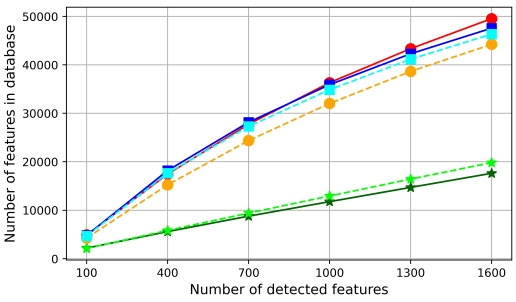
Results and Evaluation
The number of features stored in the databases for the different values for n is listed in table 4.3 as well as plotted in fig. 4.3 (A). Figures 4.4 to 4.9 show the number of selected matches (A) as well as the percentage of correct matches after onboard matching (B) for each test scenario.
First, the number of features of the respective database is considered. For both combinations using the SURF detector, an approximately linear increase in the number of features with a higher value for n can be seen. For the other four combinations, the gradient slightly decreases with higher values for n, but the number of stored features increases continuously. The approximate constant increase of the curves means that after including less unique features, matching still works well and that these features are recognized correctly. When creating the database, features are matched solely based on their nearest neighbor regarding descriptor distance. Thus, as all algorithms detect the same number of features for a given n, a higher number of features in the database generally indicates a better combination of algorithms that either detects more distinctive features or describes them more uniquely. In this context it is noticeable that there is a big difference of the number of stored features between the different algorithms. In general, using the own descriptor rather than the FREAK algorithm for each detector leads a similar amount of features in the database. In this case, the difference is the biggest for ORB. Furthermore, it can be seen that both combinations including SURF results in a significantly lower number of features. This presumes that using the SURF detector could also result in less features matched onboard.
In the following, the curves of the number of onboard matches (A) and the percentage of correct matches (B) are observed. The curves representing the percentage of correct matches show that 100 % of the selected matches after validation are correct in all cases

Detector-Descriptor	Number of detected features n					
	100	**400**	**700**	**1000**	**1300**	**1600**
ORB-ORB	4913	17448	27740	36329	43339	49542
ORB-FREAK	4257	15243	24372	32033	38648	44229
BRISK-BRISK	4803	18187	28162	35910	42288	47569
BRISK-FREAK	4662	17653	27309	34871	41160	46351
SURF-SURF	2179	5589	8771	11766	14707	17616
SURF-FREAK	2103	5822	9405	12934	16422	19825

TABLE 4.3: Number of stored features in the databases created in test 4.1.3, depending on used detector, descriptor and number of selected features n during the feature detection process.

except of one: SURF-SURF in test image 3, region 4. In this test scenario, the modified RANSAC algorithm selects up to five matches including no correct match at all. In this test run, no correct matches remain already after thresholding. As by chance the relative pixel position of at least four features fits the relative location of the matched UTM positions, an affine transformation exists which can map these pixels and UTM positions. Thus, the RANSAC algorithm wrongly selects these matches. Looking at the matched features in the corresponding satellite imagery (fig. 4.10), it can be noticed that they are actually located several hundred kilometers apart. The larger the database gets, the less likely is that by chance the relative location of the pixel positions and the matched UTM positions map. However, a certain residual risk always exists, especially if there exist regions that are very similar on satellite images over a larger area in terms of the description of detected features. Furthermore, this case only occurs if the detected features are not very distinctive or are not described uniquely enough by the descriptor algorithm, so that after thresholding almost only mismatches remain. The results let assume that the SURF algorithm is not a suitable choice for the application during AVS.

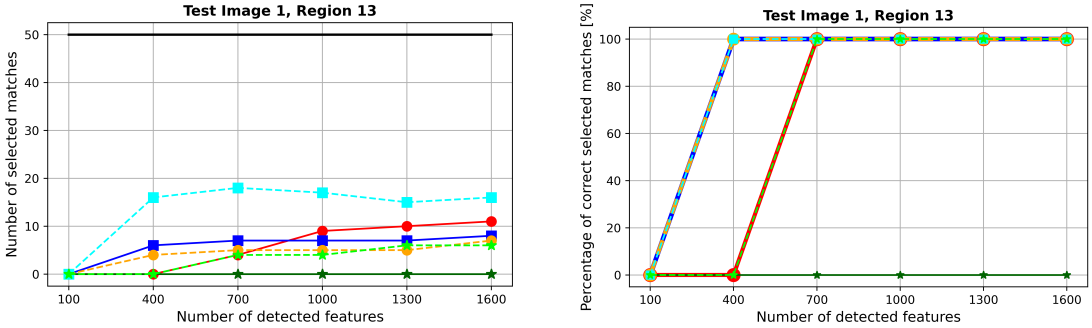
With regard to the number of matches, it is noticeable that in most cases with few selected features ($n = 100$), few to no features are matched onboard. Afterwards, a strong increase is observed up to $n = 400$ and sometimes up to $n = 700$. Unlike the number of features in the database, this flattens out sharply beyond, and the number of matched features hardly increases any further. In some cases, the number of selected matches even decreases as n increases. This can be explained by the fact that with larger values



(A) Number of features in database.

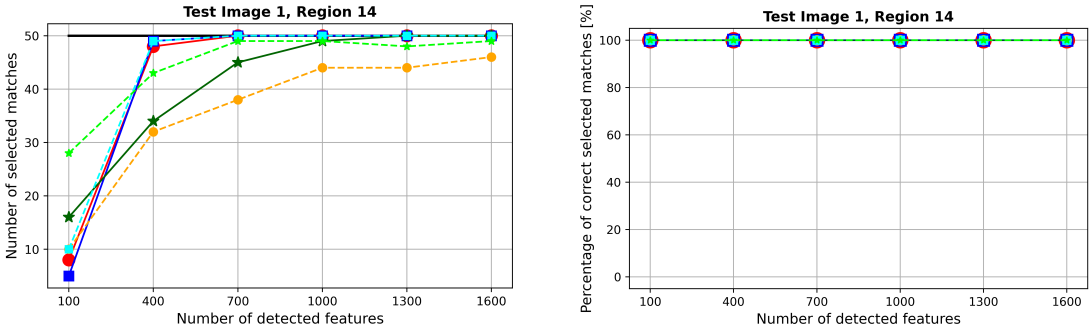
(B) Legend for all plots in 4.1.3.

FIGURE 4.3: Number of features stored in the databases for different values of n in (A) and the legend for all plots in test 4.1.3 in (B).



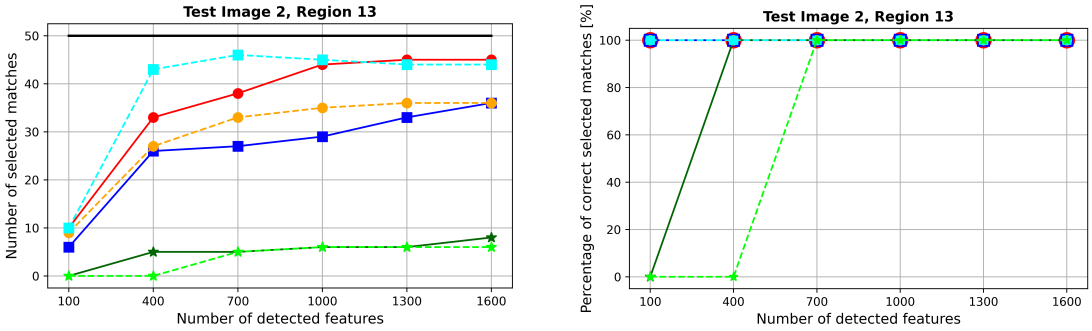
(A) (B)

FIGURE 4.4: Results of test image 1, region 13.



(A) (B)

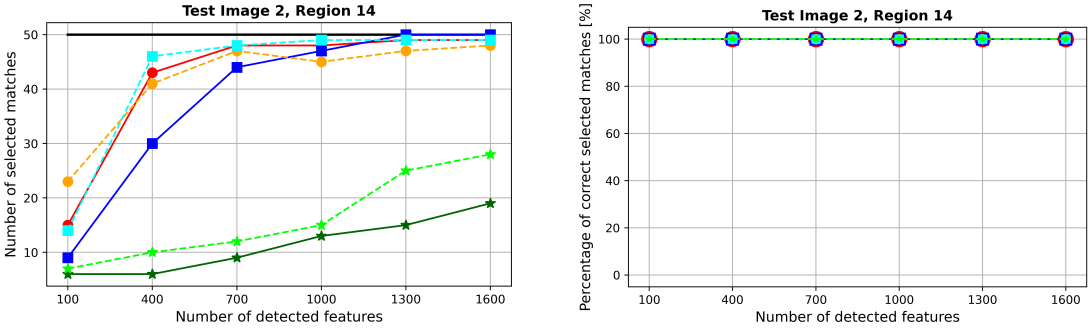
FIGURE 4.5: Results of test image 1, region 14.



(A) (B)

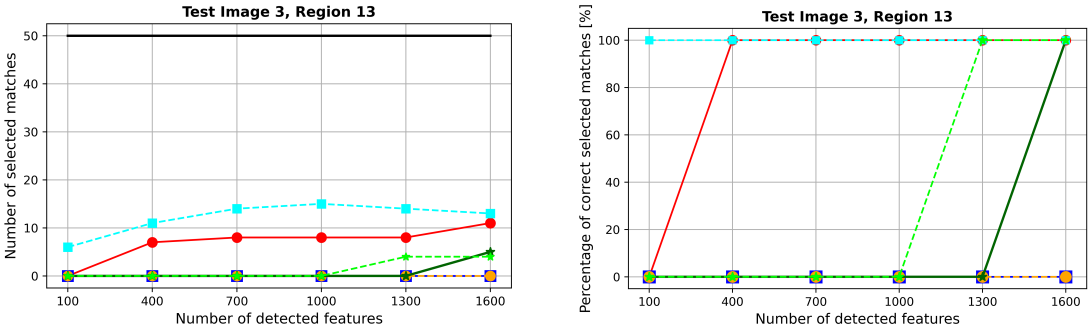
FIGURE 4.6: Results of test image 2, region 13.

for n, the number of features in the database also increases, which leads to a higher probability of mismatches. Although these mismatches are reliably sorted out by the modified RANSAC algorithm in most cases, they partially reduce the number of correct matches that remain after thresholding and therefore after validation as well.
Again, it has yet not been investigated how the number of matches affects the accuracy of determining the attitude and position. Thus, it cannot be predicted whether the amount of matches at a value of $n = 400$ or $n = 700$ is sufficient or if the very few additional matches resulting at higher values for n could improve the accuracy significantly. Currently, this experiment only shows to what extent the longer computation time during onboard matching and the higher memory requirements are worthwhile



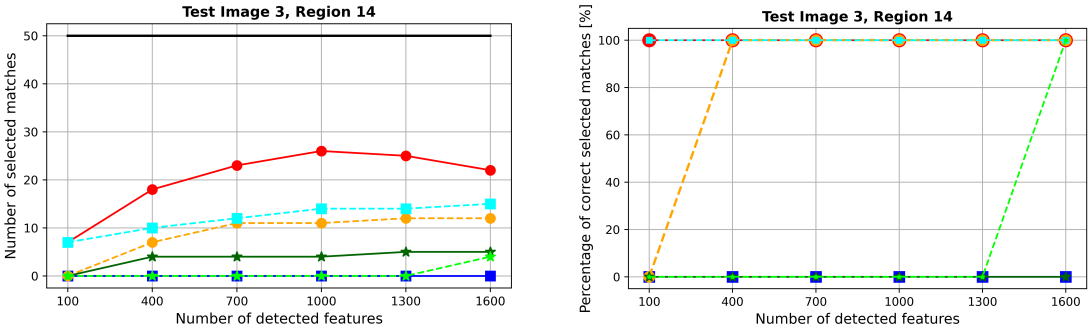
(A) (B)

FIGURE 4.7: Results of test image 2, region 14.



(A) (B)

FIGURE 4.8: Results of test image 3, region 13.



(A) (B)

FIGURE 4.9: Results of test image 3, region 14.

regarding to a higher number of resulting matches and not regarding to a higher accuracy during attitude and position determination. In conclusion, the best performance with respect to a reliable onboard matching, while keeping the database as small as possible, is achieved by $400 \leq n \leq 700$ regardless of the used algorithms. Smaller values for n lead to a significantly smaller number of matched features. Larger values hardly lead to any added value but to significantly more computational effort and required memory space due to the larger database. In some cases, it even results in a smaller number of resulting matches.

Lastly, the performance of the different algorithms is considered in detail. First, it is notable that the performance strongly depends on the region displayed in the onboard

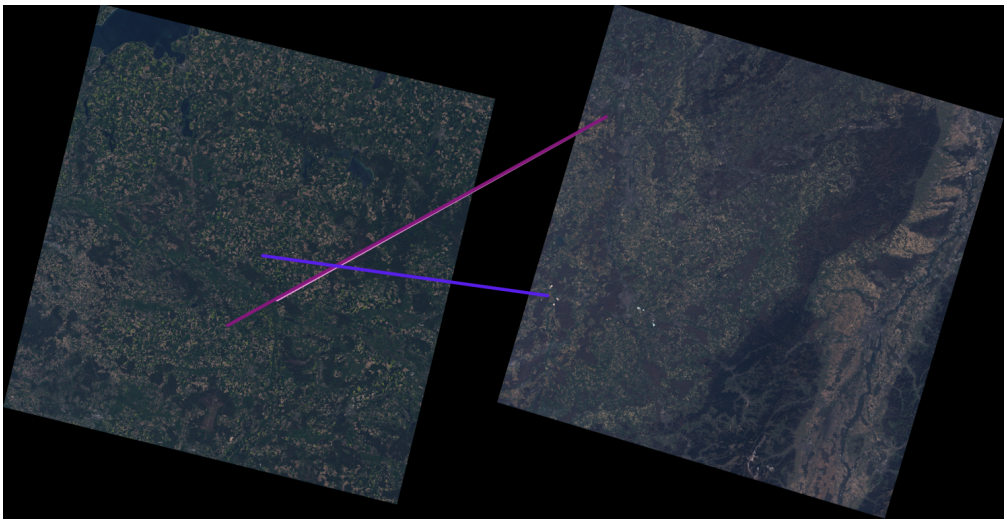


FIGURE 4.10: Illustration of the four selected matches after onboard validation for test image 3, region 14 using SURF-SURF combination at $n = 700$. The left image shows the detected features in the test image (path 194, row 023). The right image represents the part of the database where the matched features are located (path 196, row 025). The pink line overlays the fourth match due to the close location of the matched features.

image. In some regions, a combination of algorithms selects many matches (e.g., BRISK-BRISK in fig. 4.5 (A): max. 50 matches). In other regions, onboard matching with the same combination results in only few to no matches at all, independent of the value of n (e.g., BRISK-BRISK in fig. 4.9 (A): constant 0 matches). These results confirm, as also seen in section 4.1.2, that there exist regions which, according to the algorithms, contain only few unique features in the satellite images and that the performance of the onboard matching decreases drastically in terms of number of matched features within these regions. Changing the number of detected features n and thus the size of the database does not lead to better results in these areas. At this point it is unclear if or how this problem could be solved.

Additionally, it is observed that the performance within a region also strongly depends on the used algorithms. As pointed out before, all combinations provide a reliable onboard matching by only selecting correct matches in all cases except of the combinations using SURF detector. However, the different algorithms lead to a different amount of selected features as can be seen in table 4.4 showing the average number of selected matches for $n = 400$ and $n = 700$. It is noticeable that SURF provides significantly less matches after validation than the other four combinations of algorithms (approx. 2 times less than ORB-FREAK and BRISK-BRISK and approx. 3 times less than ORB-ORB and BRISK-FREAK). This is the case using SURF's own as well as the FREAK descriptor. This behavior is consistent with the low number of stored features, indicating less unique features detected using SURF. However, for the other combinations, the results of the onboard matching behave differently according to the resulted number of features in the respective database. Even though a similar number of features is stored in these cases, the resulting number of selected matches varies significantly. The combination of the BRISK detector and the FREAK descriptor achieves most matched features,

Detector-Descriptor	Number of detected features n	
	400	**700**
ORB-ORB	24.8	28.5
ORB-FREAK	18.5	22.3
BRISK-BRISK	18.5	21.3
BRISK-FREAK	29.2	31.3
SURF-SURF	8.2	10.5
SURF-FREAK	8.8	11.0

TABLE 4.4: Average number of selected matches during test 4.1.3, depending on used detector, descriptor and number of selected features during the feature detection process.

especially significantly more (approx. 31.9 %) than using the BRISK descriptor. The second most matches are achieved using the combination of ORB detector and descriptor, selecting slightly more (approx. 21.8 %) features than using the ORB detector combined with FREAK. It is also important to mention that the combinations ORB-ORB and BRISK-FREAK are able to match a high number of features within regions where the other combinations struggle (test image 3, region 13 and 14). An explanation why these certain combinations are able to select significantly more matches could not be found.
So, in summary, the test provides the following findings. A reduction of detected features during database creation to the range of 400 to 700 per region turns out to be the best trade-off between database size and a reliable onboard matching. An exact best value for n is not determined based on the results of this experiment as it is only ensured that n is in between 400 and 700. Therefore, for the following tests, n is set to 700. Furthermore, it needs to be investigated how the number of matches influences the accuracy of attitude and position determination, so that the value of n is also satisfying for this requirement. In ground regions with few distinctive features, only few matches continue to emerge. The best results in the performed test scenarios regarding number of matches as well as a reliable matching process are achieved by the combinations BRISK-FREAK and ORB-ORB with BRISK-FREAK requiring less features stored in the database. The combinations with SURF result in significantly less matches, including mismatches in some cases.

4.1.4 Invariance to Rotation

In the following test, the impact of the viewing angle of the onboard camera on the performance of the different detector and descriptor combinations is investigated.

Requirement
During all previous tests, the test image used during the onboard matching is aligned exactly like the images used for database creation. In orbit the satellite might not be orientated as the Landsat 8 satellite and thus the onboard image might be rotated compared to the Landsat 8 image data even if both satellites are in nadir-pointing mode. Therefore, to ensure a reliable and sufficient onboard matching at all rotation angles of the satellite the detection and description algorithms need to be invariant to rotation. This means that the same features need to be detected and described by the same

descriptor vectors in images that show the same scene but are taken from a different viewing angle. If that is the case, the number of selected matches onboard does not change if the onboard image is rotated. Otherwise, matching of a sufficient number of features could get difficult especially in regions where only few unique features appear in satellite image data and thus are stored in the database. In principle, all used algorithms should ensure rotation invariance, as described in section 2.7. In the following, it is tested, whether there are algorithms however which are more sensitive to rotation.

Experimental Setup

In order to analyze the impact of the rotation angle on the performance of each combination of algorithms, the onboard matching is executed for six test scenarios (region 13 and 14 of three test images). During this experiment, it is assumed that the satellite to perform AVS is in nadir-pointing mode and is only rotated around the nadir axis compared to the Landsat 8 satellite. Therefore, each onboard image is rotated counterclockwise by 0, 90, 180 and 270 deg. The databases for the respective combinations of algorithms are taken from the previous test of section 4.1.3 using $n = 700$. The used Landsat 8 data is the same as in section 4.1.3 and is listed in appendix A.1.

Metrics

To evaluate the impact of the rotation angle, the mean value and standard deviation of the number of selected matches after onboard matching over all four rotations are considered.

Results and Evaluation

The number of selected matches for the different rotation angles for each test scenario are plotted for each algorithm in fig. 4.11. Table 4.5 summarizes the mean value and standard deviation of the number of selected matches.

First, it needs to be mentioned that SURF-SURF in test scenario 3/14 and SURF-FREAK in test scenario 1/13 only select wrong matches (indicated by red font in table 4.5). Again, this suggests that SURF is not a good choice for reliable matching. Looking at

Detector-Descriptor	**Test scenarios (image/region)**						**Avg. deviation in %**
	1/13	**1/14**	**2/13**	**2/14**	**3/13**	**3/14**	
ORB-ORB	4.0 ± 0.0	49.5 ± 0.5	38.0 ± 1.2	46.0 ± 2.5	8.5 ± 0.5	23.0 ± 0.7	3.1 %
ORB-FREAK	4.5 ± 0.5	39.5 ± 2.6	30.5 ± 1.8	41.0 ± 3.7	0.0 ± 0.0	13.3 ± 1.3	7.1 %
BRISK-BRISK	2.8 ± 2.9	50.0 ± 0.0	19.3 ± 5.9	36.8 ± 4.3	0.0 ± 0.0	2.3 ± 2.3	39.8 %
BRISK-FREAK	15.8 ± 2.3	50.0 ± 0.0	41.5 ± 3.8	49.0 ± 0.7	10.5 ± 2.1	8.3 ± 2.6	12.7
SURF-SURF	0.0 ± 0.0	45.3 ± 0.4	5.0 ± 0.0	9.0 ± 0.0	0.0 ± 0.0	*4.0* ± *0.0*	0.1 %
SURF-FREAK	*2.0* ± *2.0*	48.5 ± 0.5	5.0 ± 0.0	11.5 ± 0.5	0.0 ± 0.0	0.0 ± 0.0	17.6 %

TABLE 4.5: Mean and standard deviation of number of selected matches using different rotation angles in test 4.1.4, depending on used detector, descriptor. Values in red font indicates that the selected matches in these test scenarios include mismatches.

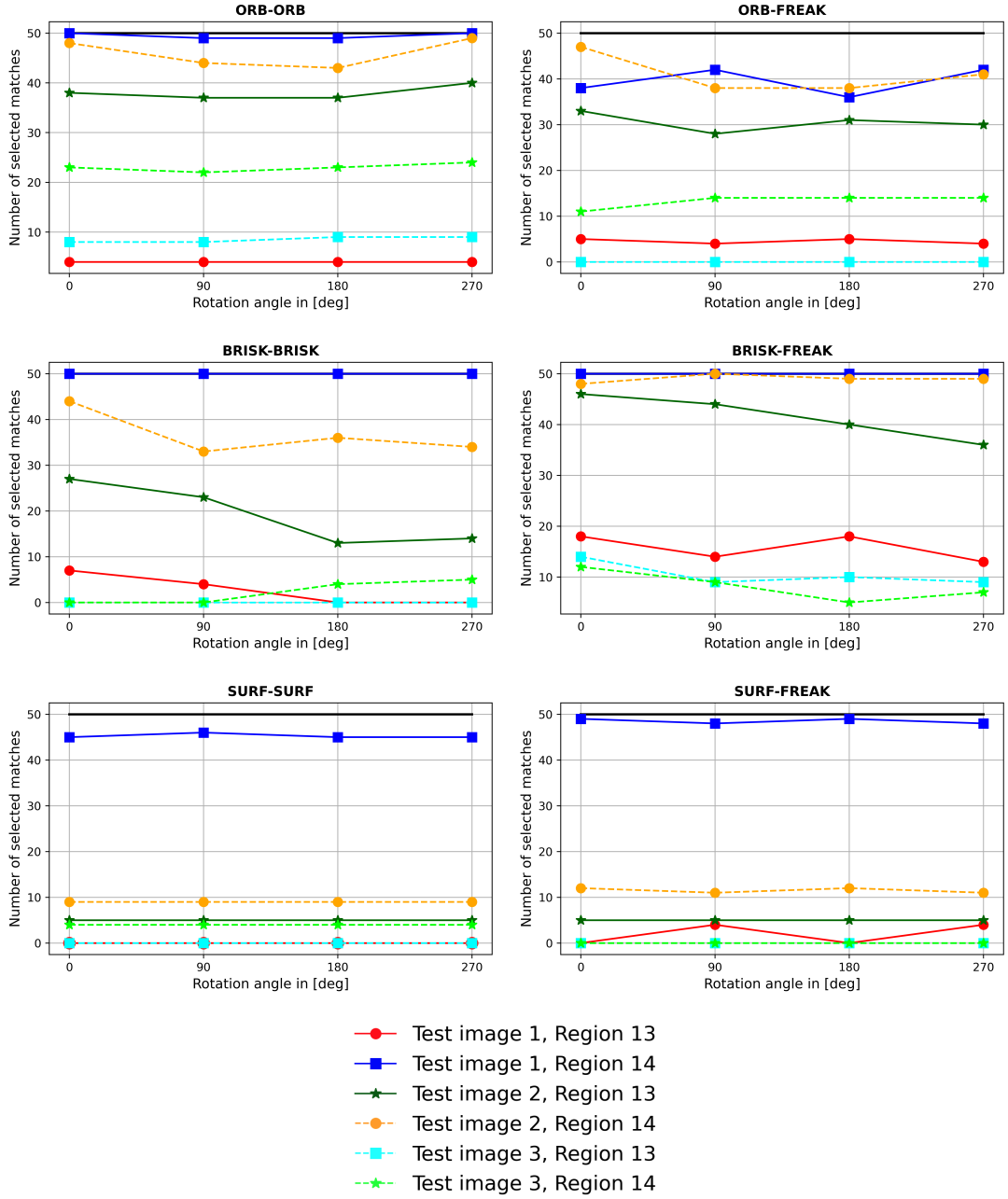


FIGURE 4.11: Plot of the number of selected matches after onboard matching for different rotation angles using the different combination of algorithms. Each line represents a test scenario specified by the test image and region.

the plots in fig. 4.11, it is noticeable that the matches obtained vary greatly for some combinations but are very constant for others. According to desire, all curves are parallel, as the same number of matches is selected regardless of the image's orientation. The values using ORB-FREAK, BRISK-BRISK as well as BRISK-FREAK vary particularly strongly. In contrast, the results of ORB-ORB, SURF-SURF and SURF-FREAK appear to be robust against rotation. This assumption is largely confirmed by the calculation of the standard deviation in table 4.5. A low average percentage deviation indicates a constant number of selected matches across all rotation angles. In contrast, a high value

suggests a rotation-sensitive combination of algorithms. However, although the SURF-FREAK curves indicate that the combination is robust, its average deviation appears rather high. This can be explained due to the outlier in test scenario 1/13. In general, however, this combination seems to be very robust against rotations. In summary, indeed, there are differences regarding the robustness against rotation for the different algorithms. Especially the combinations of the BRISK detector are very sensitive. The combinations of the ORB detector are significantly more robust and the combinations of the SURF detector perform best. But it is important to emphasize that in both combination using the SURF detector, false matches are included in two test scenarios. Furthermore, it needs to be mentioned that this test only covers the rotation around the nadir axis. However, the satellite to perform AVS might be rotated not only around the nadir axis but around all three axis. The impact of further rotations on the performance of AVS needs to be investigated in future work.

4.1.5 Memory Space

In this section, the required memory space of the database for each combination of algorithms is investigated.

Requirement
As there is only limited memory space available onboard the satellite, the memory requirements of the GCP database must be kept as low as possible. Each entry of the database consists of the feature descriptor vector and its ground position. The memory requirement of the database is affected by the number of stored features as well as by the size of the descriptor vector. The position of a feature is stored using two float32 values needing 4 bytes each. As mentioned before, the different databases contain different numbers of features. Furthermore, the dimensions and data types of the descriptor vectors differ from each other, as can be seen in table 4.6, which leads to different memory requirements for a single vector for each descriptor algorithm.

Experimental Setup and Metric
To compare the memory requirements of the different combinations of algorithms, the databases created in test 4.1.3 for $n = 700$ are used. These databases are based on 12 scenes covering an area of approximately 550 km $\times$ 700 km = 385 000 km^2. For each of these six databases, the required memory space is calculated. Based on this, the memory space required for the databases covering the entire land area of the Earth (149 430 000 km^2 [61]) is estimated. The reason why the water surface of the earth is neglected is explained in section 4.1.8.

Descriptor	Dimension	Data type	Memory space
ORB	32D	uint8	32 B
BRISK	64D	uint8	64 B
SURF	64D	float32	256 B
FREAK	64D	uint8	64 B

TABLE 4.6: Memory space for different descriptor vectors.

Detector - Descriptor	Number of features	Memory space for			
		one descriptor	all descriptors	ground positions	entire database
ORB-ORB	27740	32 B	0.89 MB	0.22 MB	1.11 MB
ORB-FREAK	24372	64 B	1.56 MB	0.19 MB	1.75 MB
BRISK-BRISK	28162	64 B	1.80 MB	0.23 MB	2.03 MB
BRISK-FREAK	27309	64 B	1.75 MB	0.22 MB	1.97 MB
SURF-SURF	8771	256 B	2.25 MB	0.07 MB	2.32 MB
SURF-FREAK	9405	64 B	0.60 MB	0.08 MB	0.68 MB

TABLE 4.7: Memory space for 12 scenes databases using different combinations of algorithms.

Results and Evaluation

Table 4.7 lists the required memory space for the database based on the 12 scenes. The estimated storage needed for the entire land area of the Earth is given in table 4.8. It is noticeable that the required memory space of the descriptor vector has the biggest impact on the database storage needed. Furthermore, it can be seen that there are big differences between the combinations. The database of SURF-SURF needs significantly more storage than any other combination, even though it contains the fewest features. This is because of the large memory space required by a single SURF descriptor vector. The SURF-FREAK database contains similar few features, but as the descriptor vector of FREAK only requires a quarter of the storage compared to the one of SURF, this database needs by far the least memory space. Due to the higher amount of included features using ORB-FREAK and BRISK-FREAK, their databases require 3 times as much storage as FREAK combined with SURF but still significantly less (approx. 20 %) than using SURF-SURF. The descriptor vectors created by BRISK need equal memory space as the ones of FREAK. As the database of BRISK-BRISK also keeps a similar number of features as those of ORB-FREAK and BRISK-FREAK, the required storage is similar, too. The descriptor vector of ORB only needs half of the memory space as those of FREAK or BRISK. Therefore, the database of ORB-ORB only requires approximately half the storage as those of ORB-FREAK, BRISK-BRISK and BRISK-FREAK as they all keep a similar number of features.

In summary, the combination of ORB-ORB provides the best performance regarding memory space, as due to the low required storage of a single descriptor vector, a high number of features can be contained in the database, while still keeping the memory

Detector - Descriptor	Number of features	Memory space for			
		one descriptor	all descriptors	ground positions	entire database
ORB-ORB	$10.8 \cdot 10^6$	32 B	345.6 MB	86.4 MB	432.0 MB
ORB-FREAK	$9.5 \cdot 10^6$	64 B	608.0 MB	76.0 MB	684.0 MB
BRISK-BRISK	$10.9 \cdot 10^6$	64 B	697.6 MB	87.2 MB	784.8 MB
BRISK-FREAK	$10.6 \cdot 10^6$	64 B	678.4 MB	84.8 MB	763.2 MB
SURF-SURF	$3.4 \cdot 10^6$	256 B	870.4 MB	27.2 MB	897.6 MB
SURF-FREAK	$3.7 \cdot 10^6$	64 B	236.8 MB	29.6 MB	266.4 MB

TABLE 4.8: Estimated memory space for entire databases using different combinations of algorithms.

space of the entire database low. The SURF descriptor might not be the optimal choice for the application of AVS due to its memory-intensive descriptor vector.

4.1.6 Runtime

In the following tests, the runtime of the onboard feature matching using the different combinations of algorithms is investigated.

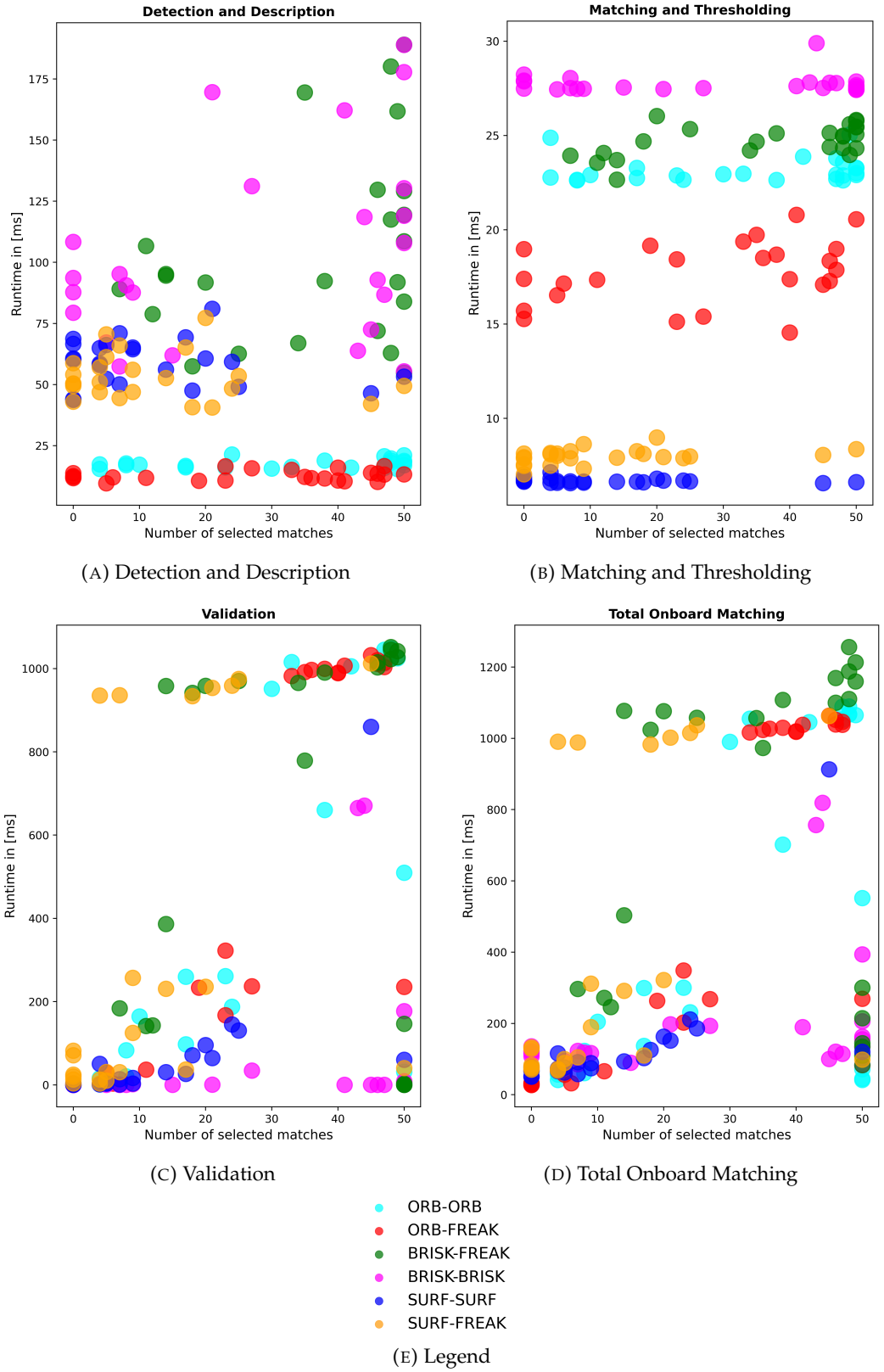
Requirement
Determining the attitude and position onboard the satellite is a time-critical task and thus the used methods need to be real-time capable. Therefore, the runtime of the onboard feature matching process is an important factor to decide if and how AVS could be integrated in future missions and to decide which combination of algorithms should be chosen.

Experimental Setup and Metric
For comparison of the different combinations, 24 test scenarios (region 6-9 and 12-15 of three scenes) are executed for each combination and the runtime for each part of onboard feature matching is measured. The databases created in test 4.1.3 for $n = 700$ are used.

Results and Evaluation
The runtime for each part of the onboard feature matching depending on the number of matches selected after validation during the test scenarios is shown in fig. 4.12. Each datapoint represents one test scenario. The color of the datapoint illustrates the used combination of algorithms. The mean values of the runtime for each step are summarized in table 4.9. For better understanding of the results of the total runtime, at first each step is analyzed individually.
It is noticeable in both fig. 4.12 and table 4.9 that the runtime of the feature detection and description mainly depends on the used detection algorithm and that there are only small differences between using the own or the FREAK descriptor. This indicates that describing $1.2 \cdot n = 840$ features contributes with a much smaller share to the total runtime compared to the detection process. Therefore, the differences caused by the used descriptor hardly matter. It is noted that the combinations of the ORB detector take by far the shortest (about 7 and 4 times faster than those with BRISK detector and

Detector - Descriptor	**Runtime**			
	Detection + Description	**Matching + Thresholding**	**Validation**	**Total**
ORB-ORB	17.61 ms	23.16 ms	519.20 ms	(559.96 ± 456.13) ms
ORB-FREAK	12.75 ms	17.73 ms	554.32 ms	(584.81 ± 457.30) ms
BRISK-BRISK	104.43 ms	27.74 ms	65.22 ms	(197.39 ± 189.03) ms
BRISK-FREAK	104.42 ms	24.72 ms	615.62 ms	(744.79 ± 444.94) ms
SURF-SURF	59.07 ms	6.69 ms	66.09 ms	(131.85 ± 168.23) ms
SURF-FREAK	53.18 ms	7.98 ms	330.59 ms	(391.76 ± 404.04) ms

TABLE 4.9: Mean values of the runtime for different steps of the onboard feature matching process: Feature detection and description, matching and thresholding, validation as well as the mean value and standard deviation of the total runtime.



(A) Detection and Description

(B) Matching and Thresholding

(C) Validation

(D) Total Onboard Matching

(E) Legend

FIGURE 4.12: Plot of the runtimes of each test scenario for each step of the onboard matching: Detection and description (A), matching and thresholding (B), validation (C) and total runtime (D). The different colors of the datapoints represent the combinations of algorithms (E).

those with SURF detector respectively). The combinations using the BRISK detector are about 2 times faster than those with SURF.
It can also be seen that the BRISK datapoints are very scattered. This is the case because the number of detected features of the BRISK detector also varies significantly more than by using the other detectors. In general, in some regions hardly any features are detected, even less than by the other detectors. But in other regions significantly more features are found which then leads to more computational effort. In addition, it is noticeable that the outliers of the datapoints are mainly present at a higher number of selected matches. This is explained by the fact that features from regions with many, very unique appearances in satellite imagery are also more likely to be matched. However, as so many features can be found in these regions by the BRISK detector, the detection also takes more time. To prevent detecting too many features, the parameter of the algorithm, that indicates at which score a pixel is identified as a feature, could be raised. However, as it needs to be ensured that at least 840 features are found in each region, and as the BRISK detector detects only very few features in some regions, this approach is not feasible.
In general, it needs to be mentioned that the runtime of the detection could still be improved for all combinations. This is because the parameters, that specify at which score a pixel is indicated as a feature, are set for test 4.1.3 so that at least $1600 \cdot 1.2 = 1920$ features are detected in each region. As now $n = 700$, this parameter could be readjusted for each detector to keep the total number of detected features as low as possible. But this would lead to better results for all detectors and the focus of this test is not on absolute time, but on the relative performance of the different combinations. In addition, it is shown later that the runtime of feature detection only forms a small part of the total runtime and a minimal improvement is of little effect overall. Therefore, this is not pursued further within this thesis.
Furthermore, as already described, there are small differences between the use of the own and the FREAK descriptor. FREAK is on average 27.6 % faster than ORB, although the ORB descriptor vector contains only half as many entries. There is no difference regarding runtime between the BRISK and FREAK description algorithms. Additionally, FREAK also is on average 10.0 % faster than the SURF descriptor.
Looking at the runtimes of matching and thresholding, distinct differences between the combinations are notable. During matching, for each of the 840 described features the nearest neighbor in the database is determined. Therefore, the runtime mainly depends on the number of features in the database k, as $840 \cdot k$ distances are calculated and compared. Thus, the larger the database, the more computational effort is required. This explains the differences in runtime for the combinations when looking at the number of features stored in the respective databases in table 4.3. The combinations with SURF are 2-4 times faster than the others because of the few features stored. The runtime of ORB-FREAK is about 30 % shorter than that of the three remaining combinations as it keeps about 3000 less features. The differences in the number of stored features for the remaining algorithms are very small. Therefore, they all have similar runtimes, too. As expected, BRISK-BRISK needs the longest for the matching process. However, it is noticeable that ORB-ORB performs slightly faster than BRISK-FREAK, although this

combination contains a few more features in the database. This is explained by the fact that the descriptor vector of ORB only has half the size as that of FREAK and thus the distance is calculated faster. As the same number of calculations is performed in each test scenario, there is no correlation between the number of selected matches and the runtime.
In order to understand the results of the runtime for validation, at first it is important to know how the modified RANSAC algorithm works in detail. From the set of selected matches after thresholding (max. 50), all possible combinations of groups of 3 are formed. For each group, the affine transformation is determined and the number of inliers is calculated. In general, this is performed for all possible groups of 3, unless an affine transformation is found that fits for all matches of the set. Then, the algorithm terminates immediately. In return, if there still is at least one wrong match left after thresholding, all possible combinations of groups of 3 are checked. As shown in section 3.3.2, the number of possible combinations and thus the runtime increases exponentially with the size of the set of remaining features. So, on the one hand, regarding runtime, it is highly desirable that only correct matches are left after thresholding. This would then be detected very quickly during the RANSAC algorithm and hardly any computational effort would be required. However, if at least one mismatch remains after thresholding, then it is desirable that the number of mismatches is as small as possible. Otherwise, the runtime would increase exponentially. How well the thresholding only selects correct features is mainly influenced by the description algorithm. A good description algorithm ensures that only the descriptor vectors of the same features have a small distance.
Looking at the results for the validation, three large groups of points with respect to the runtime are noticeable. The first one is at the level of 0 ms. The second group lies between 25 and 400 ms and the third group is located at 900 to 1000 ms. The scenarios of the first group contain only correct features after thresholding. These are detected by one of the first affine transformations tried and the RANSAC algorithm terminates immediately. For the scenarios of the second group, there are two possibilities. The first possibility is that the set also contains only correct matches, but these are recognized by an affine transformation of a group of 3 tried later, e.g. due to inaccuracies of the UTM position computation. This is the case for datapoints with a high number of selected and thus remaining matches. The second possibility is that the set contains one wrong match, but both the number of mismatches as well as the number of correct matches are small, so that it does not take too long to try all possible combinations. This is the case for the datapoints in the range of low numbers of selected matches (less than 25). The third group contains the test scenarios with a high number of remaining matches after thresholding, of which at least one is wrong. Thus, it needs to iterate over all possibilities of groups of 3, which leads to runtimes of up to 1000 ms. If the matching after thresholding performs very well with respect to the number of correct matches, but there is at least one mismatch left, a very high runtime results. This also gets clear in the plot, as the datapoints of group 3 mostly lead to at least 25 selected matches.
Now, by looking at which algorithm is represented in which group, the average runtimes in table 4.9 can be explained. This is because the more datapoints of an algorithm

are located in group 1 and 2, the better is the used descriptor and the shorter is the expected runtime of the validation. On closer inspection, it is notable that the datapoints of BRISK-BRISK and SURF-SURF are mainly found in group 1 and 2. For the other four combinations, the datapoints are evenly distributed across all three groups, with SURF-FREAK also having a large proportion located in group 1. This observation is confirmed by determining the average runtimes, where SURF-SURF and BRISK-BRISK differ only slightly, but are about 5 times faster than SURF-FREAK and about 8 times faster than the remaining combinations. This allows to conclude that the SURF and BRISK descriptor generally are the most expressive, and that no false matches remain after thresholding. The better average runtime of SURF-FREAK, compared to the other combinations which also use FREAK, is explained by the lower number of remaining matches after thresholding. This is evident in the plot as well, as the datapoints mostly have less than 25 selected matches and are located in group 2.
Looking at the distribution of the datapoints in the plot for the total runtime, a strong similarity to the plot for validation can be seen. This implies that the validation forms the main part of the total runtime, and both detection and description as well as matching and thresholding only contribute slightly to the total runtime. This is confirmed by looking at the magnitudes of the average runtimes of the individual parts in table 4.9. Detection and description requires between 10 and 105 ms and matching and thresholding only 6 to 28 ms. On the other hand, validation requires on average about 65 ms for the best-performing combinations. But, as already mentioned, the other combinations take significantly longer and result in average validation times of up to 615 ms. Therefore, same as with the runtime of the validation, there are significant differences in the total runtime, too. SURF-SURF and BRISK-BRISK are the fastest overall, with SURF-SURF being 33.2 % faster than BRISK-BRISK due to the faster detection and description. SURF-FREAK takes about twice as long as BRISK-BRISK. The combinations of the ORB detector are about 30 % slower despite the fast detection and description due to the high validation time. BRISK-FREAK is by far the slowest (about another 25 % slower), as detection and description as well as validation take very long. Furthermore, by looking at the standard deviation of the total runtime in table 4.9, it gets clear that a runtime cannot be exactly predicted for any of the combinations, as the results scatter very strongly. This is explained by the wide range of validation runtimes between approximately 0 to 1000 ms. The mean values can thus only be regarded as an approximate reference.
Additionally, it can be noted that the runtime for the onboard matching takes up to 1 s even by use of a powerful laptop and of a small database that only covers an area of approximately 550 km $\times$ 700 km. If the database would be enlarged, the runtime for detection and description would remain the same, but matching and thresholding would take significantly longer and might account for a larger proportion of the total runtime than it does at the moment. In addition, with a larger database, the risk of remaining mismatches after thresholding would probably also increase for SURF-SURF and BRISK-BRISK, so that using these combinations would also take a similar amount of time for validation and thus overall, as the other algorithms. Furthermore, the attitude and position is calculated based on the selected matches afterwards, what also

costs additional computational effort. If all this is done on a less powerful processor onboard the satellite, this could lead to runtimes of several seconds. However, as the determination of position and attitude is a time-critical task of the satellite, the runtime could be one of the major limitations of the applicability of AVS.

4.1.7 Choice of Combination of Algorithms

In the following section, the performance of the different combinations of algorithms is summarized and based on the results, a certain combination is suggested.

All combinations provide a reliable onboard matching process by only selecting correct matches except of SURF-SURF and SURF-FREAK. The use of these two combinations leads to including mismatches in 1 of 6 test scenarios during test 4.1.4. Furthermore, significantly less features are selected by using SURF-SURF or SURF-FREAK. ORB-FREAK and BRISK-BRISK achieve twice as many and ORB-ORB as well as BRISK-FREAK 3 times as many matches. Even though these results are only addressed in detail in test 4.1.3, they can also be confirmed by the results of test 4.1.4 and 4.1.6 (see table 4.5 and fig. 4.12).
Regarding invariance to rotation around the nadir axis, it is shown that the combinations with the BRISK detector are very sensitive and the number of selected matches varies strongly. In contrast, all other combinations are much more robust against rotation and the number of selected matches only differs slightly.
In test 4.1.5, it gets clear that there are big differences regarding required memory space of the database. SURF-FREAK needs the least storage. ORB-ORB is also a good choice with respect to required memory space, as a high amount of features can be stored while keeping the required storage low due to the small size of the descriptor vector of

Requirement	**Detector-Descriptor**					
	ORB-ORB	**ORB-FREAK**	**BRISK-BRISK**	**BRISK-FREAK**	**SURF-SURF**	**SURF-FREAK**
Nr. of matches	28.5	22.3	21.3	31.3	10.5	11.0
Reliability	100 %	100 %	100 %	100 %	83.3 %	83.3 %
Invariance to rotation	3.1 %	7.1 %	39.8 %	12.7 %	0.1 %	17.6 %
Memory space [MB]	432.0	684.0	784.8	763.2	897.6	266.4
Runtime [ms]	559.96 ± 456.13	584.81 ± 457.30	197.39 ± 189.03	744.79 ± 444.94	131.85 ± 168.23	391.76 ± 404.04

TABLE 4.10: Summary of the performance of the different combinations of algorithms. For all results $n = 700$ is set. Results of ‹Nr. of matches› are taken from test 4.1.3 indicating the average number of selected matches after validation over 6 test scenarios. ‹Reliability› is evaluated based on the results of test 4.1.4, where the combinations of the SURF detector selected mismatches in 1 of 6 test regions. ‹Invariance to rotation› is assessed based on the average percentage deviation of the number of selected matches over four rotations of each of the 6 test images. The entries of ‹Memory space› represent the estimated storage needed for a database covering the entire land area of the Earth calculated in test 4.1.5. ‹Runtime› is evaluated based on the mean values and standard deviation of the runtime of 24 test scenarios.

ORB. All other combinations need significantly more storage.
In terms of runtime, SURF-SURF and BRISK-BRISK achieve by far the best results due to the faster validation. BRISK-FREAK has by far the slowest average runtime. However, as already described, these significant differences could resolve if a larger area is covered by the database. Then, the validation, which forms the main part of the total runtime, would probably need the same amount of time for all algorithms. Depending on how much the runtime of the matching process would increase by storing more features, the total runtime of all algorithms would probably be similar and there might occur only limited advantages of a certain combination.
When the results are considered altogether (see table 4.10), there is one combination achieving the best results overall: ORB-ORB. This combination provides a reliable matching in all test scenarios and achieves one of the highest numbers of selected matches. It is robust to rotation and due to the small size of its binary descriptor vector, it is possible to store a large number of features in the database while keeping the memory requirements low. In the current test application, this combination does not have the shortest runtime, but, as described, with a larger number of stored features, the onboard matching would probably take a similar amount of time for all combinations. Therefore, the combination of ORB-ORB is suggested for the application of AVS based on the results of this thesis. It needs to emphasize that other combinations of algorithms achieve better results than ORB-ORB in certain tests. Therefore, there might exist applications for which another combination would be the better choice. In general, the used combination of algorithms needs to be chosen based on the specific requirements of a certain application. Therefore, table 4.10 provides an overview of the results. Additionally, it is important to mention that there might be a combination of algorithms not investigated in this thesis, but even yielding better results.

4.1.8 Distribution of the Stored GCPs

In the following section, the applicability of AVS is investigated by evaluating the distribution of the GCPs stored in the database.

Requirement
As noticed in the previous experiments, there are regions on Earth where only few to no features are matched onboard. Now, in order to evaluate the applicability of AVS in general, the question of how many of these regions exist on Earth needs to be addressed. One requirement of AVS is to determine the attitude and position during the entire orbit. However, this is only possible if a sufficient number of features is continuously matched in the regions displayed in the onboard images. If matching is only possible on a small part of the Earth, this could mean that AVS cannot be implemented as originally planned. Therefore, it is necessary to investigate how large the proportion of the area covered by the database is in which only few to no features are likely to be matched onboard.
To be able to predict this for an area, the database, which has been created based on satellite imagery that depicts that area on ground, is analyzed. As mentioned in section 4.1.2 (see fig. 4.2b), a correlation between the number of GCPs stored in a region

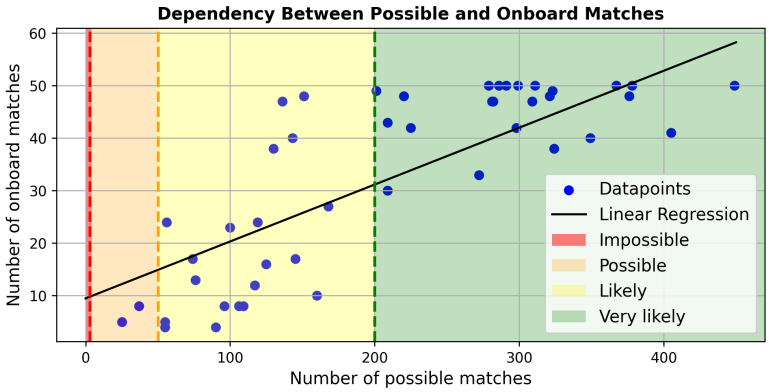


FIGURE 4.13: Plot of the dependency between number of possible matches and number of onboard matches within a certain test region. The colored area represent the four categories of the onboard matching regarding the number of possible matches.

(possible matches) and the number of onboard matches can be assumed. To confirm this after setting additional parameters during the other tests, 48 test scenarios (region 4-9, 12-15 and 18-23 of three scenes) are run for the database from section 4.1.3 using ORB-ORB (with $n = 700$). The plot of the resulting dependency between number of possible matches and number of onboard matches is shown in fig. 4.13. These results confirm that the probability for a high number of onboard matches increases (see linear regression) with a higher number of possible matches in this region. This is explained by the fact that some regions only have few unique appearances in satellite imagery. In these regions, only a few features can then be both included in the database as well as detected onboard. This results in very few to no features being matched onboard. Unfortunately, there currently is no approach to address this problem and therefore to improve onboard matching in these regions.

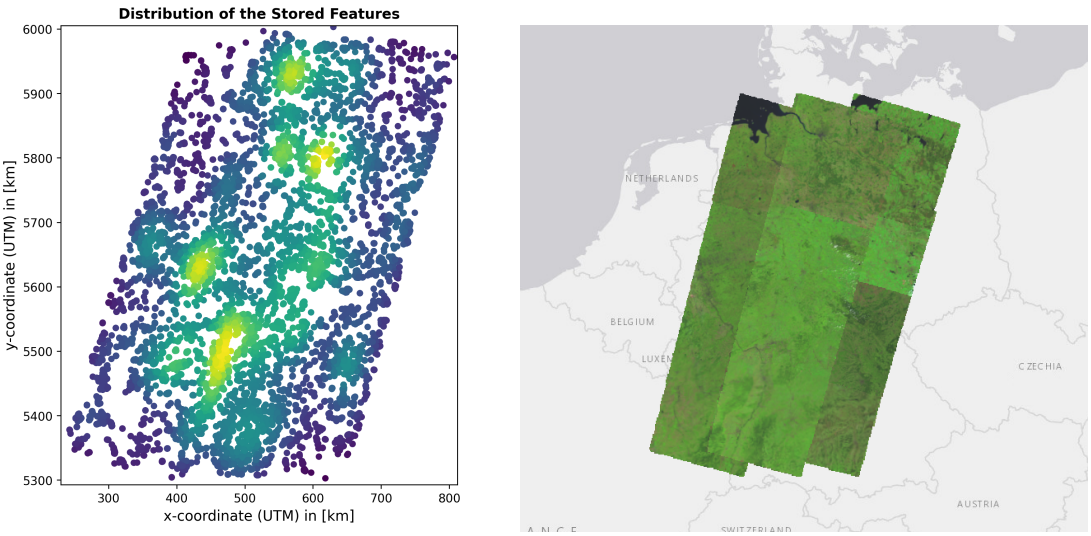
Experimental Setup and Metric

In the following experiment, two databases are investigated with respect to the proportion of the covered area in which probably only few to no features would be matched onboard. The first database is the one used in the previous tests. It is created based on 12 scenes and roughly maps the area of Germany. It contains 27740 features. For the creation of the second database 9 scenes are used, which cover the area around Denmark. It keeps 9344 features. Both databases are created using ORB detector and descriptor.

Now, to assess the proportion of the area where onboard matching is unlikely, the covered area on ground is divided into equally sized windows. The size of a single window represents the FOV of the satellite camera. Thereby, as during the entire test application, the camera integrated in TOM is used as a reference. It has a swath width of 50 km and a swath height of 40 km. For each of these windows the number m of features which are located in this region and are stored in the database (possible matches in this region) is determined. Based on this number, it is evaluated how likely a successful onboard matching is in this region. Therefore, four categories are distinguished. The boundaries are determined based on the results from fig. 4.13 and are also illustrated in the plot:

- $\mathbf{m} < \mathbf{4}$: Onboard matching impossible. As mentioned in previous sections, at least four features have to be matched onboard and remain after thresholding, so that the modified RANSAC algorithm is able to recognize and select them.
- $\mathbf{4} \leq \mathbf{m} < \mathbf{50}$: Onboard matching possible, but unlikely. Enough features are stored in these regions, so that matching would be possible, but due to the small number it is unlikely. If matching would work in these regions anyway, the number of matches would then be sufficient to determine the attitude and position, but probably only with low accuracy.
- $\mathbf{50} \leq \mathbf{m} < \mathbf{200}$: Onboard matching likely. Enough features are stored in these regions, so that onboard matching is likely. The number of matches in these regions should also ensure satisfying accuracy of attitude and position.
- $\mathbf{200} \leq \mathbf{m}$: Onboard matching very likely. Enough features are stored in these regions that onboard matching is very likely. In addition, a number of at least 30 onboard matches is expected, which would guarantee a high accuracy of the attitude and position.

Based on the number of windows of the different categories, it is then evaluated how likely an onboard matching is within the entire area covered by the database. As it has not yet been investigated how the number of matches and the accuracy of attitude and position are related in detail, the differentiation of the first three groups is most important. It is also important to note that this approach is only a rough estimate and the results are only indicative.



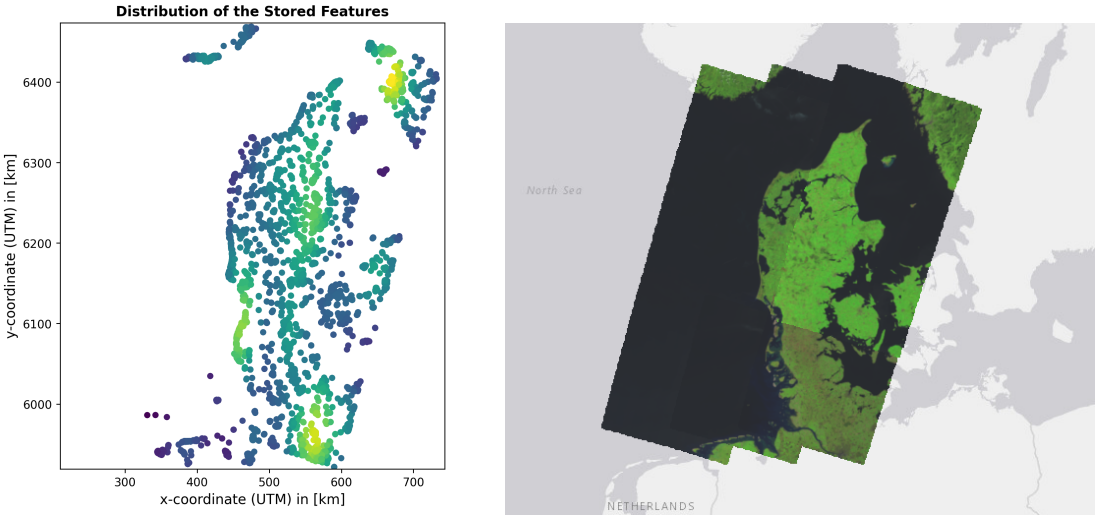
(A) Plot of the GCPs (B) Covered area on ground

FIGURE 4.14: Distribution of the GCPs for the first database (Germany) in (A). Each datapoint illustrates a GCP. The color of the datapoints represent the density of the GCPs: The brighter the color, the more GCPs are located within this region. (B) shows the area on ground covered by the database. Image taken on Earth Explorer [50].

Results and Evaluation

Plots (A) of fig. 4.14 and of fig. 4.15 illustrate the distribution and density of the GCPs stored in the first and second database respectively. Plots (B) of fig. 4.14 and of fig. 4.15 show the covered area on ground for the first and second database respectively. The percentage of the different categories of the total area for the first and second database is shown in fig. 4.16 (A) and (B) respectively.

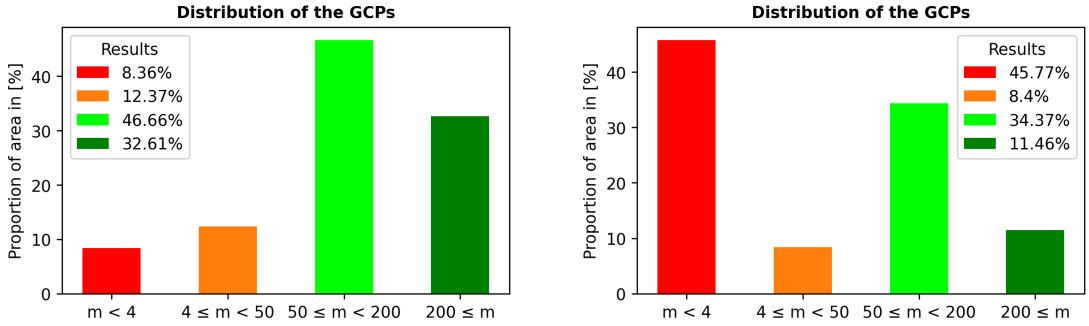
At first, the database covering the area of Germany is evaluated. In general, it is seen in fig. 4.14 that the GCPs are distributed over the entire covered area. Furthermore, it is noticeable that some parts contain significantly more features than others. Only a few features are located in the upper and lower left parts indicating regions with less unique appearances in satellite imagery. Many features are located in regions of a central vertical strip. This indicates regions with many unique appearances. Additionally, it is also noticeable that the scenes based on which the database is created overlap exactly in these regions. One could assume that this is the only reason why more features are collected. However, as duplicates are sorted out during database creation, this still indicates regions with significantly more unique appearances in satellite imagery. Looking at the results in fig. 4.16 (A), they are consistent with the observations just presented. The proportion of the area in which matching is likely to be impossible is estimated to only 8.36 %. In turn, this means that onboard matching should be possible in more than 90 % of the total covered area. Furthermore, the results show that matching is even likely to happen in 79.27 % of the area. Thus, after the satellite takes multiple images during the overflight, the distribution of GCPs for this database should ensure that regions, where sufficient GCPs are stored, are displayed in the majority of images. This should then ensure that the satellite can determine its attitude and position when it passes over the covered area.



(A) Plot of the GCPs

(B) Covered area on ground

FIGURE 4.15: Distribution of the GCPs for the second database (Denmark) in (A). Each datapoint illustrates a GCP. The color of the datapoints represent the density of the GCPs: The brighter the color, the more GCPs are located within this region. (B) shows the area on ground covered by the database. Image taken on Earth Explorer [50].



(A) First database: Germany

(B) Second database: Denmark

FIGURE 4.16: Results of the distribution of the GCPs: Percentage of the different categories of the total covered area.

Next, the second database covering the area around Denmark is investigated. Looking at the distribution of the GCPs in fig. 4.15, it is immediately noticeable that the stored features are located only in land areas. The areas showing water bodies do not contain any features at all. This observation is confirmed when looking at the results in fig. 4.16 (B). No matching is likely to be possible in about 45 % of the entire covered area which roughly matches the proportion of water bodies displayed in fig. 4.15 (B) of about 40 %. This observation is both easy to explain as well as problematic regarding the applicability of AVS in general. Water areas appear monochromatic and without high contrasts due to no corners or edges in satellite images. In addition, the scenery in water bodies changes every second, so even if a feature is detected in an image of such a region, it cannot be matched with another image. The related problem is that about 70 % of the Earth is covered by water [61]. This leads to the conclusion that, only for this reason, no features can be stored in the database or matched onboard on 70% of the Earth. However, as the objective of AVS is to determine the satellite's attitude and position during the entire orbit, this circumstance constitutes the most serious limitation regarding the applicability of AVS. Looking at the proportions of the categories of the second database and excluding the proportion of water areas, similar values result compared to those of the first database (cat. 1: 9 %, cat. 2: 14 %, cat. 3: 57 %, cat. 4: 19 %). Again, in most of the total land area (approx. 76 %), it is likely that the onboard feature matching can be performed successfully.

In summary, there are very few regions in the investigated land areas that contain too few unique appearances in satellite imagery. Therefore, it is likely that the onboard matching and thus the attitude and position determination can be successfully performed during an overflight over these land areas. However, this test also makes clear that over water areas no features can be added to the database or matched onboard. This forms a huge constraint for the application of AVS as a majority of the Earth's surface is covered by water bodies.

4.2 Summary of Test Results

In the following, the results of all the tests performed in section 4.1 are summarized. At first, it is shown that an additional validation method after standard matching and thresholding is necessary to ensure a reliable onboard matching. The proposed method based on the RANSAC algorithm fulfills this requirement and is able to only select correct matches in almost all test scenarios. On average, 95.9 % of all remaining correct matches are selected. In test 4.1.3, it is demonstrated that the number of detected features during database creation and onboard matching and thus the number of stored features can be reduced, indicated by the parameter n. Therefore, the required memory space as well as the computational effort for the onboard matching can be decreased, too, while not loosing any added value. The best value for the parameter n lies between 400 and 700. Furthermore in this test, as well as in tests 4.1.4 to 4.1.6, it is shown that there are big differences depending on which feature detection and description algorithm is used regarding the performance of

- number of resulting matches after onboard feature matching,
- invariance to rotation around nadir axis of the onboard image,
- required memory space of the database and
- runtime of the onboard feature matching.

After comparing the different selected combinations of algorithms, one combination is suggested as a suitable choice for the application of AVS as convincing in almost every test: ORB-ORB. Furthermore, multiple constraints are identified during these tests regarding memory space, runtime and distribution of the GCPs. These and other constraints of AVS are addressed in the following section in detail.

4.3 Constraints and Future Work

The following section addresses constraints regarding the applicability of AVS that have either arisen during the work on this thesis or that are currently concerned. Therefore, it is presented what still needs to be investigated in future work. Lastly, it is described how the applicability of AVS can be tested further with data recorded by TOM.

4.3.1 Cloud Coverage

When creating the database, explicit care is taken to ensure that the chosen Landsat 8 images had 0 % cloud coverage, so that all unique appearances of the Earth's surface within the displayed region could be detected and included in the database. Furthermore, this is also taken into account when choosing the test images. In reality, however, it could happen that a part of the image or even the entire image is covered by clouds. If this is the case, then fewer or even no GCPs can be detected and thus fewer or no matches remain after the onboard feature matching. As a consequence, the position and attitude can only be determined less precisely or even not at all. This is one major constraint of AVS and has to be considered in future work.

4.3.2 Seasonal Changes in Vegetation

When selecting the Landsat 8 image data to create the database, attention is not only paid to cloud coverage but also to the date of record. All selected images have been captured in the months of March to July. This is done because at the time of this writing it is unknown how seasonal changes in vegetation affect feature detection in satellite imagery. It is possible that the changing visual appearance of vegetation in images could result in different features detected in images showing the same scene but are recorded at different times of the year. This would result in the need to create a separate database for each season, which would significantly increase the memory requirements on the satellite. In this context, it is important to emphasize that the entire analysis in this work, with respect to the algorithms and the parameters, is tuned to the months of March to July. Therefore, all results are to be interpreted under this restriction. How seasonal changes in vegetation affect the performance of AVS in detail and how the parameters might need to be adjusted, needs to be further investigated in future work.

4.3.3 Eclipse

Another major limitation of AVS is the dependency on sunlight. Features can only be detected if they are illuminated and thus can be captured by the onboard camera. Unfortunately, the eclipse phase for a LEO satellite lasts about 35 % of the orbit period [41]. During this phase features can only be detected in areas that are strongly illuminated at night, such as large cities. However, these detected features appear completely different in the image than those during daytime, so that no matching with the stored GCPs would result successfully. In addition, these features are much less unique and not constant over time, so that it doesn't make sense to create an additional database for the eclipse phase. Therefore, during about 35 % of the orbit period, no GCPs can be detected and no attitude and position can be determined based on AVS.

4.3.4 Distribution of GCPs

In section 4.1.8 it is shown that a worldwide even distribution of stored GCPs is not possible, because no features can be matched in images of water bodies, which, however, make up about 70 % of Earth's surface. Therefore, AVS is not applicable to at least 70 % of the Earth. Furthermore, it is shown that in the two land areas studied (Germany and Denmark), the distribution of GCPs is likely to be sufficient to apply AVS when overflying these areas. However, these results cannot be used to conclude that onboard matching is possible over every land area on Earth. For example, there are large land areas that appear monochromatic and without contrasts in satellite imagery just like water areas, such as Antarctica or large deserts. In these areas, it could then also happen that no features are detected and thus no GCPs are created. Additionally, there are large land areas that may appear very similar in satellite images, such as areas in rainforests. Here, it could happen that features are detected, but their descriptor vectors are so similar that many or only wrong matches remain after thresholding. This could then lead to incorrect matches being selected even after applying the validation method, which in turn would result in incorrect attitude and position. Therefore, as

soon as the orbit of a satellite, at which AVS is to be applied, is determined, it needs to be analyzed in detail how well GCPs can be generated for the overflown areas. If the overflown areas are poorly suited for AVS, the orbit could be adjusted accordingly.

4.3.5 Size of the Database

In this work, the entire process of the onboard feature matching is tested using databases that cover a maximum of 12 scenes, which in this case corresponds to an area on ground of approximately 500 km $\times$ 700 km. As shown, onboard matching with such a database works quite well for the tested area on Earth. However, it is not clear whether matching would work as well with a database covering a larger area and thus including more features. This is because when using a database that stores a significantly higher amount of features, the risk of incorrectly matched features also increases, as the probability increases that by chance the database keeps another descriptor vector with a shorter distance. These mismatches then might be sorted out by the validation method but reduces the number of resulting matches. How strongly this effect occurs and how big its impact on the performance of AVS is, also needs to be investigated in future work.
Even though the databases used in the tests of this thesis only have a limited size, nevertheless problems have got clear that would be amplified with a larger database. In test 4.1.5, it gets obvious that a database covering the entire land area of the Earth requires a high amount of storage with approximately 300 to 500 MB (using the current setup). This high demand regarding memory space on the satellite might get even higher if multiple databases need to be created and integrated due to seasonal changes.
Unfortunately, as seen in test 4.1.6, the onboard feature matching is also associated with a high computational effort and thus has a long runtime even using the limited database. Increasing the covered area by the database and therefore the number of stored features, would increase the computational effort for the matching process significantly. Considering as well the limited computing power available onboard the satellite, runtimes of multiple seconds are to be expected. This restricts the real-time capability of AVS enormously. To reduce the runtime for the matching process, the ANN approach, introduced in section 2.8, could be used as searching technique instead of using the Brute-Force approach. However, this might lead to a higher rate of mismatches after thresholding, which then would reduce the number of remaining matches after the onboard matching process. If using ANN still would be a promising alternative to the Brute-Force searching technique, needs to be investigated in future work.

4.3.6 Accuracy of Attitude and Position Determination

In the current setup for creating the database, the ground positions of the GCP features are determined based on the UTM positions of the four corners of the respective Landsat 8 image. The UTM coordinate system projects the spherical surface of the Earth to a two-dimensional plane. Therefore, the UTM system suffers from distortions that get bigger the further one moves away from the equator. Thus, the ground positions of the GCPs could contain errors. Calculating the attitude and position of the satellite based on these positions might cause problems regarding a satisfying accuracy. The

metadata-file provides the ground positions not only in the UTM system but also in the ECEF frame (latitude and longitude). Thus, the ground positions of the GCP features could be determined based on these values. Unfortunately, this non-trivial computation could also result in error-prone positions due to the projection of the three-dimensional scene to a two-dimensional image plane even though the used coordinates of the corners are more precise. Therefore, it needs to be examined what the best method is to determine the ground positions of the detected GCPs and how strongly the resulting accuracy of these positions impact the accuracy of the attitude and position determination of the satellite in detail.
Furthermore, it needs to be investigated how the number of matched features onboard is related to the accuracy of the attitude and position determination in detail. In the current setup, maximum 50 matches are selected after thresholding and given as input to the proposed validation method. Thus, in any case maximum 50 matches could result. If a less number of matched features would lead to a sufficient accuracy of attitude and position determination, less matches could be selected after thresholding. This would save a high amount of computational effort during validation. As the validation method forms a large proportion of the total runtime, this could greatly improve the long total runtime of the onboard matching.

4.3.7 Angle of View

Another constraint that could complicate the application of AVS is the impact of the onboard camera's viewing angle on the onboard matching performance. As GCPs are three-dimensional structures on Earth's surface, they may appear differently or are even covered by other objects on ground on multiple images, if these images are taken from different camera angles. Thus, other structures on ground might be more unique in these images according to the detection algorithm. This could result in stored GCP features not being detected and matched onboard. How strongly and at which camera angles this effect occurs needs to be studied.

4.3.8 Onboard Camera

For the proposed test application, the image data used for creating the database and for the onboard matching is taken by the same satellite and thus by the same camera. In future missions AVS is supposed to be integrated on different satellites. These satellites might not carry the same camera as the one of the Landsat 8 satellite. Thus, two aspects need to be investigated in future work.

Spectral Bands	**Landsat 8**		**Sentinel-2A**	
	Wavelength	**Resolution**	**Wavelength**	**Resolution**
Blue	0.45 µm–0.51 µm	$30\,\mathrm{m\,px^{-1}}$	0.46 µm–0.53 µm	$10\,\mathrm{m\,px^{-1}}$
Green	0.53 µm–0.59 µm	$30\,\mathrm{m\,px^{-1}}$	0.54 µm–0.58 µm	$10\,\mathrm{m\,px^{-1}}$
Red	0.64 µm–0.67 µm	$30\,\mathrm{m\,px^{-1}}$	0.65 µm–0.68 µm	$10\,\mathrm{m\,px^{-1}}$

TABLE 4.11: Comparison of spectral bands of Landsat 8 [52] and Sentinel-2A [13].

First, the onboard camera might have a different ground pixel resolution. For example, the camera used in TOM provides a ground pixel resolution of approximately $20\,\mathrm{m\,px^{-1}}$, which is lower than the one provided by the Landsat 8 onboard camera (see table 4.11). A different ground pixel resolution could lead to detected features in the onboard image that are different to those detected in the Landsat 8 images. This would result in fewer or no matched features onboard. Although all presented feature detection and description algorithms are supposed to be invariant to scale, it still has to be investigated whether a different ground pixel resolution influences the performance of the onboard matching. If this is the case and significantly fewer features are matched, there are currently two ideas to solve this issue. If the value of the ground pixel resolution of the Landsat 8 image data is higher, the onboard images could be downscaled before onboard matching, so that the edited images have the same resolution as the Landsat 8 image data. If the value of the ground pixel resolution of the Landsat 8 image data is lower, the Landsat 8 image data could be downscaled before creating the database. How well these approaches might work, needs to be examined.
Second, the RGB-images captured by the camera sensors on the satellites that are to perform AVS could cover a different spectral range than the images used to create the database. Although the spectral ranges of the single color channels might overlap, it is very likely that they are slightly shifted and do not cover the identical range of wavelength (see table 4.11). This could result in the RGB-images depicting different structures and thus, just as with different resolution, different features being detected onboard the satellite as stored in the database. Whether and how strong this effect occurs depends on the camera sensors used and has therefore to be investigated individually for a certain mission, just like regarding the effect with different resolutions. In summary, it is important to take into account which camera is integrated, both the one onboard the satellite that is to perform AVS and the one onboard the satellite whose captured images are used to create the database.

4.3.9 Upcoming Tests

In the following, an approach is presented, how the functionality and applicability of AVS, as well as how the impact of the described constraints can be tested in future work. For this purpose, data is used that is recorded by TOM and that is sent to the ground station. First, images from the onboard camera are downloaded. For each of these images, the attitude and position data at the time of record is also transmitted. Furthermore, databases of GCPs are created, covering the areas on ground that are depicted in the TOM images. Subsequently, onboard matching is simulated for the satellite images, as it is performed in the proposed test application. Based on the matched features, the attitude and position of the TOM satellite at the time of record of the image is determined. The exact method required for attitude and position determination based on matched features needs to be developed in advance. The results then are compared with the true, recorded attitude and position data of TOM. In this way, a wide variety of aspects can be tested and investigated, such as the impact of seasonal changes in vegetation, of the angle of view and of the used onboard camera on the performance of

AVS as well as the accuracy of attitude and position determination based on matched features.

4.4 Outlook

The following section provides an outlook on how AVS might be adapted in order to handle the described constraints and how AVS might be extended in the future.

4.4.1 Adjustment of the Application of AVS

Originally, the objective of AVS is to provide reliable attitude and position determination during the entire orbit in real-time. However, after the tests performed in this thesis and the constraints described, it gets clear that the attitude and position determination both during the entire orbit and in real-time are not feasible. Therefore, the following idea is presented of how the application of AVS nevertheless might qualify to be useful in future missions.
To determine the position ($\vec{x} = (x, y, z)$) and attitude (θ, ϕ, ψ) of a satellite, often, a Kalman filter is used. The Kalman filter is a powerful mathematical method for iteratively estimating and predicting the state $\vec{s}$ of a dynamic system (in this case $\vec{s} = (\vec{x}, \theta, \phi, \psi)$) based on error-prone sensor measurements. The basic procedure consists of the following steps. For more details, refer to [22].

1. Prediction: Predict system state $\vec{s}_k$ at time k based on the state $\vec{s}_{k-1}$ at time $(k-1)$, the control input $\vec{u}_{k-1}$, the state-transition model $\boldsymbol{F}_{k-1}$ and the control-input model $\boldsymbol{B}_{k-1}$:
$$\vec{s}_k^P = \boldsymbol{F}_{k-1} \cdot \vec{s}_{k-1} + \boldsymbol{B}_{k-1} \cdot \vec{u}_{k-1} \,. \tag{4.1}$$

2. Estimation: Estimate system state $\vec{s}_k$ at time k based on the sensor measurements $\vec{y}_k$ at time k and the observation model $\boldsymbol{H}_k$:
$$\vec{s}_k^E = \boldsymbol{H}_k \cdot \vec{y}_k \,. \tag{4.2}$$

3. Correction: Determine system state $\vec{s}_k$ at time k based on the difference between predicted and estimated state and Kalman gain γ:
$$\vec{s}_k = \vec{s}_k^P + \gamma \cdot (\vec{s}_k^P - \vec{s}_k^E) \,. \tag{4.3}$$
The Kalman gain γ gets adjusted in each step based on the process and measurement noise.

For determining the attitude and position of a satellite, multiple sensors are used to estimate the system state $\vec{s}_k^E$, such as sun sensors, GPS modules, gyroscopes or star trackers. However, some of these sensors are inaccurate or affected by drift, so that a state estimation based on their measurements over longer periods of time would be insufficient, because it would get increasingly inaccurate. Other sensors provide higher accuracy and would, combined with the less accurate sensors, improve the state estimate, but

are elaborate to operate in orbit, such as star trackers. Therefore, it is desirable to have a sensor that can be operated with little effort and that provides attitude and position measurements with high accuracy so that the inaccurate state estimation based on the other sensors can be corrected if the drift gets too big. The update rate of these measurements does not need to be as high as for the other sensors, but it is sufficient to correct the drifting state estimate in a larger time interval. Between two corrections, the accuracy of the other sensors is sufficient.
Such a sensor could be realized by AVS. Position estimates from AVS could be involved in the Kalman filter as soon as a sufficient number of features are detected and matched. Then, the current position and attitude estimation of the drift affected sensors could be corrected. If the correction by AVS would not be possible for a short time due to the described constraints (cloud coverage, eclipse phase, insufficient distribution of GCPs or long runtime), the state estimates of the other sensors would be sufficient during these short periods. As soon as an onboard feature matching would successfully take place again, the state would get corrected. This approach would have several advantages. On the one hand, the operation of AVS would require little effort, as only the database would need to be integrated in the satellite. The images taken by the payload camera would be used as onboard image data. On the other hand, the performance of AVS could be improved. The large database could be divided into several small databases, each covering a certain area of the Earth. As the approximate position would be known by the other sensors, the onboard detected features would only be matched with the database whose GCPs would be possible to detect at that time. This would reduce the risk for mismatches as well as the required runtime. In summary, AVS could serve as an additional correction method, which could be included without much additional effort.

4.4.2 Transfer of AVS to Other Celestial Bodies

In recent years, more and more efforts have been made to further explore other celestial bodies, such as the Moon or Mars, in order to build manned stations there. Therefore, it is conceivable to bring satellites into an orbit around another celestial body than the Earth, too. However, the sensors used to determine the attitude and position of a satellite in orbit around the Earth is not easily transferable to orbits around other celestial bodies. For example, GPS positioning is not possible in such orbits. For such missions, AVS could be a very promising approach. To implement this, however, sufficient image data of the surface with exact position data would have to be acquired first, so that a GCP database can be created. In addition, it would also need to be investigated whether and how well the surface is suitable so that unique features can be detected and correctly matched in satellite images.

Chapter 5

Conclusion

This thesis aims to analyze state-of-the-art algorithms for feature detection, description and matching as well as to investigate the applicability of these processes regarding AVS. Therefore, it is presented how suitable GCPs can be extracted from satellite image data of the Landsat 8 dataset in order to create the onboard database. Furthermore, it is proposed, how the procedure of onboard feature matching can be implemented. A test application is developed in order to test the two processes of database creation and onboard feature matching as well as to analyze the performance of different feature detection and description algorithms.
First, it is demonstrated that an additional validation method after the onboard feature matching is necessary to ensure a reliable matching, so that no mismatches are selected. The proposed approach based on the RANSAC algorithm fulfills this requirement. Additionally, it is shown that the number of detected features n during database creation and onboard matching, and thus the amount of stored GCPs, can be reduced, while not loosing any added value. This reduces the required memory space of the database as well as the runtime of the onboard matching.
Furthermore, multiple test scenarios are run through and the performance of the different combinations of algorithms is evaluated and compared under multiple aspects: Number of matches after onboard matching, reliability, invariance to rotation, required memory space of the database as well as runtime. It is presented that there are big differences in the performance of the algorithms and that several combinations offer certain advantages regarding AVS. Based on the results of this thesis, the most suitable combination of algorithms for a specific application of AVS can be selected. When the results are considered in total, the ORB detector combined with the ORB descriptor is suggested as the best combination in general.
Moreover, it is shown that the objective of AVS - a reliable matching of a sufficient number of GCPs during the entire orbit in real-time - is not feasible as originally planned due to many constraints arisen during the work on this thesis. How some of these constraints impact the applicability of AVS in detail, needs to be studied in future work. However, a promising adjustment of the application of AVS is proposed. The attitude and position derived from the satellite images could be embedded in a Kalman filter eliminating the need of bulky and highly power consuming sensors that are elaborate to operate in orbit. In this way, the constraints could be handled. As it is demonstrated, that the process of detecting features in satellite images and matching these with GCPs

stored in a database generally works, this adjustment could be feasible and could lead to great improvements in attitude and position determination onboard small LEO satellites.

Appendix A

Landsat 8 Data

A.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 [55]:

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.1.1 Data Used for Figures

Figure ID	Landsat 8 image data ID
3.2, 3.3, 4.1 (left)	*LC08_L1TP_195025_20200624_20200823_02_T1*
4.1 (right)	*LC08_L1TP_195025_20200507_20200820_02_T1*
4.10 (left)	*LC08_L1TP_194023_20220623_20220705_02_T1*
4.10 (right)	*LC08_L1TP_196025_20210602_20210608_02_T1*

TABLE A.1: Landsat 8 data references for figures.

A.1.2 Data Used for Tests

Test 4.1.1: Adjustment of Parameters

Description	Landsat 8 image data ID
Test image	*LC08_L1TP_195025_20200624_20200823_02_T1*
Database image 1	*LC08_L1TP_195025_20200507_20200820_02_T1*
Database image 2	*LC08_L1TP_195025_20190419_20200828_02_T1*

TABLE A.2: Landsat 8 data references for test 4.1.1.

Tests 4.1.2: Onboard Feature Validation - 4.1.6: Runtime

Table A.3 lists all the data used during the second experiment of 4.1.2 and during the tests of 4.1.3 - 4.1.6. For the first experiment of 4.1.2, the same test images are used and the three single scene databases are created based on the data of line *Database Scene 2/1* and *2*, *Database Scene 3/1* and *2* as well as *Database Scene 12/1* and *2*.

Description	Landsat 8 image data ID
Test image 1	*LC08_L1TP_195025_20200624_20200823_02_T1*
Test image 2	*LC08_L1TP_195024_20200507_20200820_02_T1*
Test image 3	*LC08_L1TP_194023_20230509_20230517_02_T1*
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.3: Landsat 8 data references for test *Onboard Feature Validation*.

Test 4.1.8: Distribution of the Stored GCPs

Table A.4 lists all the data used for creating the second database in section 4.1.8 covering the area of Denmark.

Description	**Landsat 8 image data ID**
Database Scene 1/1	*LC08_L1TP_196020_20200615_20200823_02_T1*
Database Scene 1/2	*LC08_L1TP_196020_20180509_20200901_02_T1*
Database Scene 2/1	*LC08_L1TP_196021_20200615_20200823_02_T1*
Database Scene 2/2	*LC08_L1TP_196021_20190629_20200827_02_T1*
Database Scene 3/1	*LC08_L1TP_196022_20230421_20230425_02_T1*
Database Scene 3/2	*LC08_L1TP_196022_20220418_20220427_02_T1*
Database Scene 4/1	*LC08_L1TP_197020_20190417_20200829_02_T1*
Database Scene 4/2	*LC08_L1TP_197020_20180516_20200901_02_T1*
Database Scene 5/1	*LC08_L1TP_197021_20190417_20200828_02_T1*
Database Scene 5/2	*LC08_L1TP_197021_20150422_20200909_02_T1*
Database Scene 6/1	*LC08_L1TP_197022_20220308_20220315_02_T1*
Database Scene 6/2	*LC08_L1TP_197022_20190417_20200829_02_T1*
Database Scene 7/1	*LC08_L1TP_198020_20230708_20230718_02_T1*
Database Scene 7/2	*LC08_L1TP_198020_20150702_20200909_02_T1*
Database Scene 8/1	*LC08_L1TP_198021_20230419_20230429_02_T1*
Database Scene 8/2	*LC08_L1TP_198021_20180507_20200901_02_T1*
Database Scene 9/1	*LC08_L1TP_198022_20230419_20230429_02_T1*
Database Scene 9/2	*LC08_L1TP_198022_20180507_20200901_02_T1*

TABLE A.4: Landsat 8 data references for test 4.1.8.

A.2 World Reference System-2

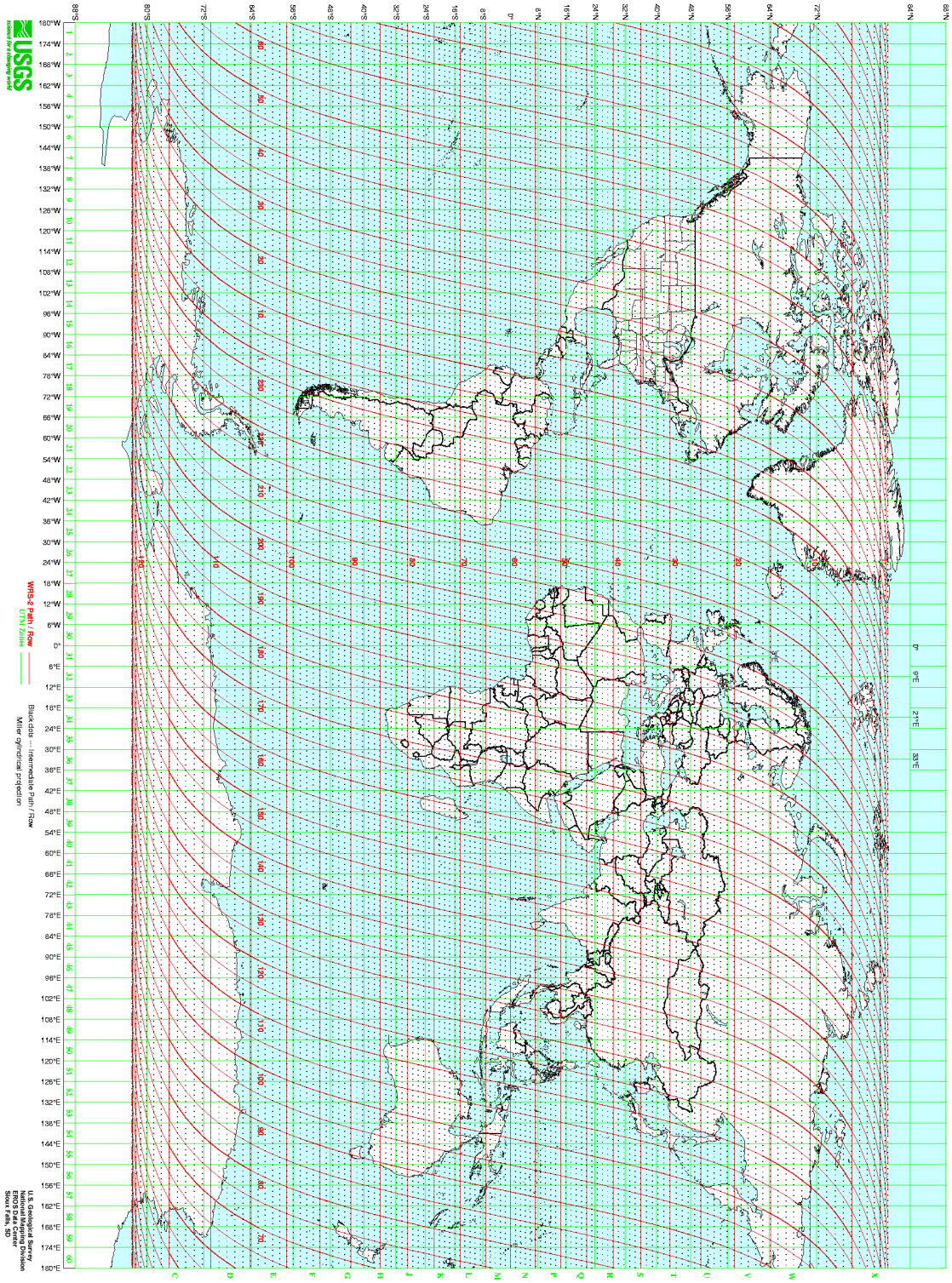


FIGURE A.1: World Reference System-2 [59]

Appendix B

Parameters of the Algorithms

An overview as well as the used values of the different adjustable parameters for the feature detection and description algorithms are listed in the following. As described in section 4.1.1 the parameters can be divided in three groups. To which group a parameter belongs is indicated with the respective superscript number. Additionally, the values of the threshold α for the different description algorithms are listed.

SURF

Parameter	Description	Default value	Used value
Hessian threshold[1]	Only features, whose hessian is larger than hessian threshold are retained by the detector.	100	300
Nr. octaves[2]	Number of gaussian pyramid octaves the detector uses	4	4
Nr. octave layers[2]	Number of images within each octave of a gaussian pyramid	3	3
Extended	Descriptor size: 128D (true) or 64D (false)	false	false
Upright[3]	Enables (false) or disables (true) orientation calculation of features	false	false

TABLE B.1: Parameters for SURF [38].

BRISK

Parameter	Description	Default value	Used value
Threshold[1]	Detection treshold score	30	35
Nr. octaves[2]	Number of gaussian pyramid octaves the detector uses	3	4
Pattern scale[3]	Scale applied to the pattern used for sampling the neighborhood of a keypoint	1.0	1.0

TABLE B.2: Parameters for BRISK [35].

ORB

Parameter	Description	Default value	Used value
Fast threshold[1]	Fast threshold value used for keypoint detection	20	35
Nr. features[1]	Maximum number of retained features	500	3500
Scale factor[2]	Pyramid decimation ratio	1.2	1.2
Nr. levels[2]	Number of pyramid levels	8	4
First level[2]	Level of pyramid the source image is put to; Previous layers are filled with upscaled versions of the source image	0	0
Edge Threshold[2]	Size of the border where the features are not detected	31	35
WTA_K[3]	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[3]	Size of the patch used by the oriented BRIEF descriptor	31	35

TABLE B.3: Parameters for ORB [37].

FREAK

Parameter	Description	Default value	Used value
Nr. octaves[3]	Number of octaves covered by the detected keypoints	4	4
Pattern scale[3]	Scale factor of description pattern	22.0	22.0
Scale normalization[3]	Enables (true) or disables (false) scale normalization	true	true
Orientation normalization[3]	Enables (true) or disables (false) orientation normalization	true	true

TABLE B.4: Parameters for FREAK [36].

Threshold α

The following table lists the used values of the parameter alpha needed for thresholding during the onboard feature matching.

Algorithm	Value for α
SURF	0.12
BRISK	65
ORB	30
FREAK	60

TABLE B.5: Values for thresholding parameter α depending on the used descriptor algorithm.

Bibliography

[1] AIRBUS. *Airbus Ground Control Points*. (online; accessed 11-July-2023).

[2] 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.

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

[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] Bayraktar, Ertuğrul and Boyraz, Pinar. "Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 25.3 (2017), pp. 2444–2454.

[6] Brown, Matthew and Lowe, David G. "Invariant features from interest point groups." In: *Bmvc*. Vol. 4. 2002, pp. 398–410.

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] Drusch, Matthias et al. "Sentinel-2: ESA's optical high-resolution mission for GMES operational services". In: *Remote sensing of Environment* 120 (2012), pp. 25–36.

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

[15] 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.

[16] Google, Earth Engine Data Catalog. *USGS Landsat 8 Collection-2 Tier-1 TOA Reflectance*. (online; accessed 13-July-2023).

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

[18] Hashimoto, Koichi. *Visual servoing*. Vol. 7. World scientific, 1993.

[19] 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.

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

[21] 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.

[22] Kalman, Rudolph Emil et al. "A new approach to linear filtering and prediction problems [J]". In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

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

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

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

[26] 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.

[27] Lowe, David G. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60 (2004), pp. 91–110.

[28] Medium. *Introduction to Surf - Speeded Up Robust Features*. (online; accessed 07-June-2023). 2019.

[29] Mikolajczyk, Krystian. "Detection of local features invariant to affine transformations: application to matching and recognition". PhD thesis. Grenoble INPG, 2002.

[30] Moravec, Hans Peter. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Stanford University, 1980.

[31] Mukherjee, Dibyendu, Jonathan Wu, QM, and Wang, Guanghui. "A comparative experimental study of image feature detectors and descriptors". In: *Machine Vision and Applications* 26 (2015), pp. 443–466.

[32] NASA. *A Royal View of Denmark*. (online; accessed 26-May-2023). 2017.

[33] NASA. *Landsat Science*. (online; accessed 12-July-2023).

[34] Noble, Frazer K. "Comparison of OpenCV's feature detectors and feature matchers". In: *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE. 2016, pp. 1–6.

[35] OpenCV. *BRISK Class Reference*. (online; accessed 1-August-2023).

[36] OpenCV. *FREAK Class Reference*. (online; accessed 1-August-2023).
[37] OpenCV. *ORB Class Reference*. (online; accessed 1-August-2023).
[38] OpenCV. *SURF Class Reference*. (online; accessed 1-August-2023).
[39] Pritchard-Kelly, Ruth and Costa, John. "Low earth orbit satellite systems: Comparisons with geostationary and other satellite systems, and their significant advantages". In: *Journal of Telecommunications and the Digital Economy* 10.1 (2022), pp. 1–22.
[40] Quinlan, J. Ross. "Induction of decision trees". In: *Machine learning* 1 (1986), pp. 81–106.
[41] RM, Sumanth. "Computation of eclipse time for low-earth orbiting small satellites". In: *International Journal of Aviation, Aeronautics, and Aerospace* 6.5 (2019), p. 15.
[42] Rosin, Paul L. "Measuring corner properties". In: *Computer Vision and Image Understanding* 73.2 (1999), pp. 291–307.
[43] Rosten, Edward and Drummond, Tom. "Fusing points and lines for high performance tracking". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. Ieee. 2005, pp. 1508–1515.
[44] Rosten, Edward and Drummond, Tom. "Machine learning for high-speed corner detection". In: *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer. 2006, pp. 430–443.
[45] 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.
[46] Rublee, Ethan et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.
[47] Schenk, Toni. "Introduction to photogrammetry". In: *The Ohio State University, Columbus* 106.1 (2005).
[48] SEO-Analyse. *Google Earth*. (online; accessed 13-August-2023).
[49] 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.
[50] USGS, United States Geological Survey. *Earth Explorer*. (online; accessed 13-July-2023).
[51] USGS, United States Geological Survey. *Ground Control Points*. (online; accessed 11-July-2023).
[52] USGS, United States Geological Survey. *Landsat 8*. (online; accessed 12-July-2023).
[53] USGS, United States Geological Survey. *Landsat Collection-1*. (online; accessed 13-July-2023).
[54] USGS, United States Geological Survey. *Landsat Collection-2*. (online; accessed 13-July-2023).
[55] USGS, United States Geological Survey. *Landsat Collection 2 Data Dictionary*. (online; accessed 13-July-2023).

[56] USGS, United States Geological Survey. *Landsat Level-1*. (online; accessed 13-July-2023).

[57] USGS, United States Geological Survey. *Landsat Level-2*. (online; accessed 13-July-2023).

[58] USGS, United States Geological Survey. *Landsat Missions*. (online; accessed 12-July-2023).

[59] USGS, United States Geological Survey. *World Reference System-2*. (online; accessed 13-July-2023).

[60] Vuuren, Gerhard Hermann Janse van. "The design and simulation analysis of an attitude determination and control system for a small earth observation satellite". PhD thesis. Stellenbosch: Stellenbosch University, 2015.

[61] Wikipedia. *Erdoberfläche*. (online; accessed 18-August-2023).

[62] Wikipedia. *Universal Transverse Mercator Coordinate System*. (online; accessed 16-July-2023).