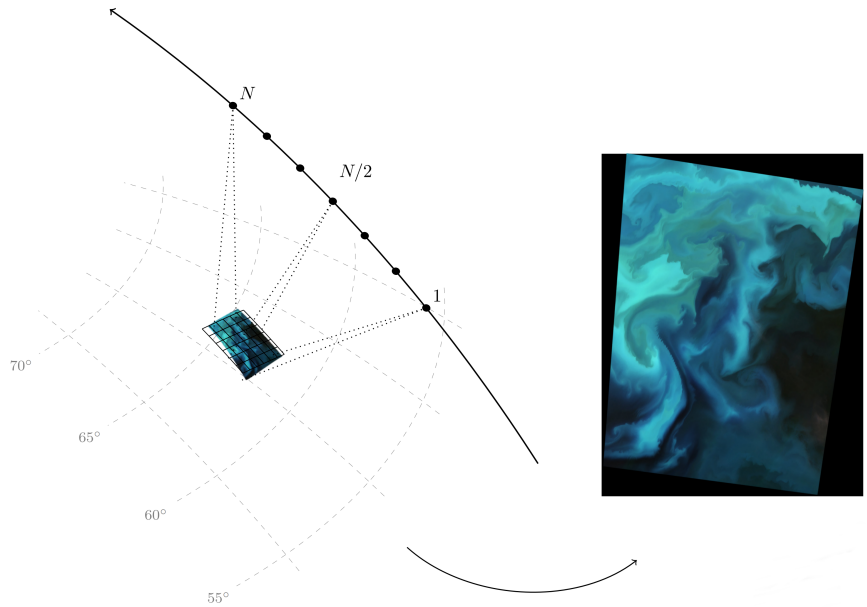
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Dennis Langer

Image registration and georeferencing with snapshot camera for the HYPSO mission

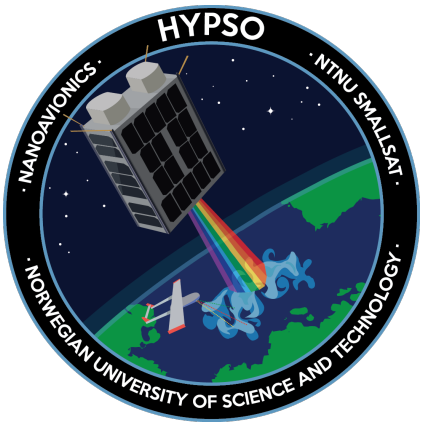
Master's thesis in Industrial Cybernetics
Supervisor: Tor Arne Johansen
June 2019



NTNU
Norwegian University of Science and Technology

Dennis Langer

Image registration and georeferencing with snapshot camera for the HYPSO mission



Master's thesis in Industrial Cybernetics
Supervisor: Tor Arne Johansen
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis concerns three different, but related topics. Image registration and georeferencing of push broom satellite images, and a snapshot camera satellite payload. Image registration and georeferencing techniques are discussed. A geometric-direct registration and georeferencing method is chosen and an algorithm to implement it has been theorized and implemented in MATLAB. It was tested in simulations where it was shown to work well. Arguments for use of a snapshot camera are collected and discussed. A snapshot camera service has been implemented as module to an existing software framework on the target hardware and successfully taken into use.

Cover page figure contains a processed satellite image from ESA's Sentinel-2A satellite and shows a plankton bloom in the Barents Sea.
Credit: ESA/Sentinel-2A - CC BY-SA IGO 3.0

Sammendrag

Denne masteroppgaven omhandler tre forskjellige, men beslektede temaer. Bilde registrering og georeferering av push broom satellitt bilder, og et snapshot kamera satellitt nyttelast. Bilde registrerings og georeferings metoder er diskutert. Et geometrisk-direkte registrering og georeferering metode er valgt og en algoritme for implementasjon har blitt utarbeided. Den har blitt implementert i MATLAB. Algoritmen har blitt testet i simuleringer, der det ble vist at algoritmen fungerer bra. Argumenter for et snapshot kamera er sammlet og diskutert. Et snapshot kamera tjeneste har blitt implementert i et eksisterende programmvare rammeverk på mål-hardware og suksessfult tatt i bruk.

Preface

This is a masters thesis written as part of the Small Satellite Laboratory at the Norwegian University of Science and Technology, on its first satellite mission, HYPSO. The thesis is about three different, but related areas, registration of push broom satellite images, georeferencing them, and work on a small industrial snapshot camera as secondary satellite payload which can aid the other two topics. The work was done during the spring semester 2019 and is not based on a previous specialization project.

Image registration and georeferencing is based on and related to a number of different disciplines, ranging from optics and remote sensing to image processing, geodesy and even a bit astronomy and orbital mechanics. Additionally, the part about the snapshot camera required some knowledge about development for embedded systems. The topic was very specialized and had little overlap with any of the courses I took during my three years at NTNU. No one told me specifically what to do beyond the topic name, and it was interesting to learn a bit about everything on this very interdisciplinary project.

I would like to thank my supervisor Tor Arne Johansen for guidance on writing, Sivert Bakken for pointer to material I should look at, Magne Hov for help getting my C code running on the embedded hardware. Thanks to Joseph Garrett discussing some aspects of the algorithm with me. He also used parts of the registration algorithm to improve his superresolution algorithm and wrote a paper about it [1].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The NTNU Small Satellite Laboratory plans to build a satellite containing a push broom hyperspectral camera to image the ocean surface and detect targets with a specific spectral signature to support ocean dependent industry like fish farms and to support ground based operations.

The NTNU Small Satellite Laboratory (SmallSat Lab) is a satellite program established through a collaboration of NTNU AMOS, the Department of Engineering Cybernetics and the Department of Electronic Systems at NTNU. Its purpose is to build satellites for the AMOS project Mission-oriented autonomous systems with small satellites for maritime sensing, surveillance and communication (MASSIVE).

1.1 HYPSO Mission

The HYPSO Mission is about building and operating the first satellite that is being build by the NTNU SmallSatLab. HYPSO is an acronym for Hyperspectral Satellite for Ocean Observation. The main payload of the HYPSO satellite is designed to take hyperspectral images of the ocean, with the purpose of achieving scientific goals for oceanography, by imaging and detecting among other things algal blooms, plankton and pollution and providing operational data that aid other robotic platforms in more detailed, in-situ sampling and analysis of these things.

The Satellite is a 6U CubeSat and is planned to be launched into a sun-synchronous circular polar orbit with an altitude of 500km. The launch date is planned to be in Q3 2020. The main focus points of the design at NTNU SmallSatLab are processing of the hyperspectral data, camera calibration and mechanical design of the optics and payloads. The satellite's non-payload subsystems e.g. frame, communication, thermal control, electrical power, housekeeping and attitude control system, are all part of a satellite bus system delivered by Nano Avionics [2].

1.1.1 CubeSats

CubeSat is short for cube satellite. It is a satellite following the CubeSat standard, which defines among other things, size and mass of the satellite. In particular, the standard defines the smallest type of CubeSat to be a cube of 10cm by 10cm by 10cm and at most 1.33kg [3]. Such a satellite cube is called a 1U CubeSat, one Unit. The larger versions have dimensions and mass of stacks of 1U CubeSats, for example a 2U CubeSat is 10cm by 10cm by 20cm with a mass of at most 2.66kg, a 3U CubeSat is 10cm by 10cm by 30cm with a mass of at most 4kg. The dimensions of the HYPSO satellite are 6U, which is 20cm by 10cm by 30cm, the size of two 3U CubeSats side by side, see Figure 1.1. The 6U CubeSat specification has its own document [4]. It specifies the maximum mass to be 12kg, which is 4kg more than the maximum mass of two separate 3U CubeSats. The HYPSO CubeSat is expected to have a mass of less than 8kg.

CubeSats are launched from a small deployer called Poly Picosatellite Orbital Deployer (P-POD), designed to be able to contain multiple CubeSats of different sizes in different combinations during launch and separate them from the carrier rocket.

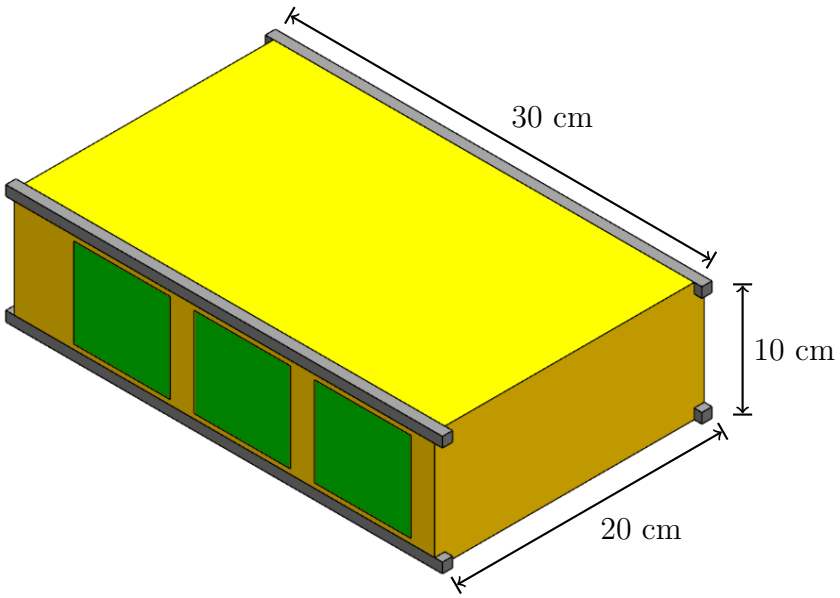


Figure 1.1: The three components of the cubesat geometry. Body (yellow), Rails (gray) and Access Ports (green). Figure from [4]

1.1.2 Satellite imaging systems

There are essentially three types of imaging systems which capture light in the range from infrared to gamma. These are 'whisk broom' scanning systems, 'push broom' scanning systems and framing systems. The whisk broom scanning system has a single detector and it has to scan pixel by pixel to build an image. Scanning is the process of doing repeated

measurements of different areas over time. It is done along two orthogonal directions, usually along track and across track. Along track is the direction parallel to the satellite's orbit motion and across track is the direction perpendicular to it.Some of the detectors on the Landsat satellites are whisk broom sensors [5]. In their design, the satellite's orbital motion does the along track scanning and a rotating mirror does the across track scanning. Push broom scanning systems are equipped with a linear array of detectors, thereby eliminating the need for cross track scanning. An image is built one line at a time, usually by scanning along track. Framing systems are equipped with a 2D array of detectors (area detector array) and capture an image without the need for scanning, an image is built instantly.

1.1.3 Hyperspectral imagers

Independent from the previous three types of image capture architecture, single optical detectors can be classified into three types as well: Monochrome, multispectral and hyperspectral. They lie on a continuum, with monochromatic and hyperspectral at the boundaries and multispectral in between. A monochrome detector is sensitive for a range of wavelengths in some part of the electromagnetic spectrum and produces one value per pixel, representing light intensity. A multispectral detector produces multiple values corresponding to different parts of the electromagnetic spectrum. These values are called channels. A multispectral detector can be build from multiple monochrome detectors, each sensitive to a different wavelength range. An example of a multispectral detector is a RGB camera contained within every smart phone today. A RGB camera has three channels per pixel, one corresponding to red wavelengths, one corresponding to green wavelength and one corresponding to blue wavelengths. Different multispectral camera systems can also contain channels corresponding to near infrared, infrared, ultraviolet or x-ray. With a hyperspectral camera, the number of channels are so numerous so that the spectral dimension of an image can be considered continuous. The resulting data from a hyperspectral camera is a 3D dataset with two spatial dimensions and one spectral dimension. Due to it being a 3D dataset, hyperspectral images are also referred to a hyperspectral datacube, or just datacube. Such a hyperspectral datacube is analogous to an image where every pixel contains data representing a spectrum.

1.1.4 HYPSO satellite concept of operation

A sun-synchronous orbit at a height of 500 km has an orbital inclination of about 97°. An inclination near 90° makes an orbit polar, which means the satellite passes close above earth's poles. Such an orbit is chosen so that the ssatellite is able to scan the Norwegian coast and the arctic ocean. A normal operating procedure is illustrated in Figure 1.2. As the satellite approaches the northern latitudes, it is scheduled to wake up from a low power state and prepare for operational data transmissions from and to ground. Target area coordinates to image are uploaded. Shortly after, the satellite starts to point and

scan the target, in what is called a slew maneuver[1]. The goal is to image a square area of 70km by 70km. After about one minute of scanning, the satellite has time to analyze the data before stepping into contact with the ground again, where relevant information which the satellite was able to extract is downloaded. When the satellite moved too close to the horizon for radio contact, it enters the low power state again, and starts to collect solar energy until the next pass again.

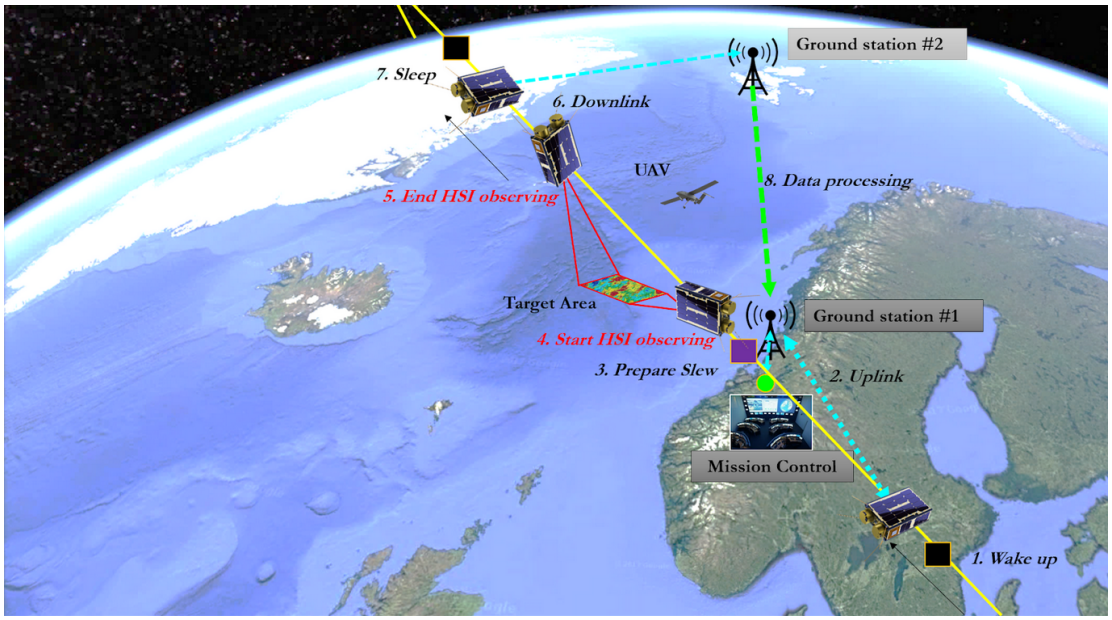


Figure 1.2: HYPSO mission basic concept of operation. Credit: Masiusz Grøtte

Some figures of interest are summarized in Table 1.1.

HYPSO Satellite and Orbit Properties	
Orbit height	500km
Orbit inclination	97°
Payload 1	Push broom Hyperspectral imager
Payload 2	Software defined Radio
Payload 3	Snapshot camera
Cube capture time	57s
Scan FPS	<40
HSI field of view	$8.35° \times 0.02°$
HSI sensor	Sony IMX249
HSI sensor resolution	1936×1216
Platform Sensors	GNSS, IMU, Magnetometer, Star Tracker
Attitue actuators	Magnetorquers, Reaction Wheels

Table 1.1: HYPSO mission, Satellite and orbit properties

[1]Explained in the next section

1.1.5 HYPSO main Payload

The HYPSO main payload consists of two parts, a detector, the hyperspectral imager (HSI) and the data processing part which is the on-board processing unit.

HYPSO hyperspectral imager

The Hyperspectral imager is shown in Figure 1.3. It is build from several objectives (lens assemblies), an aperture in the shape of a slit, a collimator, a diffraction grating and a digital camera.

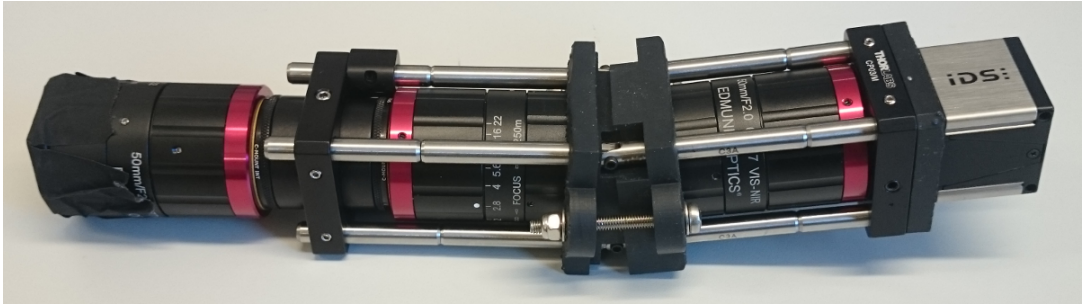


Figure 1.3: The HYPSO hyperspectral imager

The slit determines the spectral resolution and helps to prepare the light for diffraction. After passing through the optical collimator[2], the light is broken up into a spectrum by the grating. The detector is a monochrome area detector array which has a resolution of 1936 by 1216 pixels. The spectrum is spread along the dimension with the 1936 pixels, see Figure 1.4 for an example image. The digital camera is an IDS UI-5260 model with a Sony IMX249 sensor.

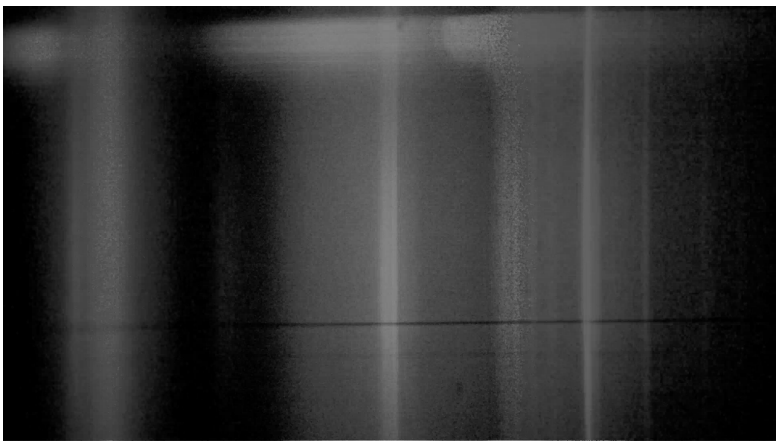


Figure 1.4: What one raw image taken with the hyperspectral imager looks like. Horizontal direction represents wavelength and the vertical dimension represents space. More than 1000 of these images make up one hyperspectral datacube

[2]Collimation is the process of parallelizing the path of multiple rays of light

The field of view of a single scan in a scanning imaging system is called the instantaneous field of view, which is 8.45 degrees by 0.02 degrees in case of the HYPSO HSI. If the satellite points straight down at a height of 500km, it results in a ground sample area of about 74km by 0.17km. Thus the HYPSO satellite must scan at least $\frac{70\text{km}}{0.17\text{km}} = 412$ images to cover the goal area of $70 \times 70\text{km}^2$. Another opjective is to improve spatial resolution to below 100 meters post sampling, by using special data processing techniques, which can combine information contained in partially overlapping ground sample area. The actual number of images captured is thus expected to be above 1000. The satellite's orbital speed is about 7.6km/s, and the satellite moves a ground distance of 70km in about 10 seconds, which means the satellite would need to take 80 images per second to cover 70km with 50% overlap, if it were to rely on its orbital motion alone for along track scanning. 80fps is beyond what is capable of the camera and the on-board processing unit, and thus the satellite must actively control its pointing direction and rotation speed to enable longer scanning time. This is called the slewing maneuver, illustrated in Figure 1.5.

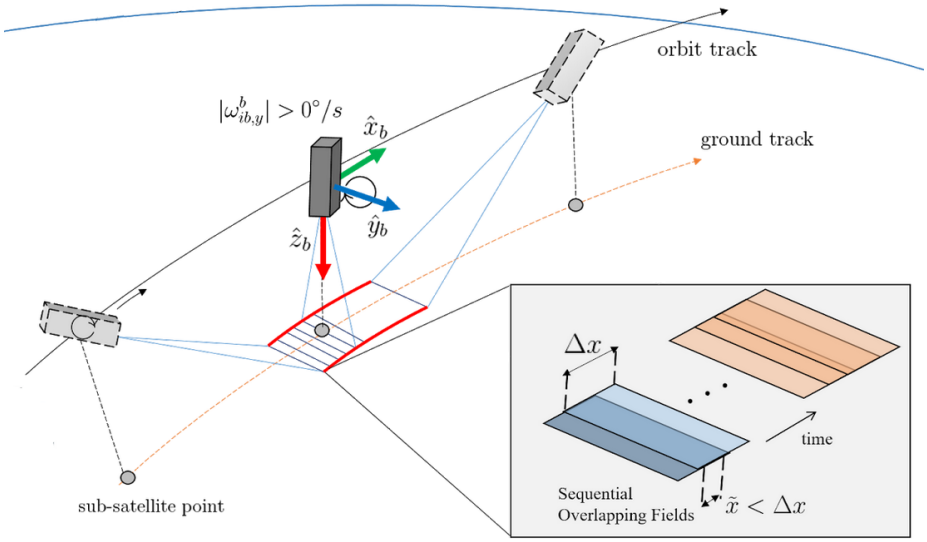


Figure 1.5: Illustrating the slew maneuver. Credit: Mariusz Grøtte

HYPSO on-board processing unit

The On board processing unit will use a module with an ARM Cortex-A9 based system-on-a-chip with embedded FPGA from Xilinx as CPU with 1GB RAM and eMMC storage, Figure 1.6. It will apply a sequence of algorithms to the raw data to for example compress it or extract information. The set of algorithms to be applied in order is called the processing pipeline. It may not be possible for every planned algorithm to be implemented on the satellite. Those that could not, may be applied on a different computer on ground after the data has been downloaded. The stages in the pipeline are listed in the following. Some of them will be briefly described.

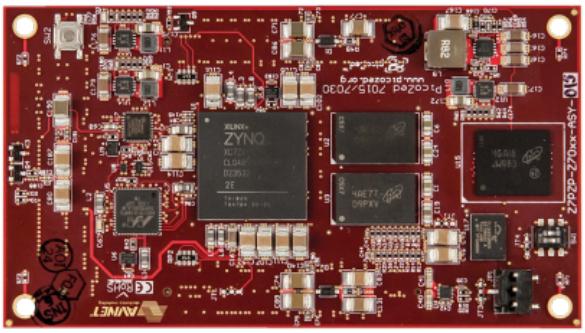


Figure 1.6: The hardware of the On-Board Processing Unit, the PicoZed system on module.

- Binning
- Registration
- Superresolution
- Dimensionality reduction
- Target detection
- Calibration
- Motion blur correction
- Atmospheric correction
- Classification
- Compression

Binning The spectral resolution of the optics as determined by the aperture is about 5nm. The spectum from 400nm to 800nm is spread across 1936 pixels, resulting in equally many channels, and a potential spectral resolution of 0.2nm. This is much higher than the optics and the camera is thereby oversampling and generating a lot of unnecessary data. Binning is the process of averaging neighboring pixels to reduce the number of channels, which reduces the amount of data and incidentally increasing signal to noise ratio, making processing and downlink faster. After binning, one datacube will have between dozens to two hundred channels.

Calibration There are three types of calibration which are to be done. Optical, geometric and radiometric calibration. Optical calibration is correcting nonlinear optics effect, resulting in distortions like for example smile and keystone distortions. Geometric calibration is about precisely figuring out quantitatively where the imager is located inside the satellite body frame, and how it is oriented. Radiometric calibration is about converting the raw binary data to physical units of radiance.

Registration is the processing step which this theses is about

Superresolution is a family of algorithms which increase spatial or spectral resolution of an image post capture.

Atmospheric Correction Ideally, it is desired to measure only the light reflected off the ocean surface. However, the light arriving at the satellite detector inevitably contains light

scattered by the atmosphere as well. This processing step is about removing this light component from the data.

Target Detection determines whether and where a specific target of known spectral characteristic exists within the datacube.

Compression, ideally loss-less compression. Reduces the amount of data that needs to be downloaded by a factor of two to three, making it two to three times faster.

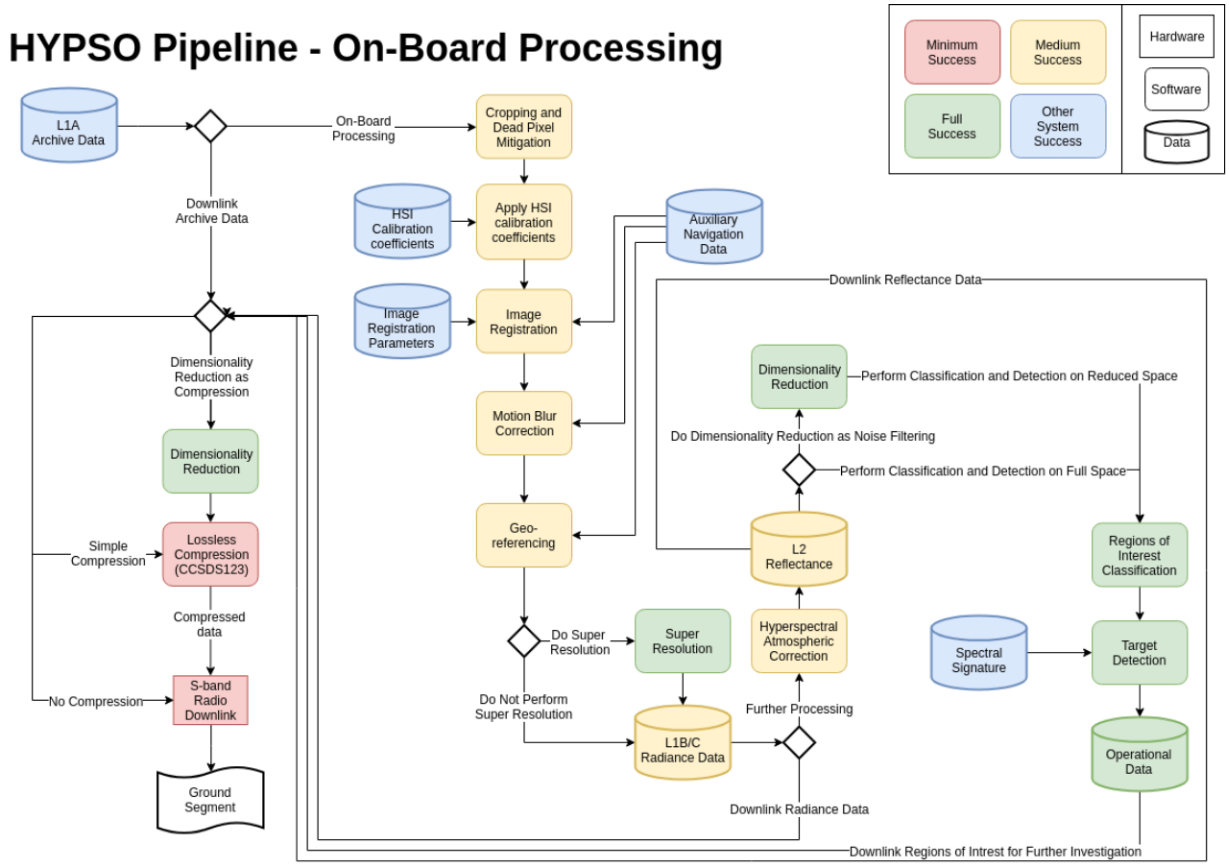


Figure 1.8: Processing pipeline, data levels and success criteria. Credit: Sivert Bakken

1.1.6 Data Products

The data generated by the satellite is allocated into levels where a higher level indicates a higher degree of processedness. There is no conventional way of defining data levels and there are almost as many definitions as there are earth observation satellites. One may define one level after each processing step. A subset of the data levels are to be released for public access. These are the data products of the mission. At the time of writing there are three data products more precisely defined, Level 0, Level 1A and Operational data. Data level 0 consists of the binned data cube. Level 1A consists of Level 0 with ancillary

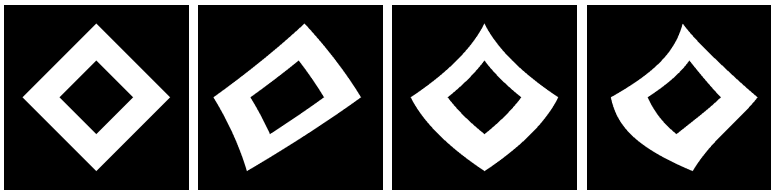
information appended containing at least optical and geometric calibration parameters and ephemeris[3] information during image acquisition. What the Operational data contains exactly is not yet defined and will depend on the types of robotic platforms in operation, but may contain data similar to georeferenced information about ocean color or algae and its amount and type.

1.1.7 HYPSO secondary payloads

In addition to the HSI payload, the HYPSO satellite has room for two secondary payloads. The secondary payloads are a software defined radio (SDR) and the snapshot camera. The purpose of the SDR payload is not relevant to this thesis and will not be further discussed. The purpose of the snapshot camera is discussed in Chapter 4. It is an extra camera in the form of an area detector array, which utilizes extra space in the satellite frame, as well as room in the mass and power budget.

1.2 The push broom registration problem

Push broom imaging systems scan areas line by line and must assemble the lines into an image post capture. If an image is assembled without a special process, by simply appending the lines after each other as they are stored in memory, it can contain distortions resulting from non uniform scanning, see Figure 1.9 and Figure 1.10. Non uniform scanning will happen because of inaccurate control, overlapping or skipping of ground area, changing point of view during scanning and from the curvature of the earth. Registration is a family of image processing techniques that can correct these distortions to generate distortion free, orthorectified images.



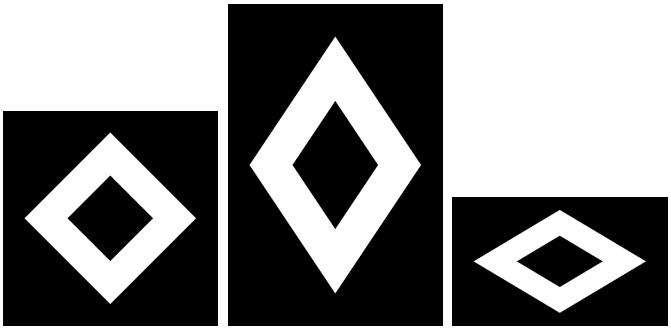
(a) Ground (b) Example 1 (c) Example 2 (d) Example 3

Figure 1.9: Examples of possible distortion resulting from non uniform scanning. (a) Ground truth (b) Distortion due to excessively tilting sideways during sampling. (c) Distortion due to imager changing its distance to the scene during sampling. (d) Combination.

[3]Timestamped position and attitude

The distortions make it harder to

- manually interpret the data and draw conclusions about it and
- integrate the data with other geographic information systems like topographic maps of Norway or maps showing the locations of robotic agents or maps of the ocean floor.



(a) Ground (b) Example 4 (c) Example 5

Figure 1.10: More examples of possible distortions. Stretching effect due to (b) overlapping of lines (c) skipping of lines.

1.3 The georeferencing problem

In order to, for example, direct drones to targets of interest for sample taking or to warn fish farm installations of harmful algae in time, it is necessary to know as well a possible where the targets are and how they are moving. Learning where on earth the targets are is done through a process called georeferencing and estimating how the targets are moving can be done by modelling ocean currents or by repeating georeferenced measurements.

Georeferencing can also mean transformation of spatially extended data (like images) into a specific coordinate system, a geographic map, generated by a map projection.

1.4 Registration and georeferencing methods

Both image registration methods and georeferencing methods are one of two classes: geometric [6] [7] and optical [8] [9]. Geometric methods work by geometric modelling of the image capture system and utilizing extra sensors. optical methods work by comparing the captured images with other available images of the same area. The other images could be for example different satellite images or topographic maps containing recognizable features like roads, rivers, coastlines or houses. The main difference between the two classes is that geometric methods use extra information belonging to the sampled image, for example

taken from attitude and position sensors that were active during image capture, whereas optical methods consider only the content of the captured images.

1.4.1 Image registration

Image registration in general is concerned with finding a transformation, which transforms one image into the coordinate system of the other, so that matching pixels of the two image overlap and correspond to the same physical region of the scene being imaged [10]. One image is said to be registered into another if the transformation has been applied and the image is resampled into the coordinate of the other. Given two images that are to be registered, one of them is chosen as the reference system, called the source or reference image, into which the other image, the target or sensed image, is referenced. An optical image registration algorithm works in four steps [10]

1. **Feature identification** This step considers both images individually and identifies features which can be for example edges, corners, intersection of lines or specific shapes or regions.

2. **Feature matching** Iterates the lists of features and determines whether two features from the two images correspond to each other.

3. **Transformation** Based on the matched features, this step identifies a transformation or transformation parameters that as good as possible transforms the position of all features in one image to the matching position in the other image.

4. **Resampling** The sensed image is resampled by some interpolation scheme using the transformation found in the previous step.

For example, two slightly overlapping images (of e.g. a town) can be registered into one image by finding the position of pixels of the same features e.g. the same houses (feature identification) in both images (feature matching), which can then be used to fit/find the parameters in the transformation model (Transformation) [11]. This transformation can then be applied to all pixels of one image, which transforms them to the corresponding position in the other image (Resampling).

Geometric methods consider 3D space as reference and use position and attitude information about the camera with a camera model to place the pixels where their light was initially reflected from in 3D space. There is not much literature discussing geometric methods in general. This may be as they are all very similar.

The problem of registering more than two images is also called global registration (chapter 5 in [12]), for example during the creation of mosaics of aerial footage. In such a case, the reference may not be derived from another image but defined in some other way.

1.4.2 Georeferencing

Georeferencing by and large is a subset of image registration. Geoferencing an image can be done by registration with a already georeferenced reference [13], thus many georeferencing methods are also image registration methods. In the context of this thesis, georeferencing can refer to two different procedures. The first procedure is only about finding the geographic coordinates of a point withing an image. The second one is about registering an image into the reference system of a geographic map, such that it can be combined with other geographic maps. The second one can be done by both by optical and geometric image registration methods. The first one is purely geometric. Geometric georeferencing is always doing a projection onto an earth model. Geometric georeferencing methods are more commonly called direct georeferencing methods, whereas optical methods are called indirect georeferencing methods.

1.4.3 Methods discussion

Optical or indirect georeferencing and registration methods are common in satellite imagery. However, in the case of HYPSO, the problem is not about registering two or more 2D spatial monochrome or RGB images together, but hundreds of 1D-1D spatial-spectral images or frames. Since optical methods require previously mapped data of the imaged area and the oceans are a changing from day to day, optical methods were considered unsuitable for a ocean observing satellite and geometric methods were favored. This method is similar to other publicized methods [6] [14], as they all consider pixel view directions and earth projections, but this method differs in details corresponding to the specific satellite and camera platform and concept of operation. This kind of attitude based direct georeferencing is especially interesting, since one of the attitude sensors is a star tracker with expected angular resolution of 0.01° or better, and thus attitude is known to a very high degree of accuracy. The image registration algorithm described in this masters thesis has no name, but is similar to the one described in [14]. It is a type of global registration where the reference system is the coordinate system of a projected map. The algorithm will be continued to be referred to as ”the registration algorithm” or just as ”the algorithm”.

1.5 Thesis Structure

Section 2 gives some recommended background info and briefly describes the algorithm in preparation for the next chapter.

Section 3 is the main Section of the thesis. It gives a detailed description of the registration and georeferencing algorithm and discusses its properties.

Section 4 argues for the snapshot camera payload and its possible applications.

Section 5 describes implementation details, shows registration algorithm results and discusses implementation of the RGB camera service.

Section 6 rounds off the report with conclusions.

Chapter 2

Background

This chapter describes some relevant concepts used in the registration algorithm and gives it a brief introductory description.

2.1 Celestial Sphere

The celestial sphere is a 2D spherical coordinate system fixes with respect to the stars. Its coordinates are called right ascension and declination, which are analogous to longitude and latitude coordinates on the earths surface respectively.

2.2 ECI and ECEF frame

When logging satellite positions, they can be given in two possible reference coordinate systems: Earth centered intertial, ECI or earth centered earth fixed, ECEF. As the name implies, both reference frames[1] have their origin at the earth's center of gravity. The ECI frame has its x-axis pointing to the origin of the celestial sphere. This point is called the vernal point, which is sometimes also called vernal equinox, even though vernal equinox specifies a point in time. The intersection of the earth orbital and equatorial plane define a line. This line intersects the celestial sphere at two points, one of them being the vernal equinox and the other the autumnal equinox. At spring equinox, when the line intersects the sun, the earth is opposite side of the vernal point. This defines the vernal point to be near some dozen of degrees south of the Andromeda galaxy. This is the direction in which the x-axis of the ECI frame points. Its z-axis points parallel to earth's rotation axis, and the y axis completes the right handed coordinate system.

[1] reference frame and reference coordinate system mean the same in this Thesis.

The ECEF frame's z-axis also points parallel to earth's rotational axis, but its x-axis is fixed to the earth and rotates with it. Thus it is not an inertial frame. The x-axis points through the prime meridian[2] and the y-axis completes the right handed coordinate system.

Both reference frames collectively are referred to as earth centered frames or ECF.

2.3 Sidereal Time

Sidereal time is an alternative time format that is referenced to earths rotation with respect to the stars, as opposed to solar time with is with respect to the sun. One sidereal day is 4 minutes shorter than a solar day. What makes this a useful concept is that sidereal time is easily converted to an angle describing by what angle the prime meridian is rotated with respect to the vernal point, that is, how the ECI and ECEF frames are rotated with respect to each other, see Table 2.1.

Sidereal Time	Angle between Vernal Equinox and the prime meridian
6h	90°
12h	180°
15.73h	235.95°
18h	270°

Table 2.1: Sidereal time values and their implied angle between the ECEF x-axis and the ECI x-axis. conversion factor $\frac{360°}{24h}$.

2.4 Aircraft principal axes

In 3D space, any rotation can be expressed as a sequence of three rotations around three orthogonal axes. In aircraft dynamics, and analogously spacecraft dynamics, the craft's orientation is expressed in terms of rotations around three principal axes. They axes are called roll-axis, pitch-axis and yaw-axis. The roll axis is usually the axis pointing along the aircraft, the pitch axis points sideways horizontally and yaw points vertically. Let the roll-axis for the HYPSO satellite be defined as its x-axis, let the pitch-axis be the its y-axis and the yaw-axis be its z-axis.

[2]Meridians are circles of constant longitude.

2.5 Algorithm brief description

The inputs to the algorithm are the following data

- Position timeseries from the on board GNSS
- Attitude timeseries from the ADCS subsystem
- Timeseries of datacube scan lines.

It is important that every measurement, every sampling is timestamped or at least it is possible to calculate the timestamps afterwards for example if measurements occur at a uniform rate and the starting time is known. Knowing position, attitude or frame capture at imprecise times will degrade image registration quality. The output of the algorithm is an array of coordinates and a new, resampled datacube. The algortihm is divided into 7 steps:

1. Ephemeris interpolation
2. Camera model and pixel view direction
3. Pixel view ray and intersection with earth model
4. ECF position to geographic coordinates
5. Map projection, boundary and resolution
6. Determining inverse mapping function
7. Backprojection and resampling

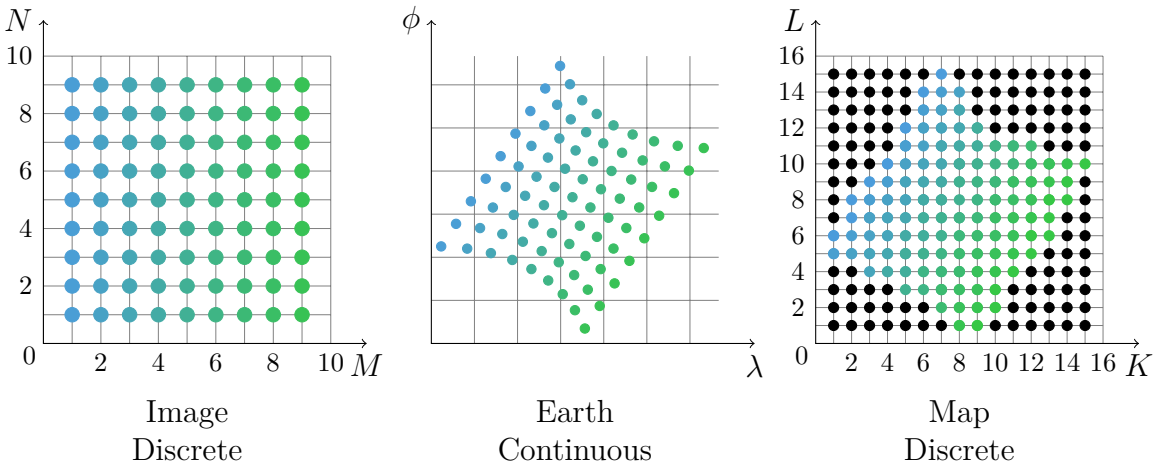


Figure 2.1: Illustrating three stages of the algorithm. Note that map resolution in terms of pixel count is higher than the original image. Each pixel corresponds to a spectrum.

An example of how the algorithm works is shown in Figure 2.1. The plot to the right illustrates the raw datacube. The pixels are arranged on a uniform grid as they are stored in memory, containing distortions. After step 1. through 4., the result looks as the middle plot. Every pixel has assigned its corresponding geographic coordinates. Then steps 5. through 7. generate the plot to the left, where a new image has been generated that is compatible with other maps of the same area. Note that the resolutions of the Image and the Map are different.

Chapter 3

Registration and georeferencing algorithm

3.1 Notation

Let the image dimension be denoted by width M in pixels, which is the along track direction, and height N in pixels which is the across track direction. Thus one frame consists of N pixels, and there are may be M frames in one datacube, though it is possible that M is different for different datacubes. Each pixel within the image is indexed by (m, n) where $m \in \{0, 1, ..., M-1\}$ and $n \in \{0, 1, ..., N-1\}$. One pixel represents roughly 100 spectral values, the exact count will depend on the binning. The registration and georeferencing algorithm does not depend on wavelength, so one spectrum will continued to be referred to as one pixel. The three dimensional spatial coordinates of a pixel projected onto the earth with respect to an ECF are denoted $p_{m,n}$. The corresponding geographical coordinates are $(\lambda_{m,n}, \phi_{m,n})$. The Map dimensions are width K by height L, where L are the pixel count along the north-south direction and K is the pixel count along the east-west direction. One map pixel is indexed by (k, l) where $k \in \{0, 1, ..., K-1\}$ and $l \in \{0, 1, ..., L-1\}$.

Let $\mathcal{P} = \{i \in \mathbb{N} \mid (p_i, t_i^p)\}$ denote the set of all satellite position measurements and their corresponding sample times and $\mathcal{Q} = \{j \in \mathbb{N} \mid (q_j, t_j^q)\}$ denote the set of all satellite orientation measurements and their their corresponding times. Similarly, each frame also has an associated capture time $\mathcal{F} = \{m \in \mathbb{N} \mid (f_m, t_m^f)\}$. The set of all (data, time) tuples is also called a timeseries. Thus there are three relevant timeseries, satellite positions, satellite attitude and hyperspectral image frames. These three timeseries form the input to the algorithm.

3.2 Ephemeris Interpolation

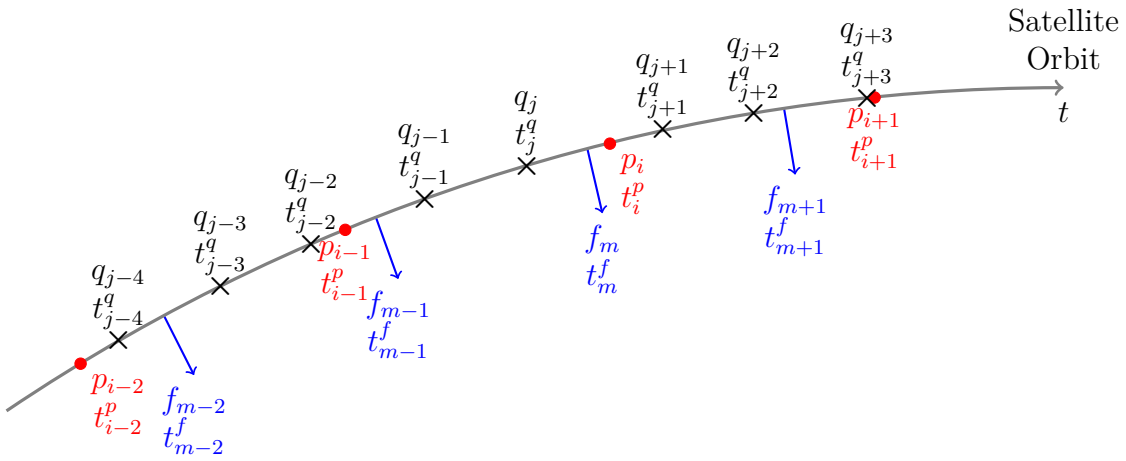


Figure 3.1: Meaning of the three time series, Position $\mathcal{P}$, Attitude $\mathcal{Q}$ and Frames $\mathcal{F}$. The rate at which the qunatities are sampled can be irregular.

The first step is to determine the exact position and attitude at frame capture time. Given frame f_m, it is desired to find index i and j such that

$$t^p_{i-1} \leq t^f_m < t^p_i \tag{3.1}$$
$$t^q_{j-1} \leq t^f_m < t^q_j \tag{3.2}$$

In words, it is desired to find the times in the time series where the first time is is less than or equal the frame capture time and the next time in the time series is larger than the frame time. In other words, it is desired to find the two times in the time series closest to the frame capture time.

The position p_{f_m} and attitude q_{f_m} for frame f_m can then be found by different interpolation schemes:

Nearest neighbor:

$$i_m = \underset{i' \in \{i-1, i\}}{\arg\min} |t^p_{i'} - t^f_m| \tag{3.3}$$
$$p_{f_m} = p_{i_m} \tag{3.4}$$

Linear:

$$p_{f_m} = \frac{t^f_m - t^p_{i-1}}{t^p_i - t^p_{i-1}} p_{i-1} + \frac{t^p_i - t^f_m}{t^p_i - t^p_{i-1}} p_i \tag{3.5}$$

The Linear interpolation scheme and also other schemes like cubic interpolation can be derived using the general Lagrangian interpolation formula [14]:

$$p_{f_m} = p(t_m^f) = \sum_{k=i-\frac{n}{2}}^{i+\frac{n-2}{2}} p_k \prod_{l=i-\frac{n}{2} \mid l \neq k}^{i+\frac{n-2}{2}} \frac{t_m^f - t_l^p}{t_k^p - t_l^p} \tag{3.6}$$

where n is even and represents the number of samples centered around the frame capture time that form the basis for interpolation. Equation (3.6) reduces to Equation (3.5) for $n = 2$. Choosing $n = 4$ results in cubic interpolation. Similar formulas for q_{f_m} apply.

3.3 Pixel view-direction and camera model

From the attitude q_{f_m}, the view direction corresponding to the central view of the camera is calculated, which points along the positive z-axis of the satellite's body frame, Figure 3.2, unless the heavy vibrations during launch changes the exact pointing direction with respect to the satellite frame. Any such pointing direction change can be parameterized using for example two euler angles, which would need to be determined during geometric calibration. Not doing so would degrade georeferencing accuracy, but not registration accuracy, as the error would introduce a constant offset that is equal across all frames, and only their relative position influences registration.

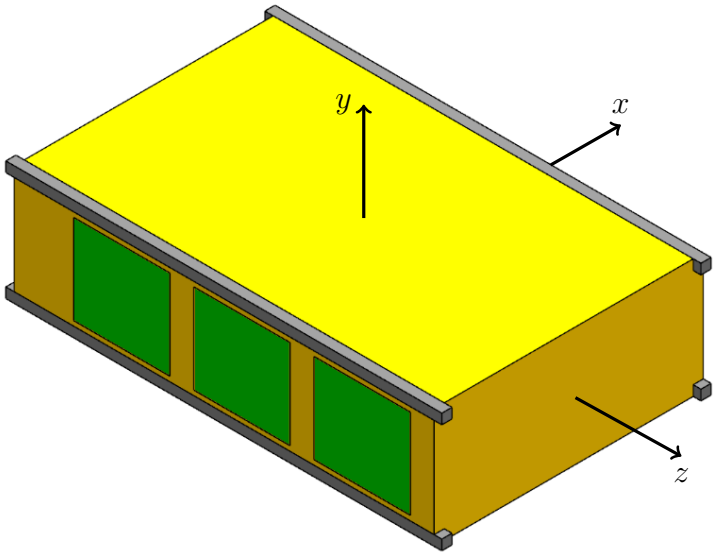


Figure 3.2: Body frame definition of the 6U cubesat

In what way the satellite overall attitude is parameterized has not yet been decided as of the time of writing, but it does not matter much for the algorithm. The only requirement

to the attitude parameterization is that it is possible to to rotate body frame vectors into ECI or ECEF frame, which any valid parameterization is able to do.

Let a quaternion parameterization be assumed, describing how the ECI frame axes need to be rotated to coincide with the satellite body axes. Let

$$q_{f_m} = \begin{bmatrix} \eta_m \\ \epsilon_{1m} \\ \epsilon_{2m} \\ \epsilon_{3m} \end{bmatrix} \tag{3.7}$$

Then the rotation matrix describing the relative rotation is calculated as follows

$$\mathrm{R}_b^f(q_{f_m}) = \begin{bmatrix} 1-2\epsilon_{2m}^2-2\epsilon_{3m}^2 & 2(\epsilon_{1m}\epsilon_{2m}-\epsilon_{3m}\eta_m) & 2(\epsilon_{1m}\epsilon_{3m}+\epsilon_{2m}\eta_m) \\ 2(\epsilon_{1m}\epsilon_{2m}+\epsilon_{3m}\eta_m) & 1-2\epsilon_{1m}^2-2\epsilon_{3m}^2 & 2(\epsilon_{2m}\epsilon_{3m}-\epsilon_{1m}\eta_m) \\ 2(\epsilon_{1m}\epsilon_{3m}-\epsilon_{2m}\eta_m) & 2(\epsilon_{2m}\epsilon_{3m}-\epsilon_{1m}\eta_m) & 1-2\epsilon_{1m}^2-2\epsilon_{2m}^2 \end{bmatrix} \tag{3.8}$$

This rotation matrix transforms body frame vector components to earth centered frame components. If the direction in which the view center points in body frame is the positive z-axis. Thus the inertial frame camera view direction c_m of frame m is calculated by

$$c_m = \mathrm{R}_b^f(q_{f_m})\mathrm{R}_g\hat{z} \tag{3.9}$$

R_g is the rotation matrix calculated from the geometric calibration. If there is no need for geometric calibration or if it was determined that the pointing direction did not change measurably during launch, then $\mathrm{R}_g = I$ and

$$c_m = \mathrm{R}_b^f(q_{f_m})\hat{z} = \mathrm{R}_b^f(q_{f_m}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2(\epsilon_{1m}\epsilon_{3m}+\epsilon_{2m}\eta_m) \\ 2(\epsilon_{2m}\epsilon_{3m}-\epsilon_{1m}\eta_m) \\ 1-2\epsilon_{1m}^2-2\epsilon_{2m}^2 \end{bmatrix} \tag{3.10}$$

The individual pixel directions are calculated by applying a camera model. The pinhole camera model will be assumed, whose geometry is shown in Figure 3.3. The variables in the figure are as follows. d is the width of the camera sensor. f is the camera system focal length. x_1 is the distance from the image center that a point P in the scene, which is z' away and x_2 across, falls on. θ_n is the viewing angle of pixel n. α is the field of view of the camera. Only a one dimensional camera model is considered because one frame has a one dimensional ground footprint.

The spatial dimension of one frame has N pixels, spread uniformly along the sensor size d. Given a pixel number n, the pixel's x-coordinate can be calculated

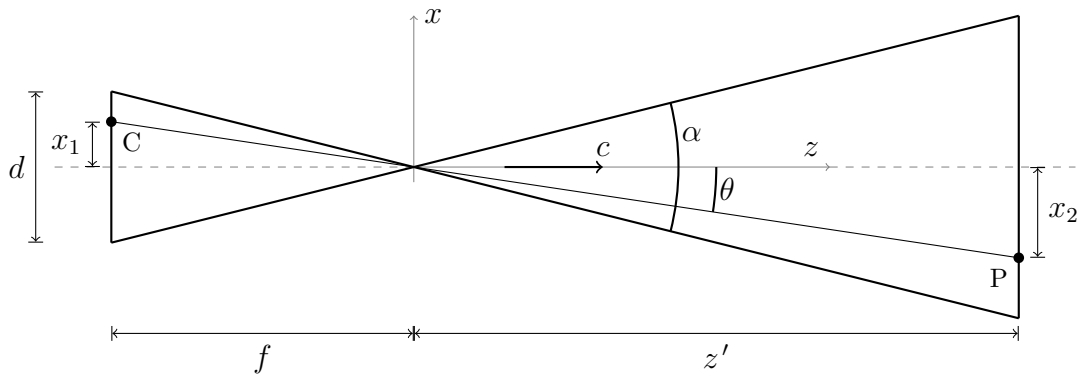


Figure 3.3: Pinhole camera model - Note the x-z coordinate system with origin at the optical center

$$x_1 = \left(\frac{n}{N} - \frac{1}{2}\right) d \tag{3.11}$$

Figure 3.3 provides two relationships:

$$\text{I:} \quad \tan\theta_n = \frac{x_1}{f} \qquad\qquad \text{II:} \quad \tan\frac{\alpha}{2} = \frac{d}{2f} \tag{3.12}$$

Substituting for x_1 in I:

$$\tan\theta_n = \left(\frac{n}{N} - \frac{1}{2}\right)\frac{d}{f} \tag{3.13}$$

we can eliminate the factor $\frac{d}{f}$ using II:

$$\tan\theta_n = 2\tan\frac{\alpha}{2}\left(\frac{n}{N} - \frac{1}{2}\right) \tag{3.14}$$

Thus, given frame pixel index, the pixel view angle θ can be calculated:

$$\theta_n = \arctan\left(2\tan\frac{\alpha}{2}\left(\frac{n}{N} - \frac{1}{2}\right)\right) \tag{3.15}$$

Where $\alpha = 8.45°$ and $N = 1216$. The individual pixel view directions are found by rotating c_n by θ_n around an axis vector normal to c_n and the direction in which the slit is oriented. Since the slit is oriented along $\hat{y}$, the rotation axis is $\hat{x}$. $\hat{x}$ must also be rotated to ECF components

$$\hat{n}_m = \mathrm{R}_b^f(q_{f_m})R_g\hat{x} \tag{3.16}$$

The direction from which pixel (m, n) was captured is thus

$$v_{m,n} = \mathrm{R}(\hat{n}_m, \theta_n)c_m \tag{3.17}$$

where

$$\mathrm{R}(\hat{n}, \theta) = I + S(\hat{n})\sin\theta + S(\hat{n})^2(1 - \cos\theta) \tag{3.18}$$

3.4 Intersection with earth model

3.4.1 Choice of earth model

There were considered three possible models to represent earth geometry. The most accurate models combine the geoid with a digital elevation model (DEM). This would give georeferencing accuracy to within a few meters. However, a DEM is only available on land and since the HYPSO satellite will primarily take images of the ocean, a DEM is not necessary. The other two models were either using the geoid alone or an ellipsoid. A geoid would be most accurate since it represents a level surface of constant gravity, which resembles the ocean surface at equilibrium, only considering the earths mass distribution and earth rotation. The differences between the geoid and the real ocean surface elevation is called ocean topography and is caused by the tides, waves caused through weather effects, ocean currents, heat expansion and contraction and salinity. The ocean topography is in the order of a few meters [15]. The difference between the geoid and a reference ellipsoid are called geoid undulations are vary between ± 60 meters over the oceans [15].

Ultimately, a reference ellipsoid was chosen because of its simplicity and accuracy. An ellipsoid is defined by only two numerical values, for example equatorial radius and eccentricity, while being accurate on the ocean surface to within 60 meters which is roughly the ground footprint of two HSI pixels. The geoid in contrast is a dataset on the order of Megabytes, though this could be reduced if the geoid is chosen to cover just the coast of Norway. It is possible to upgrade the algortihm to use the geoid if it is desirable in the future.

A spheroid is an ellipsoid with two of the three axis radii equal. A spheroid can be either prolate or oblate. It is prolate if the remaining axis radius is larger than the other two, and oblate if it is smaller. Hence earth reference ellipsoids are sometimes also called oblate spheroids.

The International Earth Rotation and Reference Systems Service (IERS) recommends the GRS80 ellipsoid parameters of semi-major axis (equatorial radius) $a = 6378137.0$m and eccentricity squared $e^2 = 0.00669438002290$ [16]. The eccentricity for a spheroid is defined as

$$e^2 = 1 - \frac{b^2}{a^2} \tag{3.19}$$

where a is the equatorial radius and b is the polar radius. Solving for b

$$b = a\sqrt{1 - e^2}$$

The GRS80 parameters thus define the polar radius to be $b = 6356752.3$m. The geometric center of the ellipsoid coincides with the center of mass of the earth. A reference ellipsoid is often alternatively specified in terms of semi-major axis a and inverse flattening factor

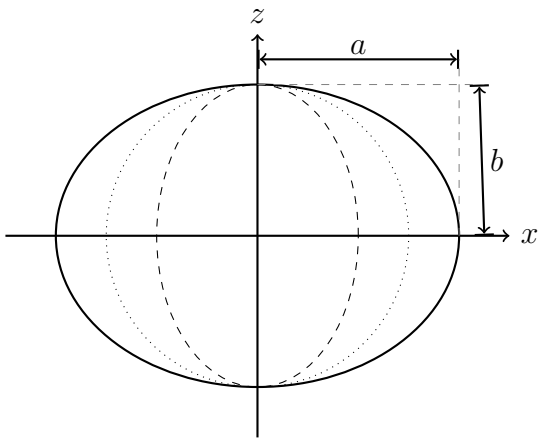


Figure 3.4: Ellipse semi-major axis a and semi-minor axis b, A spheroid is the surface of revolution of an ellipse. Every spheroid is also an ellipsoid. Example of a prolate ellipse - - - - and an oblate ellipse —— with a circle ····· for comparison.

$1/f$. Flattening is defined in terms of semi-major and semi-minor axis

$$f = 1 - \frac{b}{a} \tag{3.20}$$

The relationship between eccentricity and flattening is

$$e^2 = 2f - f^2 \tag{3.21}$$

Which can be easily verified. The equation that defines a spheroid surface in an earth centered frame with z-axis pointing along earth rotation axis is

$$\frac{x^2 + y^2}{a^2} + \frac{z^2}{b^2} = 1 \tag{3.22}$$

or expressed in terms of GRS80 parameters

$$x^2 + y^2 + \frac{z^2}{1 - e^2} = a^2 \tag{3.23}$$

3.4.2 Pixel view ray

A ray is taken from satellite position along the pixel view direction and its intersection with the earth model is to be found. Let the ray be defined by

$$p_{f_m} + t v_{m,n} \ , \quad t \geq 0 \tag{3.24}$$

This is a vector expression with x-, y- and z-components

$$p_{f_m} + tv_n = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ p_z + tv_z \end{bmatrix} \tag{3.25}$$

The ECF coordinates of the ray-spheroid intersection point is found by substituting Expressions (3.25) for x, y, and z in Equation (3.23) and solving for t. Substitution results in

$$\left(v_x^2 + v_y^2 + \frac{v_z^2}{1-e^2}\right)t^2 + \quad 2\left(p_x v_x + p_y v_y + \frac{p_z v_z}{1-e^2}\right)t + \quad p_x^2 + p_y^2 + \frac{p_z^2}{1-e^2} - a^2 = 0 \tag{3.26}$$

$$a\,t^2 \qquad + \qquad b\,t \qquad + \qquad c \qquad = 0 \tag{3.27}$$

This is a second order polynomial with two solutions:

$$t_{min/max} = -\frac{b}{2a} \pm \frac{1}{a}\sqrt{\frac{b^2}{4} - ac} \tag{3.28}$$

The discriminant

$$\frac{b^2}{4} - ac = a^2\left(v_x^2 + v_y^2 + \frac{v_z^2}{1-e^2}\right) - (p_x v_y - p_y v_x)^2 - \frac{(p_x v_z - p_z v_x)^2}{1-e^2} - \frac{(p_y v_z - p_z v_y)^2}{1-e^2} \tag{3.29}$$

characterizes the problem. If

$$\frac{b^2}{4} - ac < 0 \tag{3.30}$$

then the ray does not intersect the earth ellipsoid and there is no real solution. if

$$\frac{b^2}{4} - ac = 0 \tag{3.31}$$

then the ray is tangential to the earth ellipsoid and there is one real solution. If

$$\frac{b^2}{4} - ac > 0 \tag{3.32}$$

then the ray intersects the earth ellipsoid twice. The two solutions correspond to the two intersections a ray can have with an ellipsoid, one on the side of the ellipsoid which is facing the origin point of the ray and one at the other side. The intersection of interest is the intersection closer to the origin point of the ray which is the one corresponding to the smaller value of t

$$t_{min} = -\frac{b}{2a} - \frac{1}{a}\sqrt{\frac{b^2}{4} - ac} \tag{3.33}$$

Substituting t_{min} into Equation (3.25) results in the ECF position corresponding to the pixel.

$$p_{m,n} = p_{f_m} + t_{min} v_n \tag{3.34}$$

3.5 Cartesian coordinates to geographic coordinates

There are two caveats when converting earth centered cartesian coordinate positions to geographic coordinates of longitude and latitude. The first one is that latitude on an ellipsoid can be defined in multiple ways. The other one is the possible conversion from ECI frame to ECEF frame, if the ephemeris data was given in ECI frame.

3.5.1 Going from Cartesian to geographical coordinates

Two kinds of latitudes on an ellipsoid are discussed, geocentric and geodetic latitude. see Figure 3.5. Geocentric latitude is the angle between a line from the origin through A and the equatorial plane. Geodetic latitude is defined as the angle between a normal to the ellipsoid at the intersecting point A and the equatorial plane. Note that on a sphere, the two definitions are equivalent, since the normal and the radius vector at the same point are parallel.

Latitude and longitude are always with respect to some choice of ellipsoid as a representation of the earth. A choice of reference ellipsoid together with a definition of longitude and latitude is called a Datum. In geodesics, geodetic latitude is preferred and thus it is this latitude that the algorithm will use.

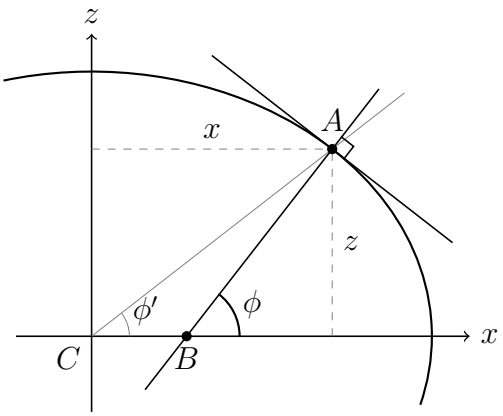


Figure 3.5: Illustrating definition of geocentric ϕ' and geodetic latitude ϕ on an ellipsoid. $\lambda = 0$

The relationship between cartesian coordinates and longitude and geocentric latitude is found by projecting the line CA onto the ECF coordinate axes. They resemble spherical

coordinates.

$$x = r(\phi') \cos \phi' \cos \lambda \tag{3.35}$$
$$y = r(\phi') \cos \phi' \sin \lambda \tag{3.36}$$
$$z = r(\phi') \sin \phi' \tag{3.37}$$

where $r(\phi')$ is some value between the polar and equatorial radius, whose exact value does not matter. λ is longitude and ϕ' is the geocentric latitude. The expressions are inverted to obtain

$$\lambda = \arctan \frac{y}{x} \tag{3.38}$$
$$\phi' = \arctan \frac{z}{\sqrt{x^2 + y^2}} \tag{3.39}$$

Expressions for calculating cartesian position from geodetic latitude and longitude are as follows [17]:

$$x = N(\phi) \cos \phi \cos \lambda \tag{3.40}$$
$$y = N(\phi) \cos \phi \sin \lambda \tag{3.41}$$
$$z = (1 - e^2) N(\phi) \sin \phi \tag{3.42}$$

where

$$N(\phi) = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi}} \tag{3.43}$$

From which the inverse relations can be derived:

$$\phi = \arctan \left(\frac{z}{(1 - e^2)\sqrt{x^2 + y^2}} \right) \tag{3.44}$$
$$\lambda = \arctan \left(\frac{y}{x} \right) \tag{3.45}$$

ϕ is the geodetic latitude. The expression for longitude λ is identical to the previous one.

3.5.2 Converting ECI longitude to ECEF longitude

If the initial ephemeris data was given in ECEF frame, there is nothing more that needs to be done, since the x-axis intersects the prime meridian and thus the angle which the ECEF position projectrd into the x-y plane does with respect to the x-axis is identical to longitude. If the initial ephemeris data was given in ECI frame, then the x-axis is rotated away from the prime meridian around the z-axis by some angle depending on the time at which the frame was captured. One can solve this problem by determining this

angle and subtracting it from the longitude value. This angle can be determined from the corresponding sidereal time at which the line was scanned. The algorithm to calculate Sidereal time from universal time is from [18] and is as follows.

It is given a date and universal time in the following format

$$\text{DD.MM.YYYY - hh:mm:ss.ms} \tag{3.46}$$

First, the Julian Day corresponding to the date DD.MM.YYYY is calculated. If the month M is 1 or 2, i.e. January or February, then replace

$$\text{YYYY} \leftarrow \text{YYYY} - 1 \tag{3.47}$$
$$\text{MM} \leftarrow \text{MM} + 12 \tag{3.48}$$

Next, calculate two intermediate variables

$$A = \left\lfloor \frac{\text{YYYY}}{100} \right\rfloor \tag{3.49}$$
$$B = 2 - A + \left\lfloor \frac{A}{4} \right\rfloor \tag{3.50}$$

The Julian day is as follows

$$\text{JD} = \lfloor 365.25(\text{YYYY} + 4716) \rfloor + \lfloor 30.60001(\text{MM} + 1) \rfloor + \text{DD} + B - 1524.5 \tag{3.51}$$

Then, the sidereal time in seconds s^{0h} corresponding to midnight at the Julian Day is calculated.

$$T = \frac{JD - 2451545}{36525} \tag{3.52}$$
$$s^{0h} = 24110.54841 + 8640184.812866\,T + 0.093104\,T^2 - 0.0000062\,T^3 \tag{3.53}$$

Subtract or add multiples of 86400 (seconds in the day) until $s^{0h} \in \{0, 86400\}$. Lastly, the seconds of the current day since midnight, i.e. the current universal time in seconds, are added to obtain the current sidereal time in seconds.

$$s^{sidereal} = s^{0h} + 1.00273790935(3600\text{hh} + 60\text{mm} + \text{ss.ms}) \tag{3.54}$$

From $s^{sidereal}$ may again be subtracted 86400 such that $s^{sidereal} \in \{0, 86400\}$. This number is then converted to the angle between the prime meridian and the vernal point by multiplying with $\frac{360^\circ}{86400}$ or $\frac{2\pi}{86400}$ and the ECI longitude is converted to proper, ECEF longitude by subtraction

$$\lambda_{ECEF} = \lambda_{ECI} - \frac{2\pi}{86400} s^{sidereal} \tag{3.55}$$

3.6 Map: projection, boundaries and resolution

Everything in the algorithm explained until now is applied per frame and is independent from the other frames in one datacube. The steps that follow however, are dependent on the other frames in one datacube and is applied on a per datacube basis.

At this point every pixel in the datacube has geographical coordinates associated with them in terms of longitude and geodetic latitude:

$$\phi_{m,n} = \arctan\left(\frac{z_{m,n}}{(1-e^2)\sqrt{x_{m,n}^2 + y_{m,n}^2}}\right) \tag{3.56}$$

$$\lambda_{m,n} = \arctan\left(\frac{y_{m,n}}{x_{m,n}}\right) \tag{3.57}$$

$$\lambda'_{m,n} = \lambda_{m,n} - \frac{2\pi}{86400} s_m^{sidereal} \tag{3.58}$$

Next, the pixels are to be resampled such that the image is oriented towards north and is orthorectified. After resampling, the result is an image of an area of the earth represented on a flat surface. Implicit in every such representation is a map projection. A map projection is the mathematical operation that transforms the spheroidal earth onto a flat plane. Any map of the earth has been generated using some choice of map projection or cartographic projection, intentional or not. A map projection is defined as two functions $x = m_1(\lambda, \phi), y = m_2(\lambda, \phi)$ connecting geographic coordinates, longitude and latitude on some choice of an ellipsoid, to 2D cartesian coordinates on a flat map. Some well known map projections are the Marcator projections, the Gall–Peters projection or the natural earth projection.

Any choice of projection is possible. Here, the simplest one is chosen, the equirectangular projection. What makes it the simplest is that the transformation functions are affine[1] and decoupled

$$x = c_1\lambda - \lambda_0 \tag{3.59}$$

$$y = c_2\phi - \phi_0 \tag{3.60}$$

The geographic coordinates $(\lambda_{m,n}, \phi_{m,n})$ for each pixel are transformed to map coordinates.

$$x_{m,n} = c_1\lambda_{m,n} - \lambda_0 \tag{3.61}$$

$$y_{m,n} = c_2\phi_{m,n} - \phi_0 \tag{3.62}$$

The smallest rectangle containing all map coordinates defines the map boundary. Within the set of all map points $\{(x_{m,n}, y_{m,n})\}$, there will be four points with

[1] Linear with translation

• least x	(x_{min}, y_{m_1,n_1})	• least y	(x_{m_3,n_3}, y_{min})
• highest x	(x_{max}, y_{m_2,n_2})	• highest y	(x_{m_4,n_4}, y_{max})

$x_{min}, x_{max}, y_{min}$ and y_{max} define the smallest rectangle containing all points $(x_{m,n}, y_{m,n})$. Let

$$x_{span} = x_{max} - x_{min} \tag{3.63}$$
$$y_{span} = y_{max} - y_{min} \tag{3.64}$$

A uniform grid with some resolution is overlain covering the rectangle and defining the positions and the amount of pixels in the map. Any choice of resolution is valid, but a choice which is high enough, such that the information lost after resampling is minimized is desirable. The number of pixels of the datacube is MN. Let f be a factor defining how many more pixels the resampled image has, i.e. $KL = fMN$. Let $r = \frac{x_{span}}{y_{span}} = \frac{K}{L}$ be the aspect ratio of the resampled image. Then

$$fMN = \frac{K^2}{r} \tag{3.65}$$

and

$$K = \left\lfloor \sqrt{rfMN} \right\rfloor \tag{3.66}$$
$$L = \left\lfloor \frac{fMN}{K} \right\rfloor \tag{3.67}$$

The map pixel $p_{k,l}$ is at position (x_k, y_l)

$$x_k = x_{min} + k\frac{x_{span}}{K} \tag{3.68}$$
$$y_l = y_{min} + l\frac{y_{span}}{L} \tag{3.69}$$

in map space.

3.7 Inverse projection and resampling

The next step is to transform all map grid points $p_{k,l}$ into image space for easier resampling. The inverse mapping function $f^{-1} : \mathbb{R}^2 \to \mathbb{R}^2$ needs to satisfy the following

$$(m, n) = f^{-1}(x_{m,n}, y_{m,n}) \tag{3.70}$$

That is, mapping the projected pixel position back to their respective pixel coordinates. If f^{-1} were known, it would be applied to the map pixels $p_{k,l}$ to find their position in image space

$$(m_{k,l}, n_{k,l}) = f^{-1}(x_k, y_l) \tag{3.71}$$

The spectral values of the map pixels $p_{k,l}$ are found by interpolating the cube pixels nearest to $(m_{k,l}, n_{k,l})$. There are multiple interpolation schemes that can be used as desired. The most common are nearest neighbor, bilinear and bicubic. The interpolation scheme is repeated per channel. These are similar to the schemes in Section 3.2 except two dimensional. Since the cube pixels in image space have integer coordinates, the nearest cube pixels that surround $(m_{k,l}, n_{k,l})$ are easily found. Let $c(k,l)$ denote the spectrum of the map pixel (k,l) that is to be found and $c(m,n)$ the spectrum of cube pixel (m,n). Then by nearest neighbor interpolation

$$c(k,l) = c(\lfloor m_{k,l} \rceil, \lfloor n_{k,l} \rceil) \tag{3.72}$$

where $\lfloor \cdot \rceil$ denotes rounding to nearest integer.

Bilinear interpolation calculates $c(k,l)$ as a weighted average of the four nearest cube pixels. These are

$$(\lfloor m_{k,l} \rfloor, \lfloor n_{k,l} \rfloor) \qquad (\lceil m_{k,l} \rceil, \lfloor n_{k,l} \rfloor) \qquad (\lfloor m_{k,l} \rfloor, \lceil n_{k,l} \rceil) \qquad (\lceil m_{k,l} \rceil, \lceil n_{k,l} \rceil) \tag{3.73}$$

or since the cube pixels in image space have integer coordinates

$$(\lfloor m_{k,l} \rfloor, \lfloor n_{k,l} \rfloor) \quad (\lfloor m_{k,l} \rfloor + 1, \lfloor n_{k,l} \rfloor) \quad (\lfloor m_{k,l} \rfloor, \lfloor n_{k,l} \rfloor + 1) \quad (\lfloor m_{k,l} \rfloor + 1, \lfloor n_{k,l} \rfloor + 1) \tag{3.74}$$

let

$$c_1 = c(\lfloor m_{k,l} \rfloor, \lfloor n_{k,l} \rfloor) \tag{3.75}$$
$$c_2 = c(\lfloor m_{k,l} \rfloor + 1, \lfloor n_{k,l} \rfloor) \tag{3.76}$$
$$c_3 = c(\lfloor m_{k,l} \rfloor, \lfloor n_{k,l} \rfloor + 1) \tag{3.77}$$
$$c_4 = c(\lfloor m_{k,l} \rfloor + 1, \lfloor n_{k,l} \rfloor + 1) \tag{3.78}$$

$$s_1 = m_{k,l} - \lfloor m_{k,l} \rfloor \tag{3.79}$$
$$s_2 = n_{k,l} - \lfloor n_{k,l} \rfloor \tag{3.80}$$

then

$$c' = s_1 c_1 + (1 - s_1) c_2 \tag{3.81}$$
$$c'' = s_1 c_3 + (1 - s_1) c_4 \tag{3.82}$$

And finally

$$c(k,l) = s_2 c' + (1 - s_2) c'' \tag{3.83}$$

Bicubic interpolation calculates a weighted average similarly except using the 16 nearest pixels.

If a map pixel falls outside the image, that is, if one or more of the following is true for $p_{k,l}$

$$m_{k,l} < 1 \qquad m_{k,l} > M \qquad n_{k,l} < 1 \qquad n_{k,l} > N \tag{3.84}$$

Then the corresponding map pixel spectrum is set to black.

3.8 Approximating the inverse projection function

The inverse mapping function will be different for each datacube and needs to be determined anew for each. A method inspired from [14] is used, where the inverse mapping function is approximated as a polynomial. The inverse mapping function $f^{-1} : \mathbb{R}^2 \to \mathbb{R}^2$ has two components, $f_m^{-1} : \mathbb{R}^2 \to \mathbb{R}$ and $f_n^{-1} : \mathbb{R}^2 \to \mathbb{R}$. They are defined as

$$f_m^{-1}(x, y) = \sum_{i=0}^{r} \sum_{j=0}^{i} a_{i,j} x^j y^{i-j} \tag{3.85}$$

$$f_n^{-1}(x, y) = \sum_{i=0}^{r} \sum_{j=0}^{i} b_{i,j} x^j y^{i-j} \tag{3.86}$$

which are rth degree polynomials and $a_{i,j}$ and $b_{i,j}$ are each $\frac{(r+1)(r+2)}{2}$ unknown polynomial coefficients. Since it is required that

$$m = f_m^{-1}(x_{m,n}, y_{m,n}) \tag{3.87}$$

$$n = f_n^{-1}(x_{m,n}, y_{m,n}) \tag{3.88}$$

The polynomial coefficients can be determined by solving the following unconstrained optimization problems

$$\min_{\{a_{i,j}\}} \sum_{m=1}^{M} \sum_{n=1}^{N} |f_m^{-1}(x_{m,n}, y_{m,n}, \{a_{i,j}\}) - m|^2 \tag{3.89}$$

$$\min_{\{b_{i,j}\}} \sum_{m=1}^{M} \sum_{n=1}^{N} |f_n^{-1}(x_{m,n}, y_{m,n}, \{b_{i,j}\}) - n|^2 \tag{3.90}$$

And this was the last piece in the registration and georeferencing algorithm. Examples where the algorithm is applied are presented in Section 5.1.3.

3.9 Geolocation accuracy analysis

In this section some expected error sources that can degrade the accuracy to which pixels are assigned geographical coordinates are listed.

3.9.1 Geoid Undulations

As was mentioned in Section 3.4.1, the difference between the geoid and the ellipsoid can be up to 60m. The geolocation error that will result from that is shown in Figure 3.7.

If an area is scanned from directly above and the imager points straight down with no nadir angle, there will be no error, no matter the undulation. Assuming the maximum undulation of 60m, when the nadir angle is 18.1° as it is at the start of a normal slewing maneuver, a point that is imaged from A will be through to be at B. The error will be about 20m. When the nadir angle is 61.8° in the case of a revisit maneuver, point A will be thought to be at C. The error will be about 112m.

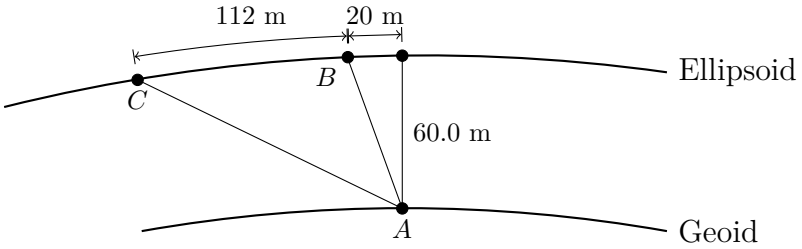


Figure 3.7: Using an ellipsoid as earth model, a pixel whose real footprint is at A is believed to be at B or C

This source of error may be mitigated by choosing a reference ellipsoid that is more accurate in the specific are that is to be imaged.

3.9.2 Geometric calibration

If the imager moved relative to the body frame due to vibration or stresses during launch or thermal expansion and contraction due to temperature variations in orbit, then the pointing direction will be different from expected. in the same way as with geoid undulations, the error will be amplified with larger nadir angle.

The snapshot camera can aid in geometric calibration as it can give an absolute metric on where the satellite is pointing. The difference between the attitude measurement from the ADCS subsystem and the attitude from there the snapshot camera image can characterize the geometric error.

3.9.3 Nonlinear optics

Varying angular increment between directions of neighboring pixels i.e. deviations from the pinhole camera model. This may be an effect that changes with varying temperature. Can be mitigated by good optical calibration on ground.

3.9.4 Finite precision effects

32bit floating point numbers lose precision in the 7th or 8th significant digit. Consider the following example. At Norwegian latitudes of about 65°, the following c program demonstrates the smallest increment from 65°:

```
#include <stdio.h>
main()
{
    float a = 65.000000f;
    float b =  0.000005f;
    printf("\%.10f\n\%.10f\n", b, b+c);
}
```

The output is

```
65.0000000000
65.0000076294
```

Thus the smallest increment from 65 is 0.0000076294, which corresponds to a location change on the earth of 0.8m, which is not so large that it needs to be worried about, however to calculating the sidereal time involves parameters which are given up to the precision of 14 significant digits, which does require higher precision, as [18] recommends. The Xilinx Zynq 7030 is based on the ARMv7-A architecture, which is a 32bit architecture and therefore would need to emulate higher precision floating point arithmetic in software, which is much slower. It would be reasonable to use 32bit floating point arithmetic for all parts of the algorithm for speed, except where higher precision is needed, as with the calculation of sidereal time, where emulated 64bit double precision floating point arithmetic should be used.

3.9.5 Earths Precession and nutation

An example in [18] shows that neglecting Earth's precession and nutation will result in a sidereal time error of about 0.2 seconds, which leads to a positional error of 9.2 meters. It is not difficult to include the correction as [18] explains how.

3.9.6 Imperfectly fitted inverse model

Can be mitigated by choosing a different degree of polynomial or by changing the optimization algorithm that solves Equation (3.89) and (3.90).

3.10 Combining Registration with Superresolution

Since superresolution includes a resampling step as well as the image registration, they may be combined into a single algorithm as in [1], in the future.

3.11 Registration of images of the moon

The satellite may for calibration purposes sometimes take images of the moon. The previously described method of registration will not work because the view directions do not intersect the earth model. Instead, one could use a sphere centered on the satellite with radius equal to or higher than the satellite's distance to the moon and calculate the view directions intersection with that sphere instead. This emulates the celestial sphere as reference system and all other parts of the algorithm can be used without change.

Chapter 4

Snapshot Camera Payload

This chapter discusses an extra camera payload in the form of a snapshot or full frame camera that utilizes extra space as well as free room in the mass and power budget of the satellite. There are three potential purposes for using using a snapshot camera.

1. Enabling the use of the processing technique panchromatic sharpening
2. Possible alternative georeferencing and registration
3. Other verification and redundancy purposes

4.1 Panchromatic Sharpening

Panchromatic sharpening, or pansharpening in short, is an image processing technique used to increase the spatial resolution of a multi or hyperspectral image, using a different monochrome image of higher spatial resolution taken of the same scene. The monochrome image has to be taken from a camera that is panchromatic. A panchromatic camera is a camera that is sensitive to all visible light wavelengths, hence the name panchromatic sharpening. More specifically, it must be sensitive to all wavelength in the multi- or hyperspectral image.

To optimally utilize an additional camera for pansharpening in the HYPSO satellite, the camera would need to be monochrome, it would need to be sensitive to light in the band from 400nm to 800nm with a known spectral response and it would need higher spatial resolution than the HSI. There are two parameters that influence the spatial resolution of a camera: sensor resolution, as in how many pixels the senor has, and field of view. Increasing increasing sensor resolution will increase the spatial resolution and increasing field of view will decrease spatial resolution. Thus the snapshot camera needs to either have:

- same sensor resolution as the HSI with lower field of view
- a higher resolution sensor with the same field of view

Decreasing the field of view of a camera below the 8.45 degrees of the HSI may put harder demands on the camera optics, making it more expensive and easier to break during launch. In addition, the panchromatic image would cover only a subset of the area which the hyperspectral datacube coveres, making pansharpening possible only to to this area within the datacube or introduces the need for a second imaging maneuver which captures more panchromatic images. The concept of operation would need to be adjusted to make time for such a maneuver.

A sensor with higher pixel count would be more expensive but would make the sensor not necessarily easier to break. It would generate more data, but monochrome images require only little data storage space compared to hyperspectral datacubes.

4.2 Georeferencing

For georeferencing, it is desirable to have a large field of view to maximize the potential for capturing geological features which can act as features for optical registration. As the satellite is designed to take images of the ocean, the most relevant geological features are coastlines. Higher spatial resolution will be useful as well, since it will increase potential for feature matching and increase georeferencing accuracy.

It may also be possible use the snapshot camera for image registration, since the snapshot camera images are captured nearly at the same time as the hyperspectral datacube, thus they might share features. The datacube is thereby georeferenced , if the snapshot camera image is georeferenced.

4.3 Redundancy and verification purposes

Including an extra non-hyperspectral camera as payload in the satellite would require much less design and implementation work than the HSI camera and would be simpler to operate, since it can be based on commercial, off-the-shelf components. In contrast, the HSI payload due to its complexity, has a higher chance of failure. A simple additional camera would not leave the satellite useless, in case of main payload failure. In addition, a single image captured before and after the slew maneuver would take only a few seconds, so the concept of operation would need very little adjustment, the snapshot images would add very little data compared with the compressed datacube and the camera would use a negligible amount of energy because it would be shut off or idling most of the time

A secondary, non hyperspectral camera could speed up the commissioning phase as verification of the attitude control system is made easier, since monochrome or RGB images show spatial features in an easily human identifiable form. To guarantee a functioning camera, space proven hardware, that is, a camera module or sensor that has been operated on a satellite before is desirable.

4.4 Discussion

Georeferencing and panchromatic sharpening are mutually exclusive. Capturing coastlines for georeferencing requires a high field of view that will degrade the spatial resolution. A high spatial resolution requires a low field of view that will lead to coastline being out of view most of the time. Considering that this is the first satellite being built by NTNU SmallSatLab, it is more important to build something that is will be functioning than to add even more features to the already packed processing pipeline. A simple camera with no extreme specifications will be robust and have a higher field of view than the HSI.

4.5 Chosen RGB camera

(a) Front of the camera (b) Back of the camera

Figure 4.1: The RGB Camera IDS UI-1250LE-C-HQ

4.5.1 Camera desciption

A different student working on a specialization project made a trade-off analysis between a number of different cameras. The result was the camera shown in Figure 4.1. Its CMOS sensor is the EV76C570. A sensor of the same series EV76C560 was confirmed to have been used successfully in a different satellite mission before [19]. The sensor is of class 1/1.8" which means its size is about 7.2mm×5.4mm. It has a resolution of 1600×1200 with a bayer

filter, see Figure 4.2. In essence, every RGB camera is also just a monochrome camera, except the presence of a color selective filter on the sensor with a special post processing technique called debayering or demosaicing. The camera uses USB2.0 as interface. It weighs 34.35g, though its plastic case will need to be replaced due to outgassing reasons as determined by the HYPSO mechanical team. Its energy use is specified to range from 300mW to 700mW

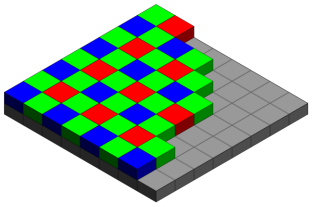


Figure 4.2: RGB bayer pattern filter on a camera sensor. By en:User:Cburnett, *Link*, CC BY-SA 3.0

4.5.2 Objective

An objective that fits to the camera has been found using the Lens Finder online tool of the camera manufacturer IDS. Its the Tamron M118FM08. It has a focal length of 8mm, variable focus and aperture, and is designed for the optical class/sensor size 1/1.8".

The field of view has been roughly measured to be 57° horizontal and 43.6° vertical. The measurement method was mounting the camera on a tripod and pointing it to a whiteboard, then turning on a live feed from the camera using tools provided by the camera manufacturer and noting on the whiteboard where the view boundaries of the camera are. Measuring the distance of the camera to the whiteboard, and the horizontal and vertical distance between the markings, the field of view can be calculated. With a field of view of 57° × 44° and pointing nadir, the ground sample area is about 543km ×404km.

(a) The chosen lens for the RGB camera

(b) Camera with lens assembled

Figure 4.3: Camera with lens Tamron M118FM08

Chapter 5

Implementation and Results

This section discusses implmentation and verification of the registration and georeferencing algorithm and the implementation of the RGB camera service on a ZedBoard running Petalinux.

5.1 Registration algorithm results

The registration algorithm is tested by comparing how well it can reconstruct an image after a simulated push broom scanning operation.

5.1.1 Image capture details

When the satellite passes directly overhead, it is only visible for about 12 minutes over the horizon. Shorter passes are more likely. It is determined that in an average pass, there is time for about 57 seconds of target scanning. Due to the satellite being such a small size, there are limits on energy generation and storage. There is room for only one scan operation per pass. The rest of the pass is spent harvesting solar energy.

In a circular orbit, the satellite's speed is constant and given by

$$v = \sqrt{\frac{GM_e}{r}} \tag{5.1}$$

where $G = 6.67 \times 10^{-11} \frac{\mathrm{m}^3}{\mathrm{kg\,s}^2}$, is the gravitational constant and $M_e = 5.972 \times 10^{24}$ kg is the mass of the earth and $r = 6871$ km. This gives about 7.6km/s for the HYPSO satellite.

From the height of the orbit, the period is calculated.

$$P = 2\pi\sqrt{\frac{r^3}{GM_e}} \tag{5.2}$$

which results in about 94 minutes.

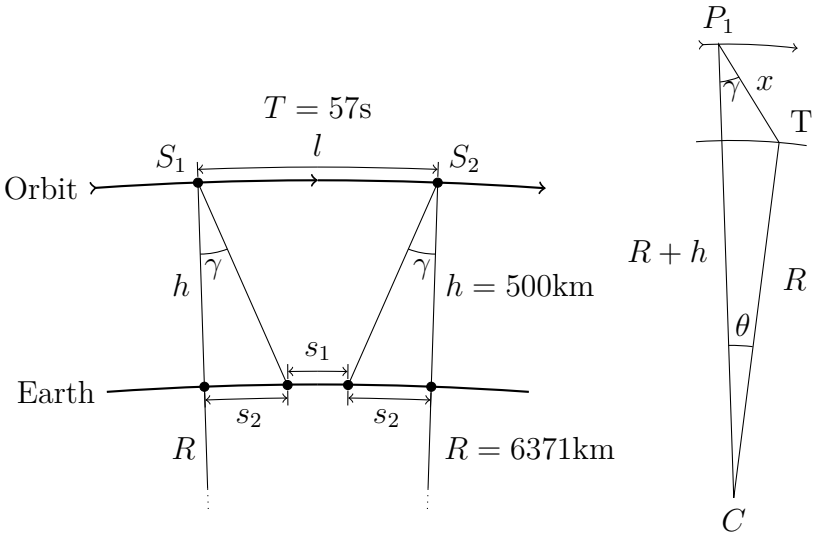


Figure 5.1: Geometry of a longitudinal view of nominal cube capture

Since the orbital speed was 7.6 km/s, during the $t = 57$ seconds of scanning, the satellite moves a distance of $l = vt = 434$km. The target area is 70km along track. This gives the scene geometry during a slew maneuver shown in Figure 5.1. The ground track is

$$s_1 + 2s_2 = 402\text{km} \tag{5.3}$$
$$s_1 = 70\text{km} \tag{5.4}$$
$$s_2 = 166\text{km} \tag{5.5}$$

Using that $\theta = \frac{s_2}{R}$ and using the cosine rule on triangle S_1CT the distance x is found. Then the initial along track nadir angle [1] is found using the sine rule to be about $\gamma = 18.1°$.

[1] Nadir is the direction from the satellite to the center of the earth. The nadir angle is an angle between nadir and the satellite's pointing direction.

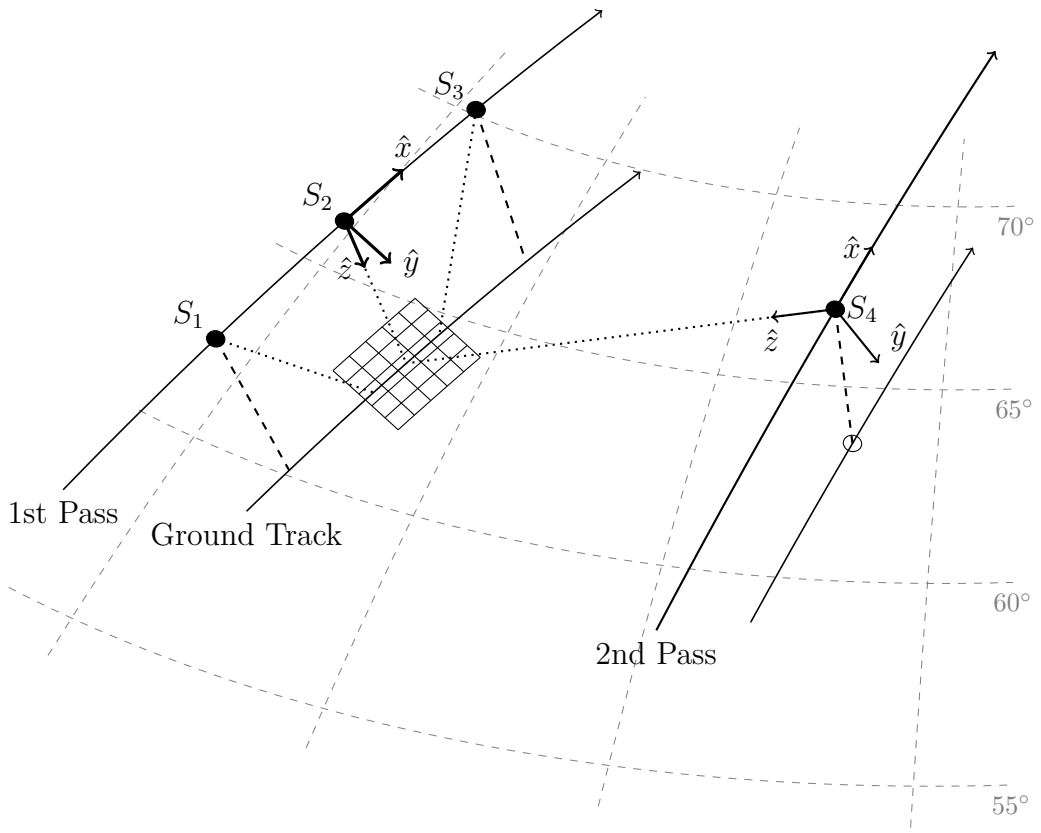


Figure 5.2: Recapturing the same target on second pass

A different maneuver as a followup to a previous one, is to scan the same area as the previous pass to measure changes, Figure 5.2. In this case, the satellite would need to point far off nadir across track, because during the 94 minutes it takes for the satellite to return, the target has moved due to the rotation of the earth. This distance is approximately

$$d = P_{orb}\,\omega_e R\cos(\phi) \tag{5.6}$$

where $P_{orb} = 94\text{min}$, ω_e is the earth's angular velocity and ϕ is the latitude of the target. The maximum possible distance, when the target lies on the equator, is 2664km. At Norwegian latitudes of about 65° the distance is 1126km. Using the cosine rule and sine in a similar way as previously, the required nadir angle to point 1126km cross track is found to be 61.8°.

The maximum cross track nadir angle is such that the outermost pixel direction is tangential to the earth surface, see Figure 5.3. A higher angle would mean that one border of the image is imaging the outer space background. From the figure it can be seen that

$$\sin\left(\gamma + \frac{\text{fov}}{2}\right) = \frac{R}{R+h} \tag{5.7}$$

Which gives a maximum cross track nadir angle of $\gamma = 63.8°$. Which means the nadir

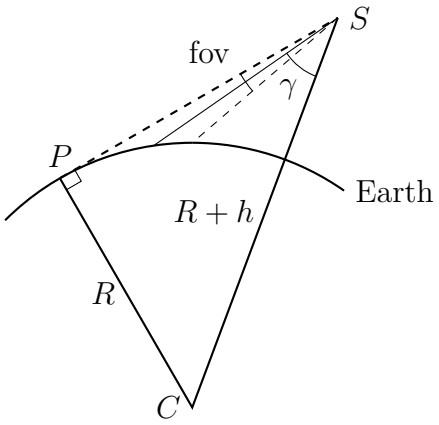


Figure 5.3: Deriving maximum nadir angle

angle required for the revisit maneuver of 61.8° is close to the threshold of what is possible to scan where only the earth in view.

Some derived quantities are summarized in Table 5.1.

HYPSO Satellite Numbers of interest	
Orbital speed	7.6km/s
Orbital Period	94 min
Starting Nadir Angle	18.1°
Ending Nadir Angle	−18.1°
Max Nadir Angle	63.8°

Table 5.1: Derived HYPSO mission properties

5.1.2 Simulating an image capture maneuver

The simulated image capture maneuver and the registration and georeferencing algorithm were implemented in MATLAB. During the simulation, the earth is assumed stationary. The satellite positions are given in ECEF frame. The orientations parameterized as body frame z- and x-axes given in ECEF frame. The orbit is assumed circular.

The parameters that are required for the simulation are listed in Table 5.2:

From the orbital period, the true anomaly change during the maneuver is calculated

$$\Delta\theta = 2\pi\frac{T}{P} \tag{5.8}$$

Radius of orbit	r	6871 km
Duration of maneuver	T	57 s
Number of frames	M	>1
Number of pixels in frame	N	>1
Initial longitude	λ_0	$(-180°, 180°)$
Initial latitude	ϕ_0	$(-90°, 90°)$
Move direction	β_0	$(0°, 360°)$

Table 5.2: Parameters from which the simulated datacube capture is determined

The starting position of the maneuver is

$$p_{f_0} = \begin{bmatrix} a\cos\lambda_0\cos\phi_0 \\ a\sin\lambda_0\cos\phi_0 \\ a\sin\phi_0 \end{bmatrix} \tag{5.9}$$

The initial move direction, which is identical to the initial orbit tangent unit vector is

$$\hat{v}_{f_0} = \begin{bmatrix} \sin\lambda_0\sin\beta_0 + \cos\lambda_0\sin\phi_0\cos\beta_0 \\ \cos\lambda_0\sin\beta_0 + \sin\lambda_0\sin\phi_0\cos\beta_0 \\ \cos\phi_0\cos\beta_0 \end{bmatrix} \tag{5.10}$$

The orbit normal unit vector is calculated

$$\hat{n} = \frac{p_{f_0} \times \hat{v}_{f_0}}{||p_{f_0} \times \hat{v}_{f_0}||} \tag{5.11}$$

An initial nadir pointing unit vector is as follows

$$\hat{z}_{f_0} = \begin{bmatrix} -\cos\lambda_0\cos\phi_0 \\ -\sin\lambda_0\cos\phi_0 \\ -\sin\phi_0 \end{bmatrix} \tag{5.12}$$

The true anomaly change $\Delta\theta$ is divided into $M-1$ intervals

$$\delta\theta = \frac{\Delta\theta}{M-1} \tag{5.13}$$

And the satellite positions for each frame capture are

$$p_{f_i} = R(\hat{n}, i\delta\theta)p_{f_0}, \quad i \in \{0, 1, 2, ..., M-1\} \tag{5.14}$$

where $R(\hat{n}, i\delta\theta)$ is a rotation matrix corresponding to rotation around axis $\hat{n}$ by angle $i\delta\theta$. A sequence of nadir vectors $\hat{z}_{f_i}$ and tangent vectors $\hat{v}_{f_i}$ are found similarly, i.e. by rotating their respective initial vectors by $R(\hat{n}, i\delta\theta)$. The vectors $\hat{v}_{f_i}$, $\hat{n}$ and $\hat{z}_{f_i}$ represent the satellite body x-axis, y-axis and z-axis respectively.

The MATLAB function `fit()` was used to solve Equation (3.89) and (3.90).

5.1.3 Results

In the following, four different simulated area scanning maneuvers are presented. Three of them are with excessive pitching, excessive rolling and excessive yawing respectively and one is with an extreme nadir angle. As ground truth for the simulations is the Sentinel satellite image shown in Figure 5.4. The first maneuver resembles a slewing maneuver, with too high sample density during the middle of the maneuver. The fourth maneuver resembles when the satellite is to scan the same area again in the following pass with a nadir angle of 63°. The second and third maneuver may resemble other kinds of faulty attitude control.

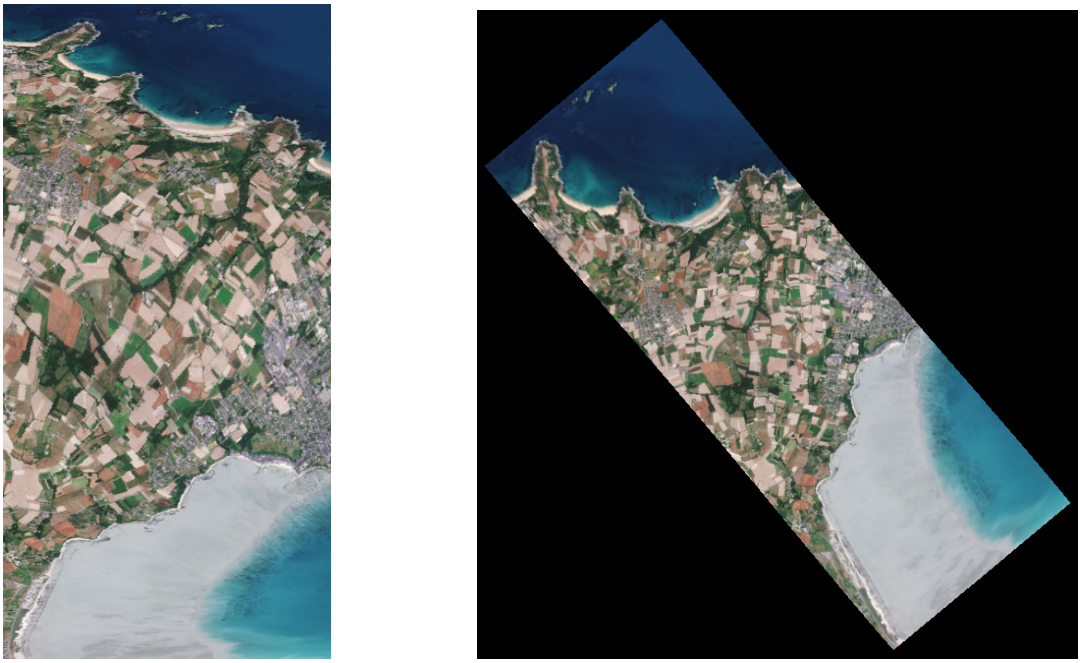
For each of the four maneuvers, five figures are shown. The first illustrated the how the image is sampled during the simulation, and where in the ground truth image it was sampled. The second and third image show the image before and after the algorithm was applied. The fourth figure illustrates the state of the algorithm after the first four stages of the algorithm, after all pixels have been ground projected and a map grid has been overlain. The last figure shows the map pixel position in image space after they have been inverse projected.

All four simulated cases demonstrate well that the algorithm manages to remove major distortions from the raw images.

Figure 5.4: Ground truth for simulated image capture. Credit: ESA/Sentinel-2 - CC BY-SA IGO 3.0

5.1.4 Maneuver 1 - Pitching

Figure 5.5: Ground sample points of pitching maneuver



(a) Raw image (b) Registered and georeferenced

Figure 5.6: Before and after applying the algorithm

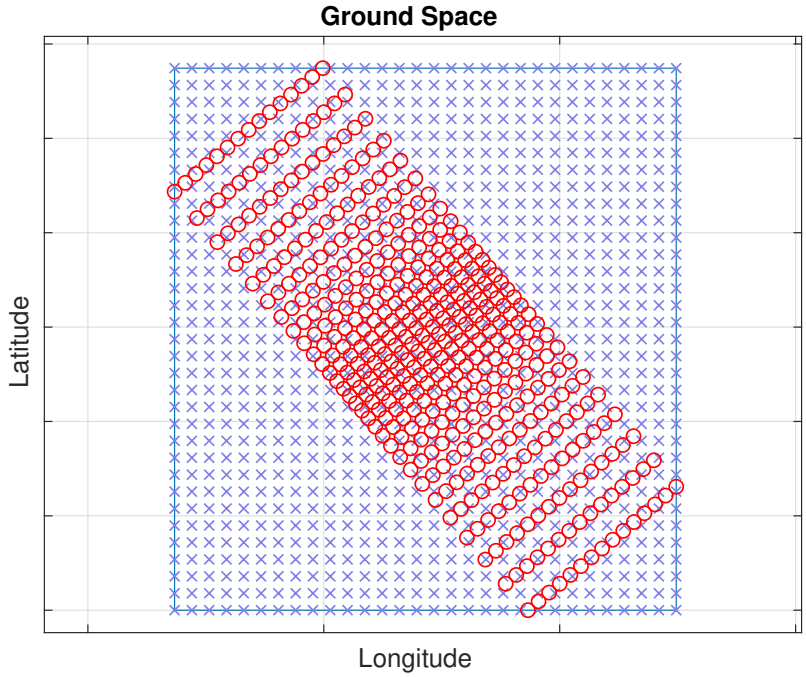


Figure 5.7: Ground space pixel positions

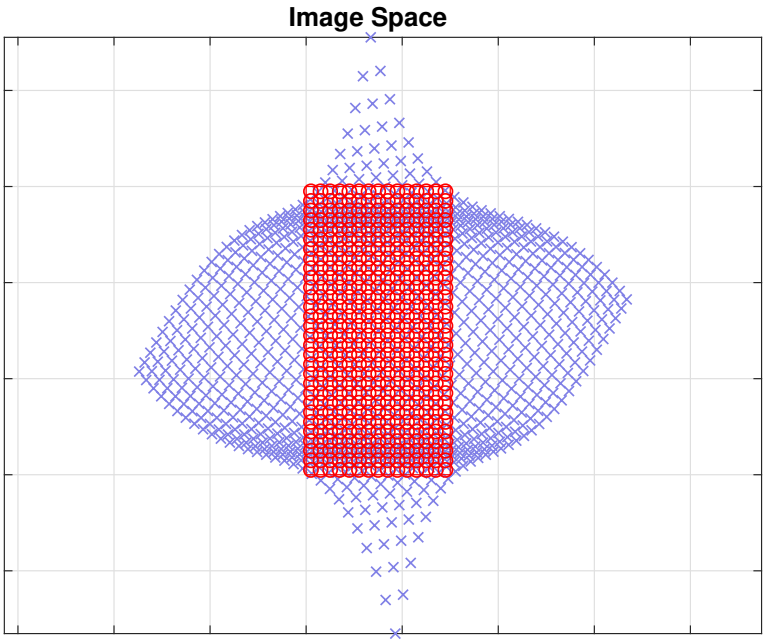
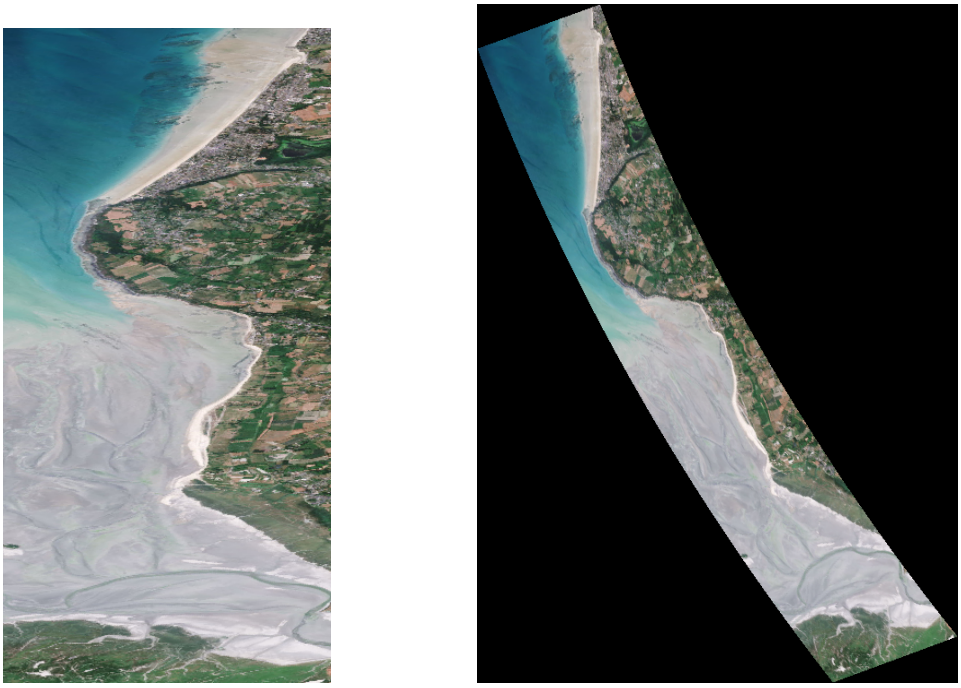


Figure 5.8: Image space pixel positions

5.1.5 Maneuver 2 - Rolling

Figure 5.9: Ground sample points of maneuver



(a) Raw image (b) Registered and georeferenced

Figure 5.10: Before and after applying the algorithm

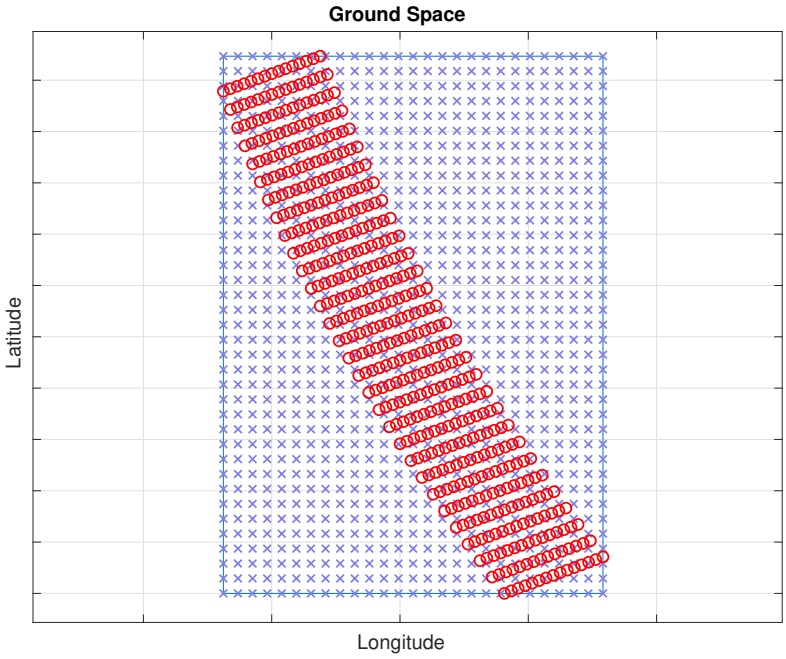


Figure 5.11: Ground space pixel positions

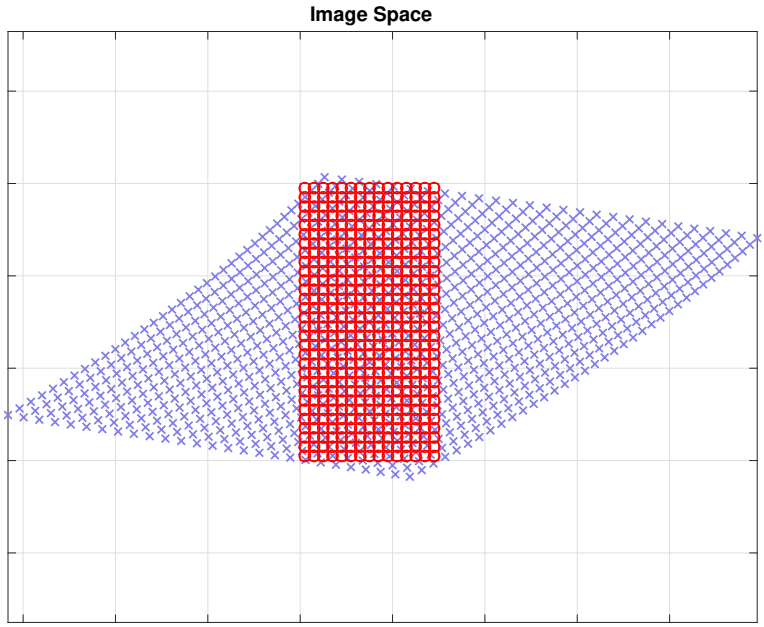
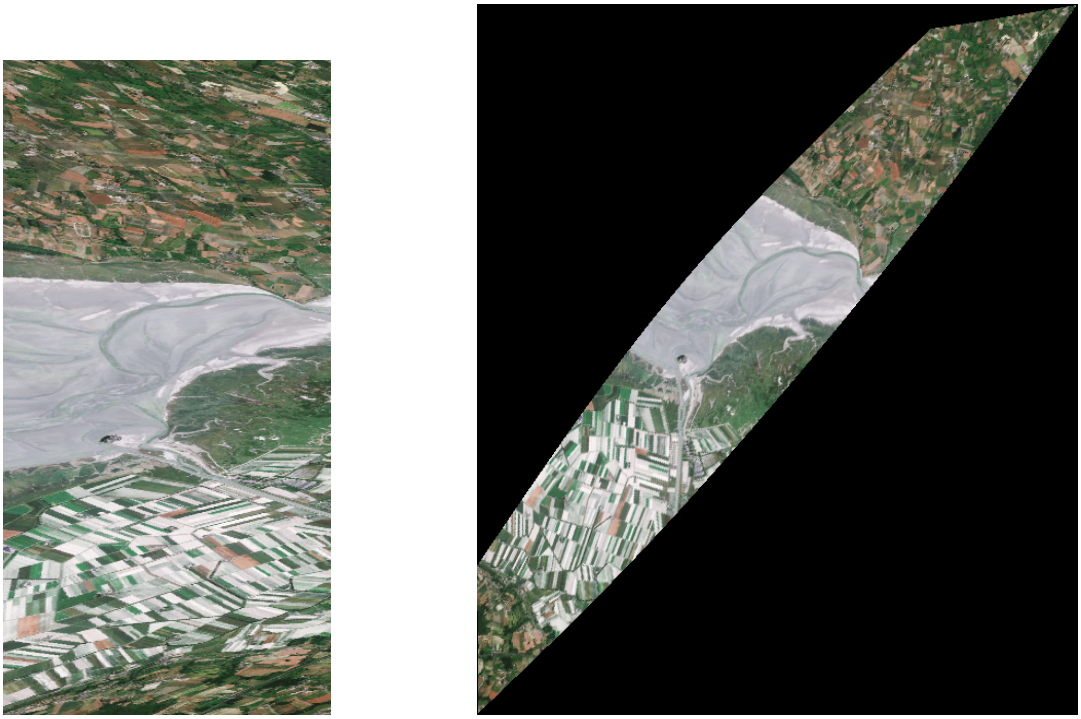


Figure 5.12: Image space pixel positions

5.1.6 Maneuver 3 - Yawing

Figure 5.13: Ground sample points of maneuver



(a) Raw image (b) Registered and georeferenced

Figure 5.14: Before and after applying the algorithm

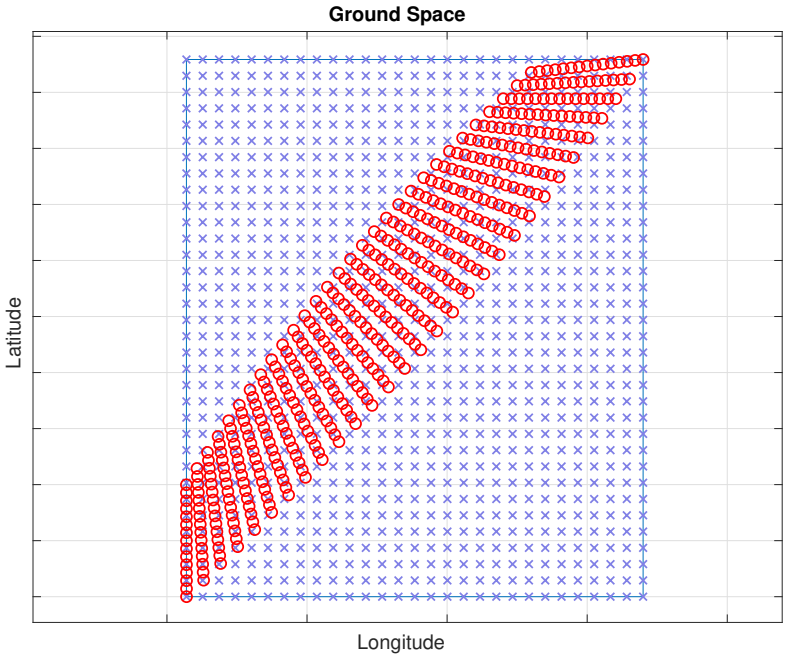
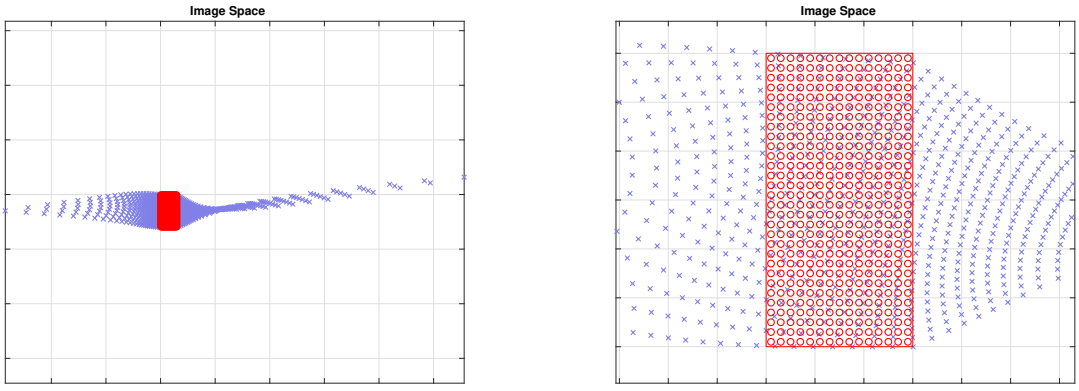


Figure 5.15: Ground space pixel positions



(a) Every pixel (b) Zoomed in

Figure 5.16: Image space pixel positions

5.1.7 Maneuver 4 - extreme nadir angle

Figure 5.17: Ground sample points of pitching maneuver

(a) Raw image

(b) Registered and georeferenced

Figure 5.18: Before and after applying the algorithm

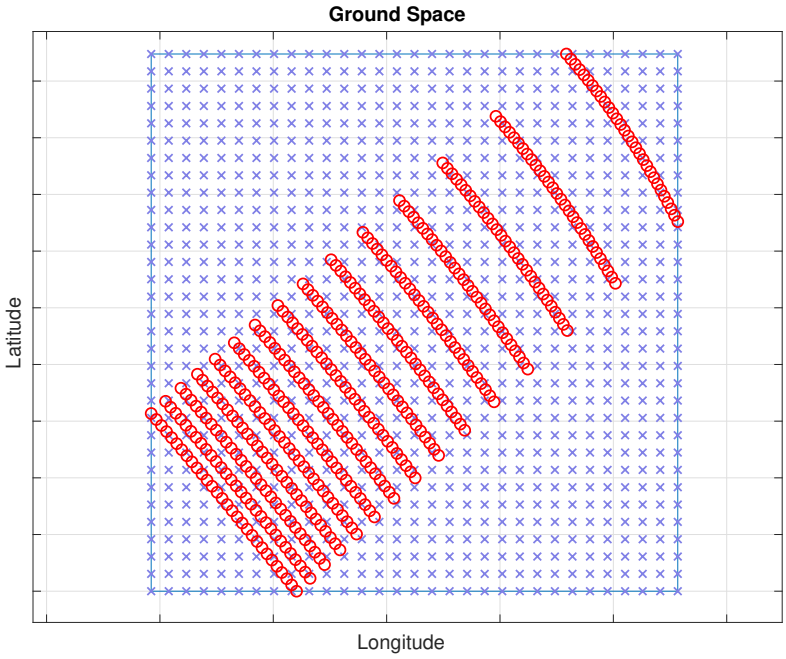
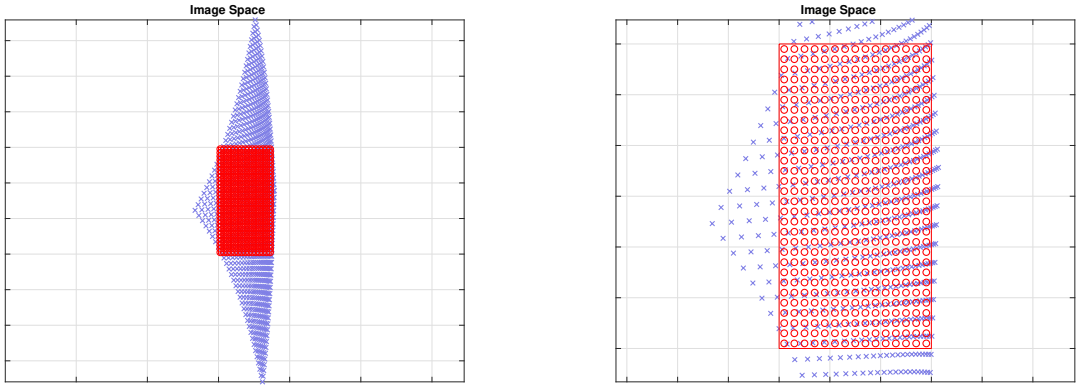


Figure 5.19: Ground space pixel positions



(a) Every pixel (b) Zoomed in

Figure 5.20: Image space pixel positions

5.2 RGB Camera

The software framework between the ground station and satellite is formed by two programs, `hypso-cli` and `opu-services`. `hypso-cli` is the command line interface running on the ground station which is used to send commands and configurations to the satellite. `opu-services` is running on the on-board processing unit and handles the configurations and commands the satellite receives from ground station. There is among others a service for file transfer, a service for running the HSI. This section will be about the RGB camera service.

5.2.1 RGB service

The RGB service is started as a thread by `opu-services`. The camera manufacturer IDS provides the `ueye` library containing an API to interface with the camera from C code. When commands sent by `hypso-cli` are prefixed with `rgb`, they are sent to the RGB service. There are five commands. `rgb init`, `rgb configure`, `rgb configfile`, `rgb capture` and `rgb deinit`.

`rgb init`, see Figure 5.21. Sets the camera from idle into a standby mode, where it is receptive to other commands and loads a pre generated textfile containing a complete camera configuration.

`rgb configure` or `rgb configfile`, see Figure 5.22. Change some specific configuration parameters manually or reload a different configuration file.

`rgb capture`, see Figure 5.23. Initiate the capture of one image. The image capture is optionally externally triggered, if it is desired to capture the image at a different point in time than when the software issue the command. The image can be read in a raw, monochrome bayer pattern image, or as a RGB image, the save file format .bmp, .png or .jpg can be specified and the path where to save the image.

`rgb deinit`, see Figure 5.24. Puts the camera from Standby back into idling, ready to be shut off.

Ideally, the RGB camera will be connected to its own power rail on the EPS that can be turned on and off. Turning power on would be the first action in a normal image capture sequence, which may continue as follows. The rgb init command is issued, which puts the RGB camera into a command receptive state. Then commands that configure individual parameters or a commands that read a file in which the whole configuration is stored can be issued, if the default configuration is not desired. Then the image capture command is issued with its respective command arguments. The camera then proceeds to capture and save an image to the specified file path. The deinit command puts the camera into a pre-initialzed state ready to have power turned off again.

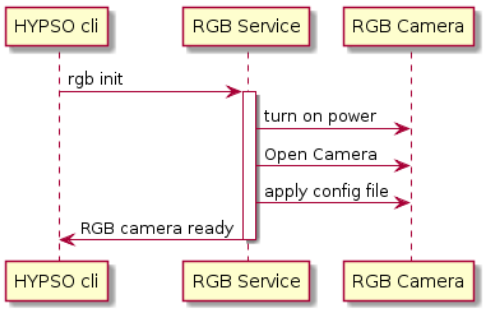


Figure 5.21: `rgb init` command sequence diagram

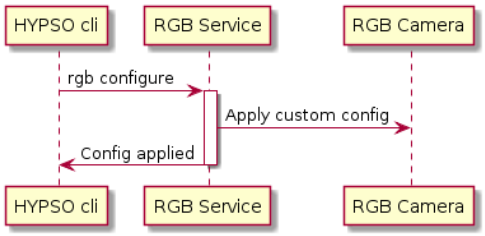


Figure 5.22: `rgb configure` (or `rgb configfile`) command sequence diagram

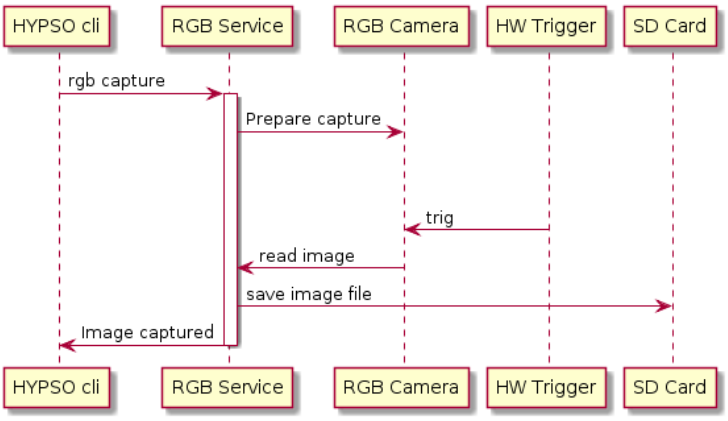


Figure 5.23: `rgb capture` command sequence diagram

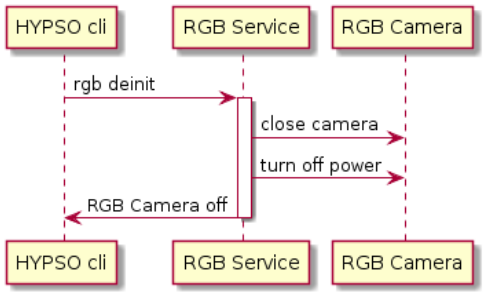


Figure 5.24: `rgb deinit` command sequence diagram

The following energy usage has been measured during the different modes of operation and configurations of the camera, see Table 5.3. It shows how pixel clock influences energy use.

Operational Mode	Current [mA]	Power [mW]
Idle	55.0	275
Standby	75.5	377.5
Capturing 10MHz	125	625
Capturing 24MHz	139	695
Capturing 35MHz	148	740

Table 5.3: RGB Camera energy usage. Supply voltage of 5V. The frequency denotes Pixel Clock.

5.2.2 Camera configuration

Parameter	Datatype	Unit	Range	Default Value
Pixel Clock	integer	MHz	10 - 35	24
Frame Rate	float	Hz	1.0 - 16.9	-
Exposure	float	s	0.02 - $\frac{1}{FrameRate}$	$\frac{1}{FrameRate}$
Gamma	integer	-	60-220	100
Per channel analog gain	integer	-	0-100	R: 11 G: 0 B:14

Table 5.4: Some configurable parameters of the IDS UI-1250LE-C-HQ Camera

Pixel Clock Determines how fast data is read out from sensor. Higher values lead to higher power consumption. Must be set to maximum value to enable highest framerate.

Frame Rate When in video mode, determines how often an image is read from the sensor. Video mode is not planned to be used

Exposure How long in milliseconds the sensor is collecting light. Higher values makes the image brighter. Must be set with considereation about the lens aperture.

Gamma A per pixel mathematical operation: $p_{out} = p_{in}^{\frac{\gamma}{100}}$. A value for γ of around 180 improves picture quality. There is also a hardware gamma setting of a fixed value of 160.

Analog Gain Changes channel brightness. Configure to adjust color balance.

5.2.3 RGB camera test

A development kit with the same family of processor has been used as test hardware, Figure 5.25, on which `opu-services` ran. An instance of `hypso-cli` ran on an Ubuntu laptop and was communicating with the ZedBoard via its CAN interface using a USB to CAN

converter. In image captured with the RGB camera of a test printed pattern is shown in Figure 5.26a. A raw image pre-debayering is shown in Figure 5.26b. If we zoom into the bayer pattern image, the pattern becomes noticeable, Figure 5.27.

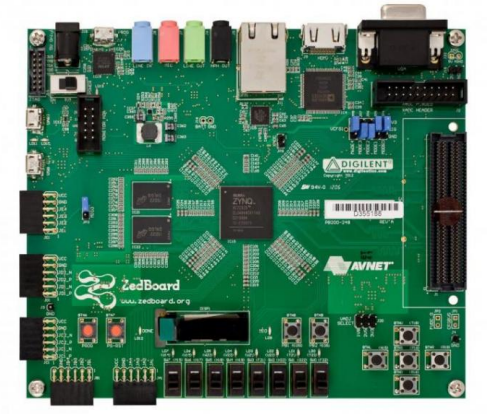
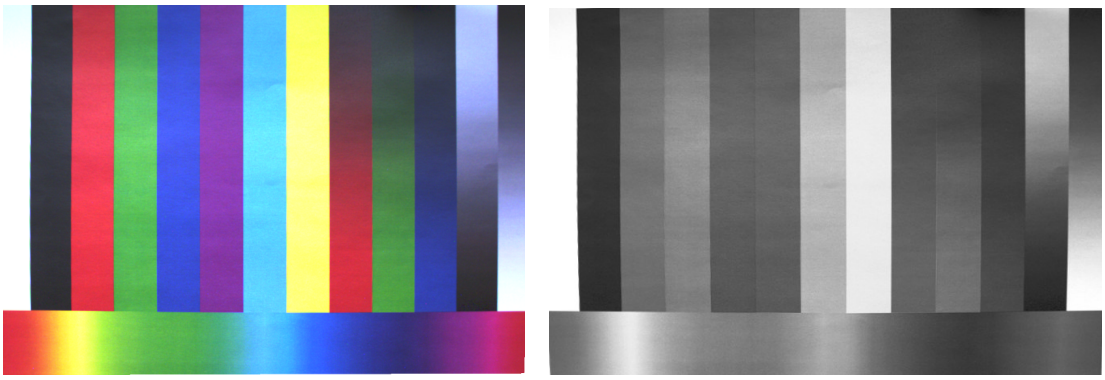


Figure 5.25: Avnet ZedBoard development kit



(a) Debayered/Demosaiced (b) Bayer pattern image

Figure 5.26: Cropped image of a printed test pattern under cloudy sunlight. Hardware gamma on.

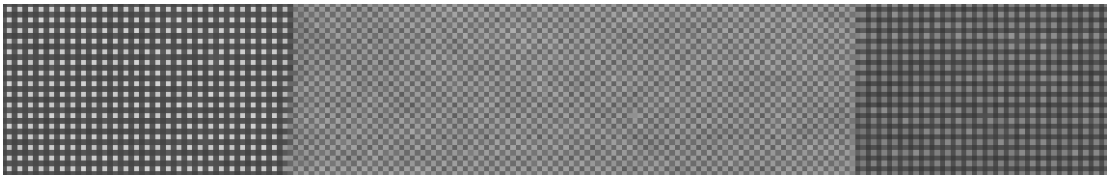


Figure 5.27: Bayer pattern image zoomed in at the Red-Green-Blue part of the test pattern

Chapter 6

Conclusions

Image registration alone is not a critical part of the HYPSO processing pipeline. Targets within the datacube can be georeferenced without applying the full algorithm. It is however always nice to have possibilities and properly registered and georeferenced images can be combined with other geographic information systems from which better conclusion may be drawn

The algorithm is working well and can remove distortions from raw images scanned through many different kinds of irregular sampling. It remains to be seen whether its possible to fully implement it on the on-board processing unit while still being practical for the operations, because the polynomial fitting and resampling part of the algorithm at full resolution of about 1216×1500 takes almost a minute, even on a laptop and that was not even with hyperspectral images, but only three channeled RGB images. It is however fully possible to assign geographic coordinates to the detected targets using on board processing power.

6.1 Future work

There are two important things that should be done next. Implementing the algorithm on target hardware and using it on real data instead of simulated data, either from other similar satellite missions or from drone operations. Applying the algorithm on real data may uncover important details which could be algorithm breaking if left unconsidered.

In addition, with the RGB camera working, one can start investigating how to best register the images from the HSI into the RGB images, if it turns out to be doable.

Appendices

A Algorithm MATLAB Code

Listing 6.1: Function to rotate a vector around axis by angle. Implements Equation (3.18)

```
function vr = rotateV(v, n, a)
%ROTATEV Rotates vector v around axis n by angle a
%   both v, n, and vr are column vectors. n is a
%   unit vector, a is in radians
    n = n/norm(n);
    vr = v*cos(a) + n*(n'*v)*(1-cos(a)) + cross(n, v)*sin(a);
end
```

Listing 6.2: Function to find the ray-spheriod intersection

```
function surfacePoint = projection(satPos, ViewDir)
%PROJECTPOINT traces a ray from satPos along ViewDir and returns
% intersection point of ray with oblate spheroid

%   Projecting includes solving a second order polynomial
%   c1 is the linear coefficient
%   c2 is the constant coefficient
%   t is found using the p-q formula and describes how long along
%   viewDir needs to be traced until intersection point.

%   will return satPos if there is no intersection with the earth

% Earth properties
earthEquRadius = 6378137;

% Oblate spheroid paramteres
```

```
a = earthEquRadius;
f = 0.003452810665;     % earth flattening
f_ = 1/((1 - f)^2);

% x^2 + y^2 +f_*z^2 = a^2

% Projecting ray onto spheroid
h = 1 - (1-f_)*ViewDir(3)^2;
c1 = (ViewDir(1)*satPos(1) + ViewDir(2)*satPos(2) + f_*ViewDir(3)*satPos(3))/h;
c2 = (satPos(1)^2 + satPos(2)^2 + f_*satPos(3)^2 - a*a)/h;


if c1*c1-c2 < 0 % no intersection case
    surfacePoint = satPos;
else
    t = -c1-sqrt(c1*c1-c2);
    surfacePoint = satPos + t * ViewDir;
end

end
```

Listing 6.3: Script to calculate sidereal time

```
% Date
Y = 2019; M = 6; D = 24;
% Time of day
h = 23; m = 47; s = 58;

if M < 3
    Y = Y - 1;
    M = M + 12;
end

A = floor(Y/100);
B = 2-A+floor(A/4);

JD = floor(365.25*(Y+4716)) + floor(30.60001*(M+1))+D+B - 1524.5;

% 0: sidereal time at midnight at the day corresponding to julian day JD
T = (JD - 2451545) / 36525;
s0h = 24110.54841 + 8640184.812866*T + 0.093104*T*T - 0.0000062*T*T*T;
while s0h < 0.0
```

```
    s0h = s0h + 24*3600;
    disp('oi1');
end

while s0h > 24*3600
    s0h = s0h - 24*3600;
    disp('oi2');
end

% sidereal time at time s seconds UT since midnight at julian day JD
s_sdrl = s0h + 1.00273790935*(3600*h + 60*m + s);

if s_sdrl > 24*3600
    s_sdrl = s_sdrl - 24*3600;
end

hrs = floor(s_sdrl/3600);
mns = floor((s_sdrl-hrs*3600)/60);
scs = floor(s_sdrl-hrs*3600-mns*60);

disp(['Sidereal Time: ' num2str(hrs) ':' num2str(mns) ':' num2str(scs)]);
```

Listing 6.4: Script which simulates the image capture

```
%% Setup - variable definitions
tic
% load ground truth image
im = imread('simGroundTruth.jpg');

% camera field ofview
fov = 8.45*pi/180;
% earth flattening
f = 0.003452810665;

a = 6871000;                              % approximate semi major axis of orbit
P = 2*pi*sqrt(a*a*a/(5.972e+24*6.67e-11));  % orbital period
T = 0*60 + 57;                            % image sampling duration in seconds
Dth = 360*T/P * pi/180;                   % true anomaly change during sampling

% sampled image dimensions
N = 30*5;
M = 15*5;
```

```
19
20 % per frame true anomaly increment
21 dth = Dth / (N-1);
22
23 slo = 0*pi/180;    % starting position longitude
24 sla = 0*pi/180;    % starting position latitude
25
26 % angle defining move direction.
27 move_angle = 50*pi/180; % 0 deg means southward, 90 deg means eastwards.
28
29 % preallocating position and orientation timeseries
30 poses = zeros(3, N);
31 viewDirs = zeros(3, N);
32 sideDirs = zeros(3, N);
33
34 % ECI Position
35 p = [cos(slo)*cos(sla); sin(slo)*cos(sla); sin(sla)];
36 % ECI move direction
37 v = [-sin(slo)*sin(move_angle) + cos(slo)*sin(sla)*cos(move_angle);
38       cos(slo)*sin(move_angle) + sin(slo)*sin(sla)*cos(move_angle);
39      -cos(sla)*cos(move_angle) ];
40 % Vector normal to orbit plane
41 n = cross(p, v);
42
43 % Calculating positions during sampling and setting orientation to orbit frame
44 for i = 1:N
45     poses(:, i) = rotateV(p, n, dth*(i-1));
46     viewDirs(:, i) = -poses(:, i);
47     sideDirs(:, i) = cross(viewDirs(:,i), n);
48 end
49 poses = poses*a;
50
51 % then calculating orientation timeseries as perutbartion to the orbit frame
52 amp = 0;
53 angles = -amp*pi/180 * cos(0:pi/(N-1):pi);
54 axs = n;
55 for i = 1:N
56     viewDirs(:,i) = rotateV(viewDirs(:,i), axs, angles(i));
57     sideDirs(:,i) = rotateV(sideDirs(:,i), axs, angles(i));
58 end
59
60 for i = 1:N
61     axs = viewDirs(:,i);
```

```
    viewDirs(:,i) = rotateV(viewDirs(:,i), axs, 100*pi/180*(i-N/2)/N);
    sideDirs(:,i) = rotateV(sideDirs(:,i), axs, 100*pi/180*(i-N/2)/N);
end


%% Simulating datacube capture
% Output of this section is one simulated image taken from the satellite
% as is moves plus corresponding position and attitude info from the
% previeous script section

% ground truth image dimensions
tmep = size(im(:,:,1));
img_px = tmep(2); % horizontal image axis
img_py = tmep(1); % vertical image axis
clear tmep;

% allocate sampled image
ims = uint8(zeros(N, M, 3));

% allocate array of sample point ground positions
samplePoints = zeros(N, M, 2);

toc
disp('starting to sample an image ...');

for i = 1:N

    pos = poses(:,i);
    viewDir = viewDirs(:,i);
    sideDir = sideDirs(:,i);

    for j = 1:M

        th = atan(2*tan(fov/2 * ((j-1)/(M-1) - 0.5)));
        % pixel view direction unit vector
        vi = rotateV(viewDir, sideDir, th);

        groundPos = projection(pos, vi);

        e2 = 2*f - f*f; % eccentricity squared

        % geodetic coordinates
        long = atan2(groundPos(2), groundPos(1));
```

```
105         lat = atan2(groundPos(3), (1-e2)*sqrt(groundPos(1)*groundPos(1) +
    groundPos(2)*groundPos(2)));
106
107         % transforming coordinates to ground truth image space
108         groundPosX = 34000*lat + 4800;
109         groundPosY = 34000*long + 2200;
110
111         samplePoints(i, j, 1) = groundPosX;
112         samplePoints(i, j, 2) = groundPosY;
113
114
115         samplingMethod = 2; % 1: nearest neighbor 2: bilinear
116         if samplingMethod == 1
117         % nearest neighbor sampling of cube
118             is = round(groundPosX);
119             js = round(groundPosY);
120
121             % constrain pixels to within the image
122             if is < 1
123                 is = 1;
124             elseif is > img_px
125                 is = img_px;
126             end
127
128             if js < 1
129                 js = 1;
130             elseif js > img_py
131                 js = img_py;
132             end
133
134             % sample
135             ims(i,j,:) = im(js, is, :);
136         else
137         % bilinear interpolation sampling of cube
138
139             % constrain pixels to within the image
140             if groundPosY < 1
141                 groundPosY = 1;
142             elseif groundPosY > img_py
143                 groundPosY = img_py;
144             end
145
146             if groundPosX < 1
```

```
                groundPosX = 1;
            elseif groundPosX > img_px
                groundPosX = img_px;
            end

            % interpolate/sample
            i_l = floor(groundPosX);
            i_u = ceil(groundPosX);

            j_l = floor(groundPosY);
            j_u = ceil(groundPosY);

            frac_i = groundPosX - i_l;
            frac_j = groundPosY - j_l;

            temp1 = (1 - frac_i)*im(j_l, i_l, :) + frac_i*im(j_l, i_u, :);
            temp2 = (1 - frac_i)*im(j_u, i_l, :) + frac_i*im(j_u, i_u, :);

            ims(i,j,:) = (1 - frac_j)*temp1 + frac_j*temp2;
        end
    end
end

disp('Done sampling an image.');
toc
```

Listing 6.5: Script which projects pixels onto earth and finds resample grid

```
%% Calculations
% 1. Giving each pixel groud coordinates
% 2. finding ground coordniate boundaries
% 3. defining registrations/resample grid

%% 1. and 2. Giving each pixel groud coordinates

disp('Geocoding pixels ...');

sampledData.pixelPoses = zeros(N,M,2);
sampledData.pixelColors = ims;

minx = inf;
maxx = -inf;
```

```
15
16 miny = inf;
17 maxy = -inf;
18
19 pointWithLeastX = [-1 -1];
20 pointWithLeastY = [-1 -1];
21 pointWithLargestX = [-1 -1];
22 pointWithLargestY = [-1 -1];
23
24 % projecting pixels onto ground /
25 % / finding pixels corresponding group position
26 % and putting the positions into data.pixelPoses
27 for i = 1:N
28
29     pos = poses(:,i);
30     viewDir = viewDirs(:,i);
31     sideDir = sideDirs(:,i);
32
33     for j = 1:M
34
35         th = atan(2*tan(fov/2 * ((j-1)/(M-1) - 0.5)));
36
37         %v = rotateV(viewDir, sideDir, -fov/2 + (j - 1)*fov/im_size_j);
38         v = rotateV(viewDir, sideDir, th);
39
40
41         groundPos = projection(pos, v);
42
43         e2 = 2*f - f*f; % eccentricity squared
44
45         % geodetic coordinates
46         long = atan2(groundPos(2), groundPos(1));
47         lat = atan2(groundPos(3), (1-e2)*sqrt(groundPos(1)*groundPos(1) +
    groundPos(2)*groundPos(2)));
48
49
50         % coordinates to image space function, part of the simulation, not
51         % part of the algorithm
52         groundPosX = 35000*long +500;
53         groundPosY = 35000*lat + 2000;
54
55         % ground truth space pixel poses of sample points
56         sampledData.pixelPoses(i,j,1) = groundPosX; % x
```

```
        sampledData.pixelPoses(i,j,2) = groundPosY; % y

        if sampledData.pixelPoses(i,j,1) < minx
            minx = sampledData.pixelPoses(i,j,1);
            pointWithLeastX(1) = sampledData.pixelPoses(i,j,1);
            pointWithLeastX(2) = sampledData.pixelPoses(i,j,2);
        end
        if sampledData.pixelPoses(i,j,1) > maxx
            maxx = sampledData.pixelPoses(i,j,1);
            pointWithLargestX(1) = sampledData.pixelPoses(i,j,1);
            pointWithLargestX(2) = sampledData.pixelPoses(i,j,2);
        end
        if sampledData.pixelPoses(i,j,2) < miny
            miny = sampledData.pixelPoses(i,j,2);
            pointWithLeastY(1) = sampledData.pixelPoses(i,j,1);
            pointWithLeastY(2) = sampledData.pixelPoses(i,j,2);
        end
        if sampledData.pixelPoses(i,j,2) > maxy
            maxy = sampledData.pixelPoses(i,j,2);
            pointWithLargestY(1) = sampledData.pixelPoses(i,j,1);
            pointWithLargestY(2) = sampledData.pixelPoses(i,j,2);
        end
    end
end

disp('Done Geocoding pixels.'); toc

%% 3. Defining registration grid

disp('Defining registration grid ...');

spanX = maxx-minx;
spanY = maxy-miny;

resolutionIncreaseFactor = 1.5^2;

aspectRatio = spanX/spanY;
pixelCount = N*M*resolutionIncreaseFactor;

K = floor(sqrt(aspectRatio*pixelCount));
L = floor(pixelCount/K);

registrationGridPositions = zeros(K,L,2);
```

```
100
101 test1 = ones(1,K)*minx + (0:(spanX/(K-1)):(spanX));
102 for i = 1:L
103     test2 = (miny + (i-1)*spanY/(L-1))*ones(1,K);
104
105     % ground truth space pixel poses of registration grid
106     registrationGridPositions(:,i,1) = test1;
107     registrationGridPositions(:,i,2) = test2;
108 end
109
110 disp('Done defining registration grid.'); toc
```

Listing 6.6: Script which determines inverse projection function inverse projects and resamples

```
1  % Putting all 2D pixel positions into one large 1D array, to put it into
2  % the right format for polynomial fitting using the function fit()
3
4  disp('Creating 1D array of pixel coordinates ...');
5
6  pixelPoses1D = zeros(N*M, 2);
7  y1 = zeros(N*M, 1);
8  y2 = zeros(N*M, 1);
9
10 for i = 1:N
11     for j = 1:M
12         % ground truth space pixel positions of sample points
13         pixelPoses1D((i-1)*M + j, 1) = sampledData.pixelPoses(i,j,1);
14         pixelPoses1D((i-1)*M + j, 2) = sampledData.pixelPoses(i,j,2);
15
16         % image space pixel positions
17         y1((i-1)*M + j) = i;
18         y2((i-1)*M + j) = j;
19     end
20 end
21
22 disp('Done creating 1D array of pixel coordinates.');
23 toc
24
25 % letting matlab do the fitting
26 disp('Starting to do the fitting ...');
27 surffit_1 = fit(pixelPoses1D, y1, 'poly55');
```

```
disp('Fitting 1 done');
surffit_2 = fit(pixelPoses1D, y2, 'poly55');
disp('Fitting 2 done');
toc

%% Inverse projecting the grid - Parallel with 4 threads (on my laptop)

registeredData.pixelPoses = zeros(K,L,2);

pPosesX = zeros(K,L);
pPosesY = zeros(K,L);

n = length(coeffnames(surffit_1));
coeffs1 = coeffvalues(surffit_1);
coeffs2 = coeffvalues(surffit_2);

% Finding image spae coordinates
disp('Backprojecting ...');
parfor i = 1:K

    if mod(i,16) == 0
        fprintf('%d ', i);
        if mod(i,16*16) == 0
            fprintf('\n');
        end
    end

    v1 = zeros(1, L);
    v2 = zeros(1, L);

    for j = 1:L

        %/*

        lm = 0;
        l = 0;
        sum1 = 0;
        sum2 = 0;
        x = registrationGridPositions(i,j,1);
        y = registrationGridPositions(i,j,2);

        for k = 1:n

```

```
71              sum1 = sum1 + coeffs1(k)*y^l*x^(lm-l);
72              sum2 = sum2 + coeffs2(k)*y^l*x^(lm-l);
73
74
75              if l == lm
76                  lm = lm + 1;
77                  l = 0;
78              else
79                  l = l + 1;
80              end
81          end
82
83          v1(j) = sum1;
84          v2(j) = sum2;
85
86      end
87      pPosesX(i,:) = v1;
88      pPosesY(i,:) = v2;
89  end
90
91  registeredData.pixelColors = uint8(zeros(K,L,3));
92
93  pColorsR = uint8(zeros(K,L));
94  pColorsG = uint8(zeros(K,L));
95  pColorsB = uint8(zeros(K,L));
96
97  % Resampling the raw image
98  parfor i = 1:K
99
100     p1 = uint8(zeros(1, L));
101     p2 = uint8(zeros(1, L));
102     p3 = uint8(zeros(1, L));
103
104     for j = 1:L
105
106         pixelPosX = pPosesX(i,j);
107         pixelPosY = pPosesY(i,j);
108
109         if pixelPosX < 1 || pixelPosX > N  ||  pixelPosY < 1 || pixelPosY > M
110             p1(j) = 0;
111             p2(j) = 0;
112             p3(j) = 0;
113         else
```

```
            i_l = floor(pixelPosX);
            j_l = floor(pixelPosY);
            i_u = ceil(pixelPosX);
            j_u = ceil(pixelPosY);

            frac_i = pixelPosX - i_l;
            frac_j = pixelPosY - j_l;

            temp1 = (1 - frac_i)*ims(i_l, j_l, :) + frac_i*ims(i_u, j_l, :);
            temp2 = (1 - frac_i)*ims(i_l, j_u, :) + frac_i*ims(i_u, j_u, :);

            temp3 = (1 - frac_j)*temp1 + frac_j*temp2;

            p1(j) = temp3(1);
            p2(j) = temp3(2);
            p3(j) = temp3(3);
        end
    end

    pColorsR(i, :) = p1;
    pColorsG(i, :) = p2;
    pColorsB(i, :) = p3;
end

registeredData.pixelColors(:,:,1) = pColorsR;
registeredData.pixelColors(:,:,2) = pColorsG;
registeredData.pixelColors(:,:,3) = pColorsB;

fprintf('\n');

registeredData.pixelPoses(:,:,1) = pPosesX;
registeredData.pixelPoses(:,:,2) = pPosesY;

disp('Done backprojecting ...');
toc
```

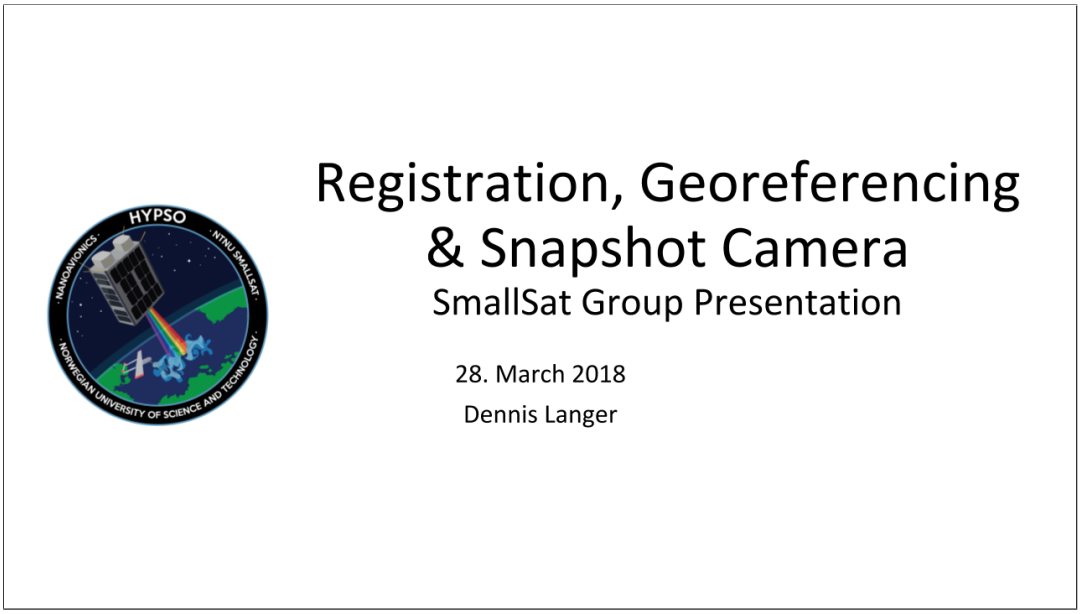
Listing 6.7: Script to generate the plots shown in the report

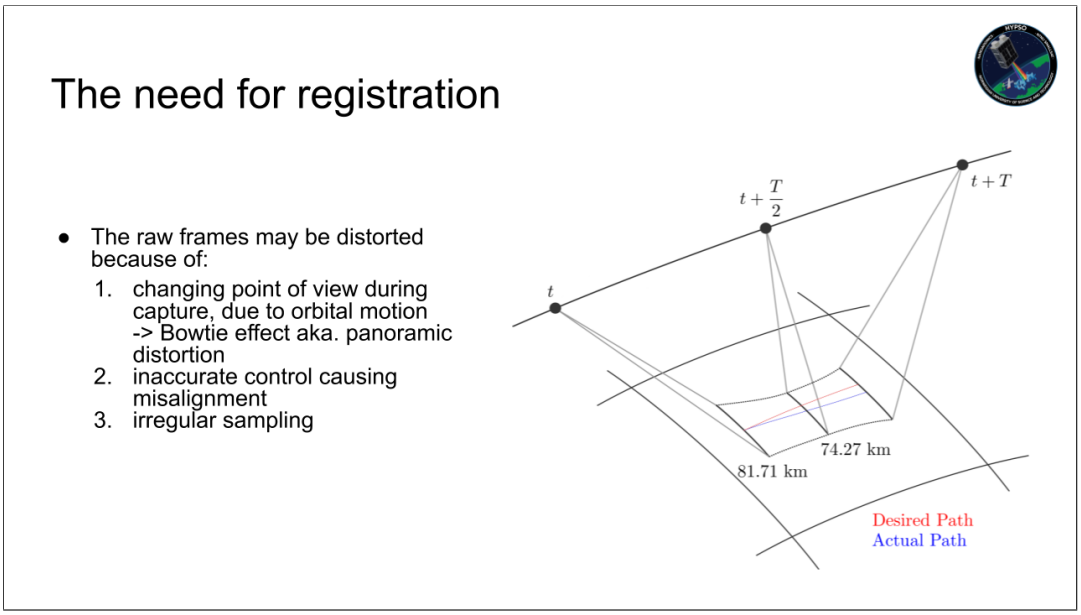
```
%% Plotting sampled image

figure;
imshow(ims);
```

```
% saving sampled image
% imwrite(ims, 'sampled.png');


%% Plotting ground sample points

figure; grid on;
imshow(im); hold on;
plot(samplePoints(:, :, 1)', samplePoints(:, :, 2)', 'o','Color',[0,0,0],
    'MarkerFaceColor', 'w');
axis equal;
title('Ground Space');


%% Plotting ground space grid with ground sample points

figure; hold on; box on; grid on;
title(['Image ' 'Projected']);

xlabel('Longitude'); ylabel('Latitude'); zlabel('z');

boundary = [minx maxx maxx minx minx;
            miny miny maxy maxy miny;
               0    0    0    0    0 ];

plot3(boundary(2,:), maxx-boundary(1,:), boundary(3,:));

for i = 1:L
    plot3(registrationGridPositions(:,i,2), maxx-registrationGridPositions(:,i,1),
    zeros(1,K), 'x', 'color', [0.5, 0.5, 0.9]);
end
for i = 1:M
    plot3(sampledData.pixelPoses(:,i,2), maxx-sampledData.pixelPoses(:,i,1),
    zeros(1,N), 'ro');
end

axis equal;
title('Ground Space');
set(gca, 'xticklabel', []);
set(gca, 'yticklabel', []);


%% Plotting Image space resample grid positions
```

```
45
46 figure; hold on; grid on; box on;
47
48 for j = 1:L
49     plot3(registeredData.pixelPoses(:,j,2), -registeredData.pixelPoses(:,j,1),
       zeros(1,K), 'x', 'color', [0.5, 0.5, 0.9]);
50 end
51 for j = 1:N
52     plot(0.5:(0.499999+M), -(j-0.5)*ones(1,M), 'ro');
53 end
54 plot([0; 0; M; M; 0], -[0; N; N; 0; 0], 'r');
55
56 axis equal;
57 title('Image Space');
58 set(gca,'xticklabel',[])
59 set(gca,'yticklabel',[])
60
61
62 %% Plotting Registered image
63
64 figure;
65 imshow(registeredData.pixelColors);
66 % save registered image
67 % imwrite(registeredData.pixelColors, 'registered.png');
```

B SmallSat Group Presentation





Registration

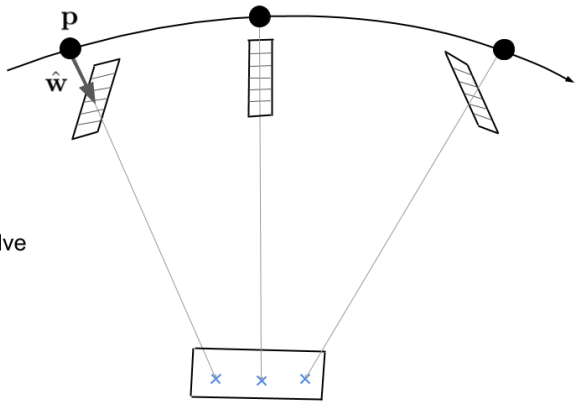
- Place each pixel onto its position
  - by drawing a ray along the view vector:

$$\mathbf{p} + t\hat{\mathbf{w}}\ ,\ t > 0$$

- And finding its intersection with earth ellipsoid

$$\frac{x^2 + y^2}{a^2} + \frac{z^2}{b^2} = 1$$

- Gives second order polynomial to solve for t
- Pixels are Geocoded/Georeferenced



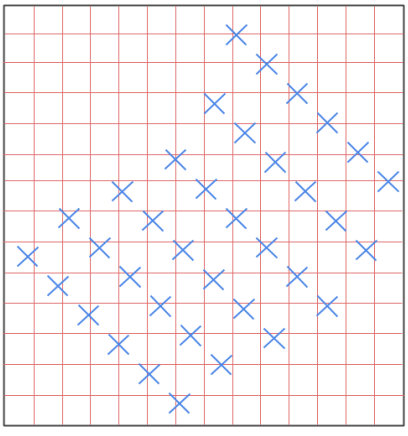
Resampling 1

Blue crosses:
earth projected image pixels

- Find smallest rectangle bounding the blue crosses
- Overlay uniform grid inside of rectangle
  - Can differ in resolution than original image

Red grid points:
Cartographic map pixels

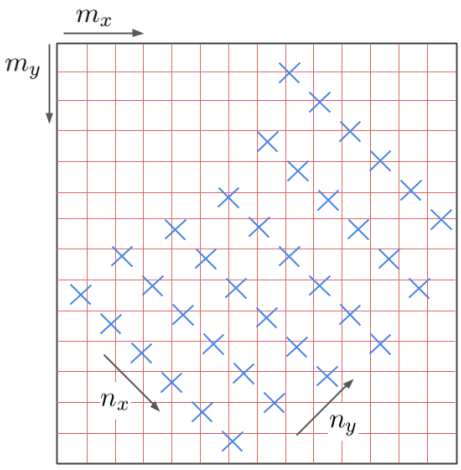
- Next: Determine color of each red grid point



Resampling 2

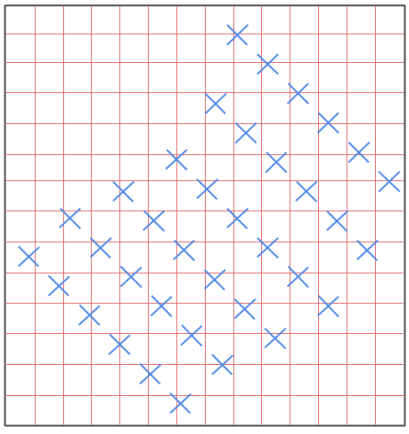
- For each map grid point, need to figure out how much each image pixel contributes to color.
- Distance determines contribution
- For a given grid point, what are the closest image pixels?
  => Need to calculate how close ***each*** blue cross is to ***each*** grid point
  => Computational Cost: HIGH

$$O(n_x n_y m_x m_y)$$



Resampling 3

- The inverse problem is much easier:
  - For a given image pixel, what are the closest map grid points?
    - can be found by rescaling and then taking floor() and ceil()
- It's easier because the map grid is uniform
- Formulate the problem as the inverse problem by first transforming into image space, where the **pixels are uniformly** spaced, and the **grid points are irregular**



Resampling 4 - Inverse transform

- The inverse mapping function is found by fitting a 2D polynomial model to the data pair:
  (pixel coordinates, pixel index)
- Minimizing square error

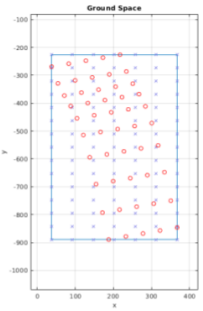
$$m = f_m^{-1}(\lambda, \phi) = \sum_{i=0}^{k}\sum_{j=0}^{k} a_{i,j}\lambda^i\phi^j$$

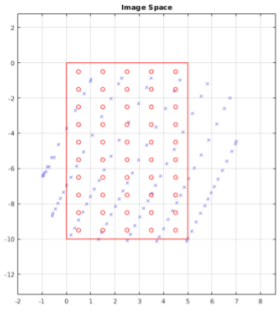
$$n = f_n^{-1}(\lambda, \phi) = \sum_{i=0}^{k}\sum_{j=0}^{k} b_{i,j}\lambda^i\phi^j$$

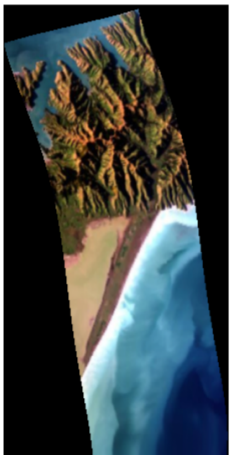
$$\min_{\mathbf{a}\in\mathbb{R}^l} \sum_{pixels} ||f_m^{-1}(\lambda_{pixel}, \phi_{pixel}, \mathbf{a}) - m_{pixel}||$$

$$\min_{\mathbf{b}\in\mathbb{R}^l} \sum_{pixels} ||f_n^{-1}(\lambda_{pixel}, \phi_{pixel}, \mathbf{b}) - n_{pixel}||$$

Example

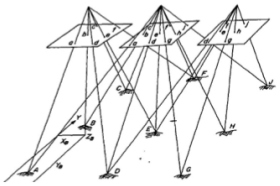






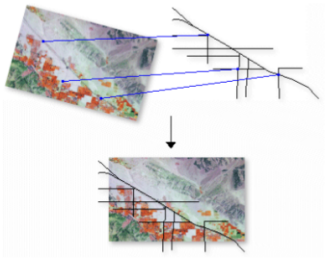
Georeferencing: Optical vs. Geometric

- Geometric georeferencing is already done by registration



F. H. Moffitt, E. M. Mikhail, *Photogrammetry*, 1980

- Optical Georeferencing requires database of coastlines, rgb image containing coastlines and processing

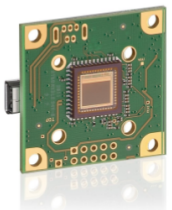


http://resources.esri.com/help/9.3/arcgisengine/java/gp_toolref/data_management_toolbox/changing_the_geographic_coordinates_of_a_raster_dataset_colon_georeferencing.htm

Snapshot Camera

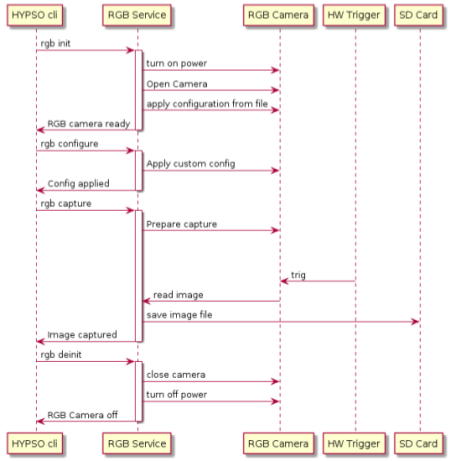
- Originally considered for:
  - *Superresolution* by *panchromatic sharpening*, or
  - *Georeferencing*
- Now mainly considered for:
  - *verification* of attitude
  - *imaging capability* in case HSI breaks

IDS UI-1250LE - Sensor: EV76C570



IDS UI-1252LE - Sensor: EV76C570

Snapshot Camera - Command interface



Demo

C Full table of relevant Success criteria and Requirements

ID	Requirement
MS-0-002	Shall observe Case 1 and Case 2 waters off the Norwegian of at least 70x70 km^2 area
MS-0-003	Should observe Case 1 and Case 2 waters globally of at least 70x70 km^2 area
MS-0-007	HSI images shall have at least 300 m spatial resolution
MS-0-008	HSI images should have at least 100 m spatial resolution
MS-0-009	S/C shall perform along-track slew maneuver at a angular velocity with magnitude of 0.01 deg stability over 60 s
MS-0-010	S/C should perform cross-track slew maneuver at a angular velocity with magnitude of 0.01 deg stability over 60 s
M-1-006	Shall have mapping knowledge error of less than 100 m (RMS)
M-1-011	HSI images shall be geometrically corrected, geo-referenced and validated against a reference with predefined geometric control points and latitude and longitude coordinates
M-2-004	Should perform image acquisition of target areas where there is highest probability of detection off the coast of Norway with off-Nadir pointing capability
M-2-005	Should use wide-FoV monochrome images as grid and reference for HSI images with known geometrical features and coordinates (lat & lon)
M-2-009	HSI image resolution shall be enabled to be less than 100 m through onboard or ground super-resolution/ deconvolution
M-2-010	Should enable onboard image registration, motion-blur correction, geo-referencing, super-resolution and other fancy algorithms
M-2-017	S/C shall cover 430 km orbit-track, image acquisition shall be at least 57 s during slew maneuver, start at 20 deg view angle and end at -20 deg view angle
M-2-018	L1A data shall have no more than 2200 frames and be less than 420 MB
M-2-020	Data processing time after shall be less than 70 s for L1A data
M-2-021	Data processing time after should be less than 120 s for operational data
SBUS-3-009	Orbit knowledge should be <10 m (2 sigma) with GPS
SBUS-3-010	PPS-signal from GNSS or other solution for time-synchronization as well as navigational (attitude and orbit) data from ADCS or OBC shall be interfaced to payload.
SBUS-3-011	ADCS shall have absolute pointing knowledge of 360 arcsec/0.1 deg (2 sigma) and absolute pointing accuracy of 3600 arcsec/1 deg (2 sigma).

Table 6.1: Relevant HYPSO mission success criteria and requirements

D Musings into line-spheroid intersections

Given a spheroid described by semi major axis x and eccentricity e

$$x^2 + y^2 + \frac{z^2}{1-e^2} = a^2 \tag{6.1}$$

and given a point p and a unit vector v that describe a line in 3D Cartesian space:

$$p + tv \ , \quad t \in \mathbb{R}$$

The values of t that give the points of intersection of the line with the spheroid are found by solving

$$\left(v_x^2 + v_y^2 + \frac{v_z^2}{1-e^2}\right) t^2 + 2\left(p_x v_x + p_y v_y + \frac{p_z v_z}{1-e^2}\right) t + p_x^2 + p_y^2 + \frac{p_z^2}{1-e^2} - a^2 = 0$$

The discriminant

$$\Delta = \frac{b^2}{4} - ac = a^2\left(v_x^2 + v_y^2 + \frac{v_z^2}{1-e^2}\right) - (p_x v_y - p_y v_x)^2 - \frac{(p_x v_z - p_z v_x)^2}{1-e^2} - \frac{(p_y v_z - p_z v_y)^2}{1-e^2}$$

characterizes the kind of intersection there is. The possibilities are

- No intersection, $\Delta < 0$
- One intersection, $\Delta = 0$
- Two intersections, $\Delta > 0$

let $e = 0$, that is, let the spheroid become a sphere, then a describes the sphere's radius and

$$\begin{aligned}\Delta &= a^2\left(v_x^2 + v_y^2 + v_z^2\right) - (p_x v_y - p_y v_x)^2 - (p_x v_z - p_z v_x)^2 - (p_y v_z - p_z v_y)^2 \\ &= a^2||v||^2 - p \times v \\ &= a^2||v||^2 - ||p||^2||v||^2 \sin\theta\end{aligned}$$

and since v is a unit vector

$$\Delta = a^2 - ||p||^2 \sin^2\theta \tag{6.2}$$

From this the following can be seen

1. if $||p|| < a$ then $\Delta > 0$.
   $||p|| < a$ means that there is a point on the line that is inside the sphere. Thus there are always two intersections.

2. if $\theta = 0$, then $\Delta > 0$.
   $\theta = 0$ means p and v are parallel and that the line goes through $(0, 0, 0)$. Since the sphere is centered around $(0, 0, 0)$, there are two intersections.

3. if $\theta = \frac{\pi}{2}$ and $||p|| > a$, then $\Delta < 0$.
   $\theta = \frac{\pi}{2}$ means that the line is perpendicular to the position vector p. If $||p|| = a$, then the line is a tangent. If $||p|| > a$, then the line is a radially translated tangent which implies that it does not intersect the circle.

Rearrange Δ for the spheroid:

$$\Delta = a^2 \left(v_x^2 + v_y^2 + \frac{v_z^2}{1 - e^2} \right) - (p_x v_y - p_y v_x)^2 - \frac{(p_x v_z - p_z v_x)^2}{1 - e^2} - \frac{(p_y v_z - p_z v_y)^2}{1 - e^2}$$

$$\begin{aligned} \Delta = a^2 \left(v_x^2 + v_y^2 + \frac{v_z}{\sqrt{1 - e^2}} \frac{v_z}{\sqrt{1 - e^2}} \right) - (p_x v_y - p_y v_x)^2 \\ - \left(p_x \frac{v_z}{\sqrt{1 - e^2}} - \frac{p_z}{\sqrt{1 - e^2}} v_x \right)^2 - \left(p_y \frac{v_z}{\sqrt{1 - e^2}} - \frac{p_z}{\sqrt{1 - e^2}} v_y \right)^2 \end{aligned}$$

Replace the z-component of v and p with $v_z\sqrt{1 - e^2}$ and $p_z\sqrt{1 - e^2}$ respectively to obtain

$$\begin{aligned} \Delta = a^2 \left(v_x^2 + v_y^2 + \frac{v_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} \frac{v_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} \right) - (p_x v_y - p_y v_x)^2 \\ - \left(p_x \frac{v_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} - \frac{p_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} v_x \right)^2 - \left(p_y \frac{v_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} - \frac{p_z\sqrt{1 - e^2}}{\sqrt{1 - e^2}} v_y \right)^2 \end{aligned}$$

$$\begin{aligned} \Delta &= a^2 \left(v_x^2 + v_y^2 + v_z^2 \right) - (p_x v_y - p_y v_x)^2 - (p_x v_z - p_z v_x)^2 - (p_y v_z - p_z v_y)^2 \\ &= a^2 ||v'||^2 - p' \times v' \\ &= a^2 ||v'||^2 - ||p'||^2 ||v'||^2 \sin\theta \end{aligned}$$

where $p' = \left[p_x, p_y, p_z\sqrt{1 - e^2} \right]^\top$ and $v' = \left[v_x, v_y, v_z\sqrt{1 - e^2} \right]^\top$. Now the expression resembles the expression for a sphere, (6.2).

Bibliography

[1] J. L. Garrett, D. Langer, K. Avagian, and A. Stahl, Eds., *Accuracy of super-resolution for hyperspectral ocean observations*, 2019.

[2] *Nano Avionics*, website: https://n-avionics.com/.

[3] *CubeSat Design Specification*, The CubeSat Program, California Polytechnic State University, 2014, https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf.

[4] *6U CubeSat Design Specification*, The CubeSat Program, California Polytechnic State University, 2018, https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/5b75dfcd70a6adbee5908fd9/1534451664215/6U_CDS_2018-06-07_rev_1.0.pdf.

[5] E. W. Mowle and C. J. Dennehy, "The landsat-6 satellite: an overview," in *NTC '91 - National Telesystems Conference Proceedings*, March 1991, pp. 277–282, https://doi.org/10.1109/NTC.1991.148030.

[6] S. Sterckx, I. Benhadj, G. Duhoux, S. Livens, W. Dierckx, E. Goor, S. Adriaensen, W. Heyns, K. V. Hoof, G. Strackx, K. Nackaerts, IlsReusen, T. V. Achteren, J. Dries, T. V. Roey, K. Mellab, R. Duca, and J. Zender, "The proba-v mission: image processing and calibration," *International Journal of Remote Sensing*, vol. 35, no. 7, pp. 2565–2588, 2014, http://dx.doi.org/10.1080/01431161.2014.883094.

[7] S. Leprince, S. Barbot, F. Ayoub, and J. Avouac, "Automatic and precise orthorectification, coregistration, and subpixel correlation of satellite images, application to ground deformation measurements," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 6, pp. 1529–1558, June 2007, https://doi.org/10.1109/TGRS.2006.888937.

[8] P. R. Hakim, A. P. S. Jayani, A. Sarah, and W. Hasbi, "Autonomous image georeferencing based on database image matching," in *2018 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, Sep. 2018, pp. 1–6, https://doi.org/10.1109/ICARES.2018.8547086.

[9] H. Sui, Z. Song, D. Gao, and L. Hua, "Automatic image registration based on shape features and multi-scale image segmentation," in *2017 2nd International Conference*

on Multimedia and Image Processing (ICMIP), March 2017, pp. 118–122, https://doi.org/10.1109/10.1109/ICMIP.2017.17.

[10] L. M. G. Fonseca and B. S. Manjunath, "Registration techniques for multisensor remotely sensed images," *Photogrammetric Engineering and Remote Sensing*, vol. 62, no. 9, pp. 1049–1056, 1996.

[11] C. Tao, Z. Zou, and H. Sun, "High-resolution satellite image registration using local feature and contour fragment," in *2012 IEEE International Geoscience and Remote Sensing Symposium*, July 2012, pp. 2344–2347.

[12] R. Szeliski, *Image Alignment and Stitching: A Tutorial.* now, 2006, https://ieeexplore.ieee.org/document/8187154.

[13] P. R. Hakim, A. P. S. Jayani, A. Sarah, and W. Hasbi, "Autonomous image georeferencing based on database image matching," in *2018 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, Sep. 2018, pp. 1–6.

[14] S. Riazanoff and J.-P. Gleyzes, "Spot123-4-5 geometry handbook," GAEL Consultant, Tech. Rep., 2004.

[15] R. H. Stewart, *Introduction to Physical Oceanography.* Department of Oceanography, Texas A & M University, 2008, https://ocean.tamu.edu/academics/resources/ocean-world/resources/Stewart_PObook.pdf.

[16] *International Terrestial Reference Frame*, website: http://itrf.ensg.ign.fr/ Ellipsoid recommendaion: http://itrf.ensg.ign.fr/faq.php?type=answer#question2.

[17] P. Osborne, *The Mercator Projections*, Jan 2013, https://doi.org/10.5281/zenodo.35392.

[18] J. Meeus, *Astronomical Algorithms.* Willmann-Bell, Inc, 1998, vol. 2.

[19] R. D. Harriss, A. D. Holland, S. J. Barber, S. Karout, R. Burgon, B. J. Dryer, N. J. Murray, D. J. Hall, P. H. Smith, T. Grieg, J. H. Tutt, J. Endicott, P. Jerram, D. Morris, M. Robbins, V. Prevost, and K. Holland, "Compact cmos camera demonstrator (c3d) for ukube-1," proc. SPIE 8146, UV/Optical/IR Space Telescopes and Instruments: Innovative Technologies and Concepts V, 81460U (27 September 2011) https://doi.org/10.1117/12.895986.

D. Langer

Image registration and georeferencing with snapshot camera for the HYPSO mission

NTNU
Norwegian University of
Science and Technology