Master's thesis

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

Andreas Horpedal Bugge

UAV-Based Hyperspectral Stereoscopic Imaging for Accurate 3D Terrain Mapping

Master's thesis in Elektronisk systemdesign og innovasjon (MTELSYS)
Supervisor: Lise Lyngsnes Randeberg
Co-supervisor: Trond Løke
June 2022

NTNU
Norwegian University of Science and Technology

Andreas Horpedal Bugge

UAV-Based Hyperspectral Stereoscopic Imaging for Accurate 3D Terrain Mapping

Master's thesis in Elektronisk systemdesign og innovasjon (MTELSYS)
Supervisor: Lise Lyngsnes Randeberg
Co-supervisor: Trond Løke
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

Hyperspectral imaging adds a new dimension to the well-tested and more traditional imaging technique of stereoscopic imaging. The presented work explores how the complementary imaging techniques of hyperspectral imaging and stereoscopic imaging can be combined with UAV technology for 3D terrain mapping of an imaged scene, from UAV altitudes of 20 m, 40 m, and 60 m above ground. This includes uncovering both limitations and possibilities and how this approach to 3D terrain mapping holds up against more conventional methods based on LiDAR technology.

The imaging system employed, Hyspex Mjolnir VS-620, is not specifically tailored for stereoscopic applications and comes with a few unconventional characteristics. Particularly the narrow baseline of only 75 mm puts a limit on the achievable accuracy and makes the system highly dependent on precise stereo matching well beneath sup-pixel resolution. On the other hand, the hyperspectral aspects of the imaging system make for a more robust model, while the imaging technique of pushbroom scanning allows for an easy and intuitive scheme for precise georeferencing.

Phase-based stereo matching, based on deriving Fourier-phase images from the stereo pair allows for precise stereo matching, while at the same time being robust against noise and radiometric differences. The two-step phase-based stereo matching algorithm developed in the presented work, achieved pixel accuracy down to 0.02 pixels when tested on synthetic data, outperforming the off-shelf intensity-based SGBM algorithm provided by the Open Source Computer Vision Library significantly.

On real hyperspectral data of the imaged scene, the stereo matching accuracy decreased somewhat and errors in the range of 0.05-0.15 pixels were reported. The corresponding elevation accuracy of the derived point clouds was calculated to be 0.4096 m, 1.2049 m, and 2.4918 m for UAV flight altitudes of 20 m, 40 m, and 60 m respectively. Despite accuracy comparable to LiDAR technology not being achieved, the results demonstrate an encouraging potential for applications with fewer constraints concerning precision, particularly evident from the point clouds derived from UAV altitudes of 20 m and 40 m above ground.

Sammendrag

Hyperspektral avbildning gir en ny dimensjon til den velutprøvde og mer tradisjonelle avbildningsteknikken stereoskopisk avbildning. Det presenterte arbeidet utforsker hvordan de komplementære avbildningsteknikkene hyperspektral avbildning og stereoskopisk avbildning kan kombineres med UAV-teknologi for 3D terrengkartlegging av en avbildet scene, fra UAV flyhøyder på 20 m, 40 m og 60 m over bakken. Dette inkluderer å avdekke både begrensninger og muligheter og hvordan denne tilnærmingen til 3D-terrengkartlegging presterer opp mot mer konvensjonelle metoder basert på LiDAR-teknologi.

Bildesystemet som brukes, Hyspex Mjolnir VS-620, er ikke skreddersydd for stereoskopiske applikasjoner og kommer med noen ukonvensjonelle egenskaper. Spesielt den smale avstanden mellom kameraene på bare 75 mm setter en begrensning på oppnåelig nøyaktighet, og gjør systemet avhengig av å detektere samsvarende piksler i bildene svært nøyaktig. På den andre siden gir de hyperspektrale aspektene ved avbildningssystemet en mer robust modell, mens avbildningsteknikken pushbroom-skanning tillater en enkel og intuitiv metode for presis georeferering.

Fasebaserte metoder, basert på å utlede Fourier-fasebilder fra stereoparene, gjør det mulig å finne samsvarende piksler i stereoparene svært nøyaktig i tillegg til å være robust mot støy og radiometriske forskjeller. Den to-trinns fasebaserte algoritmen utviklet oppnådde pikselnøyaktighet ned til 0.02 piksler på syntetisk data, og utkonkurrerte den intensitetbaserte SGBM algoritmen levert av Open Source Computer Vision Library betydelig.

På ekte hyperspektral data falt pikselnøyaktigheten noe, og feil i området 0.05-0.15 piksler ble rapportert. Dette tilsvarer en høydenøyaktighet på 0.4096 m, 1.2049 m og 2.4918 m på de beregnede punktskyene fra UAV flyhøydene 20 m, 40 m og 60 m. Til tross for at nøyaktigheten ikke kan sammenlignes med presisjon fra LiDAR-teknologi, viser resultatene et potensiale for applikasjonen med mindre krav til presisjon, spesielt tydelig fra punktskyene beregnet fra UAV flyhøydene 20 m og 40 m over bakken.

Preface

This master thesis is a continuation of the work presented in [1], a project report from the mandatory project course TFE4580 at NTNU. The project report addressed the same assignment, however, most of the work was based around implementing an open-source algorithm. The results were not the most convincing and the full potential of the system was yet to be realized. It was therefore decided to continue the work, however, with more focus on an implementation built from scratch, inspired by similar experiments from the literature. Despite not yielding the best results, knowledge and experience gathered from the project course carried on to the thesis, and a few parts from [1] have been reused. This applies in particular to figures and illustrations, parts of the introduction, the theory section, the description of the system setup as well as some of the Python scrips presented in the Appendix.

I would to like thank my supervisor Lise Lyngsnes Randeberg for excellent guidance and inspiration not only this semester but also the previous semester in relation to the project report. The same applies to Trond Løke, my co-supervisor from Norsk Elektro Optikk. Additional thanks go to everyone who has been involved from Norsk Elektro Optikk, whether it be hyperspectral imaging, processing data, or providing code samples. At last I would like to thank all my fellow students and classmates for five delightful years in Trondheim, it has truly been a pleasure.

Andreas Horpedal Bugge
June, 2022

Table of Contents

1 Introduction

1.1 Areas of Research

Hyperspectral imaging (HSI) has emerged as a highly promising imaging technique constantly being applied to new fields of research, and for the most part with encouraging results. HSI solves several issues seen for more conventional three-channel imaging including metamerism, illumination dependency [2], and is also able to provide information over a large portion of the electromagnetic spectrum. Due to its applicability and its large potential, it is often considered the "Next Generation Imaging Technology" [2]. Stereoscopic imaging is another popular imaging technique that addresses the subject of extracting three-dimensional (3D) information from two-dimensional (2D) images. Similar to HSI, stereo imaging has also seen rising popularity in recent years and has found use in vast fields of research [3].

The presented work aims to mainly explore the research areas of HSI and stereo imaging and how they can be combined to extract 3D information from a depicted scene. More specifically, how such an imaging setup, combing both HSI and stereo imaging, can be implemented on an unmanned aerial vehicle (UAV) to image from an aerial point of view for accurate 3D terrain mapping.

1.2 Background

The impressive growth of the UAV industry supplied with remarkable advances regarding sensing technology has given rise to new areas for the two technologies to be applied [4]. The combination of UAV technology with sensing technology has previously often been associated with military reconnaissance [5], however, UAVs are now viewed as a low-cost option for remote sensing. Hence, new areas combining the two technologies are now being explored to a greater extent. This includes applications such as terrain mapping, agricultural monitoring, or for environmental purposes. One example of the latter is presented in [6], where UAV technology combined with high-resolution multispectral cameras was used for monitoring water pollution. Concerning applications such as terrain mapping, the primary sensing technologies are light detection and ranging (LiDAR), synthetic aperture radar, structure from motion and stereoscopic imaging [4], with LiDAR being regarded as the "state-of-the-art" sensing technology for 3D terrain mapping reporting accuracy down to 2 cm [7]. Despite this, LiDAR technology still has some substantial drawbacks including its high cost, being highly sensitive to weather conditions, being dependent on motion for 3D mapping [4], and have traditionally been associated with huge data sets that may be difficult to interpret in real-time. Meanwhile, stereoscopic imaging can be implemented cheaply, is not necessarily dependent on motion, does not have the same weather dependence, and have a large potential for real-time applications. An example of the latter is presented in [8], where stereo imaging was implemented in self-driving cars for obstacle detection. Next, the sensing technologies relevant to the presented work and their characteristics are briefly introduced to establish a foundation for the thesis.

1.2.1 Remote Sensing Technologies

LiDAR

LiDAR technology is based on lasers releasing pulses of light projected onto the scene, while a detector measures and registers the time interval for the light to be reflected and returned. Based on the time interval, one can tell the distance to the object reflecting the light with great accuracy [3]. A common LiDAR configuration is based on a single pulsed laser, firing onto a rotating mirror sweeping a specific plane. However, more advanced configurations have been developed specifically for 3D mapping, such as the configuration presented in [9] consisting of 64 semiconductor lasers each triggered up to 20 000 times per second. This results in 1.3 million data points created each second, generating point clouds so dense street curbs and electrical wires can be identified in an urban environment at a distance of over 100 m [9]. The more recent

development of LiDAR technology with high data rates, specifically for small-scale systems, has made it an excellent option for UAV-based terrain mapping. Other than the superior accuracy, LiDAR technology enjoys the advantages of not being limited by varying light conditions as well as the ability to penetrate vegetation and uncover moderately obscured objects [7]. However, as mentioned, LiDAR technology demonstrates a significant reduction in performance in some specific weather conditions. Especially rain and fog cause unwanted scattering of the emitted laser pulses, degrading the performance, also reported under snowy and dusty conditions [3]. Yet, the largest disadvantage associated with LiDAR technology is its substantial cost, both initial and maintenance. Although the price has been declining in the past years, high-end UAV-based LiDAR technology used for terrain mapping can easily exceed 100 000$ according to Wingtra, a drone producer for mapping, survey, and the mining industry. Because of this, it is of great interest to explore to what limit other remote sensing technologies can perform before their performance starts deteriorating in comparison to LiDAR technology.

Stereoscopic Imaging

Stereo imaging recovers the depth of a scene by comparing slight differences in the location of corresponding points in images of a common scene captured with different viewing angles [3]. An example of a binocular stereo imaging setup is presented below in Figure 1.

Figure 1: Binocular stereo imaging setup. Two cameras are imaging a common scene from two different points of view, here shifted horizontally wrt each other. By determining the shift, i.e the disparity, of corresponding points in the images, the depth of the scene can be recovered. The same illustration was also used in [1].

Here, two cameras shifted horizontally with respect to (wrt) each other, images a common static scene. The depth of the scene can be recovered by determining the horizontal shift of corresponding points, from now on referred to as the disparity, in the left image wrt right image. The greater the disparity, the closer the object is to the camera. This is demonstrated in Figure 1 by comparing the disparity associated with the person located in the foreground, to the disparity associated with the sun and the mountains located in the background much further away from the cameras. Stereoscopic imaging is not limited to the field of remote sensing and computer vision, enhanced depth perception is one of several advantages of binocular vision in biology [10]. In fact, stereo imaging simply tries to mimic the human approach to perceive the world in 3D [2].

Sensing based on stereo imaging is nothing new and can be found in numerous fields of research. The stereoscope was invented as early as 1854 and commercial 3D stereo cameras became popular during the 1920s and 1950s [2] and the technology has been applied for rough distance estimation for decades [11]. More recently, NASA used stereo imaging for navigation on their exploration rovers in 2004 [12], and stereo imaging slowly becoming a greater part of our daily life with the implementation of the technology in self-driving cars [8]. However, both examples just stated highlight the areas where stereo imaging mainly has been applied, namely for on-ground vehicles under stationary conditions [4]. Consequently, to fully comprehend the technology and to better

understand its possibilities and limitations, stereo imaging should, to a greater extent, be applied where the circumstances deviate from ideal conditions. Having said that, one example combing UAV technology and stereo imaging for terrain mapping is presented in [4], where a wide baseline stereo-rig mounted on a UAV reported accuracy ranging from 56 cm to 65 cm at an altitude of 40 m. Additionally, in [13], stereo imaging was applied to render an accurate representation of the surrounding terrain of an autonomous rotorcraft for a safe landing. Thus, despite UAV-based terrain mapping not being the typical domain for the technology, it is still an area worthy of further exploration based on results revealed by previous research.

Hyperspectral Imaging

HSI incorporates the techniques of image formation and spectroscopy to capture information over large ranges of the electromagnetic spectrum. Where more conventional imaging systems capture information from a few wavelengths, typically within the visible part of the spectrum, HSI considers several hundreds of bands of wavelengths, also extending beyond the visible. Hyperspectral data forms hyperspectral cubes, sharing two spatial dimensions and one spectral. Each pixel present in a hyperspectral cube contains an electromagnetic spectrum [14]. An example of a hyperspectral cube is presented below in Figure 2.

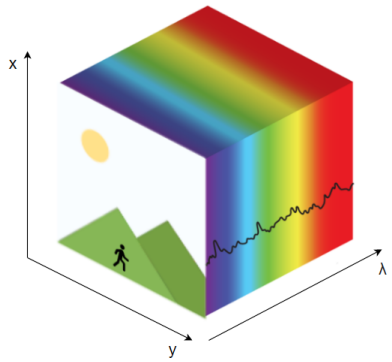


Figure 2: Hyperspectral images form hyperspectral cubes with two spatial dimensions, x and y, and one spectral dimension λ. Each pixel contains an electromagnetic spectrum. The same illustration was also used in [1].

The integrity of any HSI system is highly dependent on the spatial co-registration between bands [15]. Poor co-registration between bands of a hyperspectral cube introduces parts of the electromagnetic spectrum to each pixel from neighbouring pixels. The result is a nonphysical spectrum not corresponding to the objects in the scene. This phenomenon is typically referred to as keystone in commercial HSI systems and is the result of how the center of each pixel is shifted due to optical aberrations, as well as the point spread function's (PSF) wavelength dependency [15]. Spatial miss-alignment may be corrected by both hardware and software. However, a number of commercial HSI systems now offer implementations consisting of two separate instruments covering the visible to near-infrared (VNIR) and short-wave infrared (SWIR) part of the electromagnetic spectrum, with a common slit and fore-optics [16]. Assuming both systems are stable, triggered at the same time with equal exposure times, such configuration allows for very good co-registration between the VNIR and SWIR spectral ranges. One example of such a configuration is presented in [17], and a more detailed description of co-registration of spectral bands can be found in [16].

As already mentioned, HSI is constantly being applied to new fields with encouraging results. Its ability to collect information beyond the visible part of the electromagnetic spectrum has among other things made HSI become an emerging modality for medical applications as HSI can provide diagnostic information regarding tissue physiology, morphology, and composition [18]. With regards to stereo imaging, HSI has yet to become the new gold standard. Despite this, some research has been done and hyperspectral stereo imaging was employed in [19] for real-time

3D depth estimation in an outdoor environment reporting a mean square error of 0.0267 m^2 at a distance of 5-10 m. Nevertheless, it is easy to imagine several aspects of HSI being beneficial also for stereo imaging. Particularly collecting data outside the visible part of the electromagnetic spectrum and into the infra-red part may prove to be advantageous, as more conventional stereo imaging is highly dependent on well-illuminated scenes [2].

1.3 Motivation and Problem Description

The hardware setup utilised in the presented work, including the UAV platform, HSI system well as the LiDAR, are all provided and operated by Norsk Elektro Optikk (NEO). In terms of both airborne and ground-based HSI, NEO has become an industry-leading provider of imaging systems, recognized for their stability, flexibility, and superior data quality. With regards to the imaging system employed, Hyspex Mjolnir VS-620, the configuration consists of two separate optical systems, one VNIR system and one SWIR system, covering the spectral range of 400 - 2500 nm over 490 bands with a co-registration between bands of better than 0.2 pixels over the entire VNIR-SWIR range. The impressive co-registration over the entire spectral range is largely due to having two separate systems, VNIR and SWIR, with co-aligned optical axes and some common fore-optics, as explained in the previous section. However, with this comes an interesting property, the VNIR and SWIR systems are shifted 75 mm horizontally wrt each other, similar to the binocular setup in Figure 1. Thus, in order to construct the best co-registered hyperspectral cube, the stereoscopic aspect of the imaging setup comes as a byproduct.

It has been well established how both LiDAR technology, as well as stereo imaging, can be used to extract 3D information from a depicted scene. The question now becomes whether or not the stereoscopic byproduct of the imaging system can be exploited for 3D reconstruction of an imaged scene, despite the imaging setup not being specifically tailored for this application. For 3D terrain mapping, NEO typically employs LiDAR technology, however as stated in the previous section, LiDAR can be rather expensive. It is therefore of great interest to explore the limits to which this stereoscopic byproduct can be exploited for accurate 3D terrain mapping. Nevertheless, given that the imaging system is somewhat unconventional for this purpose, issues uncovered throughout the thesis not seen in more conventional setups will have to be addressed. On the other hand, some of these unconventional traits, such as the hyperspectral characteristics, should be viewed as opportunities rather than limitations.

The presented work aims to explore the stereoscopic byproduct of the hyperspectral imaging system, and to which degree the system can be implemented on a UAV for accurate 3D terrain mapping. This includes uncovering both limitations and possibilities, as well as how the imaging setup at hand holds up against state-of-the-art LiDAR technology for this specific application. The end goal is to develop a model employing hyperspectral stereoscopic imaging, suitable for UAV flight altitudes of 20-60 m, capable of producing precisely georeferenced pixels in the form of a point cloud, without the need for a digital surface model.

1.4 Approach

In the process of developing the model exploiting the stereoscopic aspect of the imaging system at hand, and determining if it is suitable for UAV-based 3D terrain mapping, the following aspects are explored and considered:

1. Different approaches for determining the horizontal shifts of the images, i.e the disparity, should be explored. Both off-shelf algorithms and approaches based on previous research may be considered.
2. Accuracy and robustness are prioritised. In terms of disparity accuracy, 0.1 pixels precision should be obtainable based on similar experiments conducted previously by NEO.
3. The developed model should contain a scheme for involving multiple spectral bands and different parts of the electromagnetic spectrum to improve the performance.

4. Spatially misregistered hyperspectral data and synthetic data can be employed to benchmark and calibrate the developed model.

5. The point clouds derived from the stereoscopic aspect of the imaging system should be compared to corresponding LiDAR point clouds. This way the results can easily be quantified. The derived point clouds should be on the industry-standard file format .las for this to be done efficiently and effortlessly.

1.5 Thesis Outline

Starting with this introductory chapter, introducing the problem description along with some background and motivation to establish a foundation for the thesis, the thesis is organised into six different sections.

The next section introduces the theoretical background relevant to the presented work, focusing primarily on hyperspectral imaging and stereoscopic imaging. Here, the mathematical framework is presented and can be found towards the end in Section 2.2.3 and Section 2.2.4.

Section 3 contains a description of the hardware setup utilised, with all relevant parameters. This includes the HSI system, UAV platform, and LiDAR.

In Section 4, the method and implementation are presented. This includes a description of how the hyperspectral data is acquired as well as a comprehensive description of the developed model.

The results are both presented and discussed simultaneously in Section 5, where Section 5.2 contains the results derived from hyperspectral stereoscopic data.

Finally, a conclusion and suggestions for future work are presented in Section 6.

2 Theory

2.1 Hyperspectral Imaging

As stated in Section 1.2.1, hyperspectral imaging combines the techniques of spectroscopy and imaging to obtain an electromagnetic spectrum for each pixel. The spectrum obtained is divided into different bands of wavelengths, each band containing information concerning the radiative intensity of a specific wavelength [14]. The number of bands may exceed several hundred, in opposition to more conventional imaging where commonly three bands are considered, typically within the visible part of the spectrum. Hyperspectral imaging systems are not limited to the visible part of the electromagnetic spectrum, and systems may cover both the VNIR and SWIR spectral range. Hyperspectral imaging system's advanced "color vision" makes it an attractive sensing technology for numerous applications and fields of research, including astronomy, agricultural monitoring, food production, and military surveillance [20]. The subsequent section covers the basics of hyperspectral image acquisition, focusing mainly on pushbroom scanning as it is the scanning technique employed in the presented work, the geometrical characterisation of a hyperspectral imaging sensor as well as both spectral and spatial misregistration of hyperspectral data. At last, it should be noted that the two terms spectral bands and spectral channels will be used interchangeably throughout the remainder of the thesis. However, they both constitute the same concepts, namely the different electromagnetically wavelengths considered by a HSI system.

2.1.1 Hyperspectral Image Acquisition and Pushbroom Scanning

In terms of acquiring 3D hyperspectral cubes, a few scanning techniques are commonly considered. These are typically divided into three branches: point-scanning, line-scanning, and area-scanning techniques [21]. The point-scanning method considers, as the name suggests, a single point at a time, and the hyperspectral cube is obtained pixel by pixel by the movement of either the imaging system or the object of interest. Line-scanning methods are very similar to point-scanning methods, however, instead of considering a single point at a time, a line of points is considered. Similarly, the hyperspectral cube is obtained line by line with the movement of either the imaging system or the object of interest. Both point-scanning and line-scanning techniques are regarded as spatial scanning methods and are often referred to as whiskbroom and pushbroom scanning [21]. Area-based scanning techniques, on the other hand, obtain the hyperspectral cube by "movement" in the spectral dimension and are regarded as spectral-scanning methods. A 2D image of a specific wavelength is recorded one at a time, and the hyperspectral cube is built by stacking 2D images of the common scene for different wavelengths on top of each other [21].

In Figure 3 presented below, one approach to pushbroom scanning is illustrated. Here, incoming light, collected from a set of mirrors, passed through a narrow slit, is dispersed by a grating and focused onto a 2D detector array. The narrow slit defines the spatial dimension x on the 2D detector array, while the spectral dimension λ is determined by the optical characteristics of the grating and the lens. The 2D detector array forms a hyperspectral plane of the 3D hyperspectral cube. By movement of either the imaging system or the scene, the full 3D hyperspectral cube can be built in a line-by-line fashion.

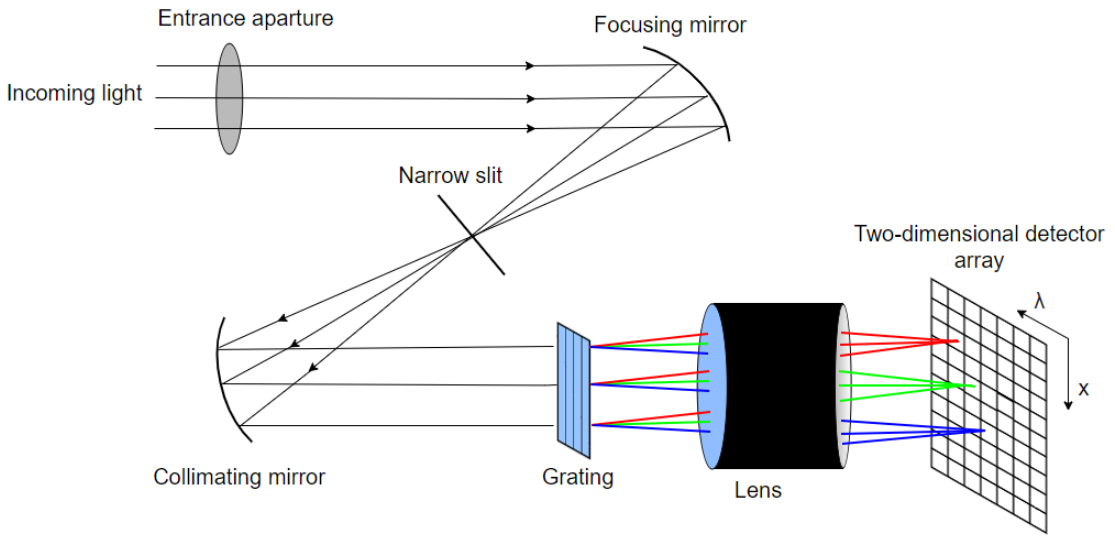


Figure 3: Light collected by a set of mirrors, passed through a narrow slit, is dispersed and focused on a 2D detector array. The 3D hyperspectral cube can be built in a line-by-line fashion by the movement of the imaging system. The same illustration was also used in [1].

Pushbroom Scanning - UAV Implementation

Figure 4 presented below, illustrates how pushbroom scanning can be implemented on a UAV for HSI from an aerial point of view. The configuration scans n pixels, all within the FOV of the imaging system, at a time in the across-track direction. By aerial movement in the along-track direction, the hyperspectral cube of the entire scene is constructed in a line-by-line fashion.

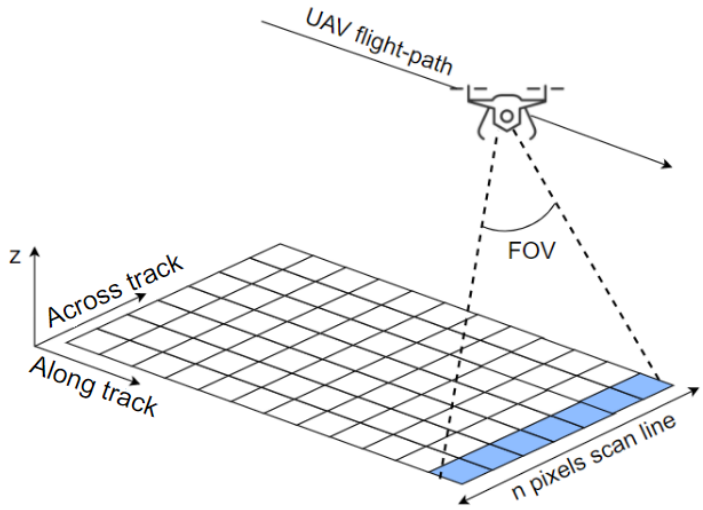


Figure 4: UAV implementation of pushbroom scanning. The same illustration was also used in [1].

The image acquisition frame rate, meaning the rate at which a new scan-line is recorded, is highly dependent on both the altitude of the UAV platform as well as the along-track velocity. It is desirable to optimize ground coverage while keeping a small on-ground pixel size and avoiding

across-track pixel elongation [22]. One approach to this is to maintain a constant frame rate while adjusting the UAV along-track velocity according to flight altitude. The UAV along-track velocity may then be determined by Eq. (1), where the dependence on flight altitude is added through the dependence on pixel size, which increases with flight altitude.

$$\text{Along-track velocity} = \text{Frame rate} \cdot \text{Across-track pixel size} \tag{1}$$

UAV pusbroom scanners are commonly equipped with a set of proprioceptive sensors to continuously monitor the internal state of the UAV platform [22]. This usually involves an inertial navigation system (INS), consisting of a global positioning system (GPS) and an inertial measurement unit (IMU). While the GPS provides positional data, the IMU provides orientation data such as roll, pitch, and heading. A precise INS is a prerequisite for accurate georeferencing of the depicted scene [22].

Another important aspect considering precise georeferencing of a depicted scene, captured by a UAV pushbroom scanner, is an accurate sensor model. The sensor model is the geometric characterisation of the HSI sensor and simply allows each pixel within a scan-line to be represented as a mathematical ray [23]. This is illustrated below in Figure 5. Here the black line illustrates the optical axis, while the dashed lines demonstrate the pixels when treated as a mathematical ray. The angles, θ, these rays make wrt the optical axis, in both x and y direction, are available in the sensor model of the HSI system. How an accurate sensor model can be combined with a precise and stable INS for accurate georeferencing will become evident in Section 2.2.3.

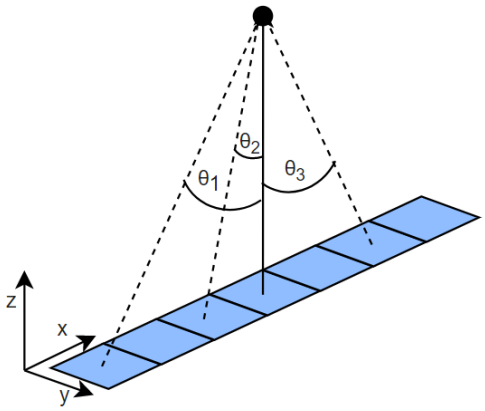


Figure 5: The sensor model allows each pixel to be represented as a mathematical ray. The angles, θ, these rays make wrt the optical axis, are contained in the sensor model of the HSI system. Based on a similar illustration from [1].

2.1.2 Spatial and Spectral Misregistration

One of the most important quality measures of a HSI system is the degree the collected data suffers from distortions, both spectral and spatial [15]. Spectral and spatial distortions are commonly referred to as "smile" and "keystone". Smile causes a center wavelength shift and is regarded as a spectral distortion, while keystone results in band-to-band misregistration and is , therefore, a spatial distortion [24]. Both effects and how they distort hyperspectral data are illustrated below in Figure 6.

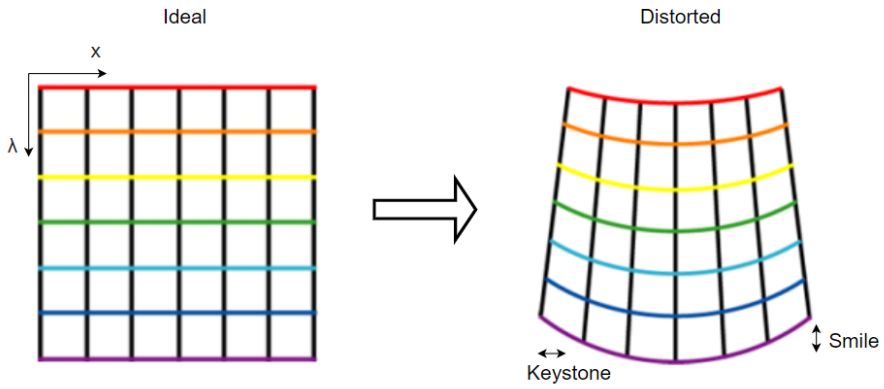


Figure 6: Ideal and distorted spectra. Smile introduces spectral distortions, while keystone introduces spatial distortions. Illustration from [25].

Here, it is evident how the smile and keystone effect may distort the spectral profile and thus degrade the quality of hyperspectral data. The origin of spatial and spectral misregistration usually boils down to two factors: optical aberrations causing light of different wavelengths to be spread over a region rather than focused to a point, and how the PSF of different wavelengths interacts with the optics of an HSI system [26]. These effects will be described in some detail next, focusing mainly on spatial misregistration.

Spatial misregistration can be explained by considering a polychromatic point source, where the total energy associated with each wavelength is normalized. Keystone causes the position of the imaged point source to be somewhat different across the spectral bands. This is illustrated below in Figure 7, where the peak of the PSF has a different position considering the individual wavelengths, λ. However, also evident from Figure 7, is how the PSF is smeared out differently across the spectral bands. The optics of a HSI system blurs the point source, illustrated by a smeared-out PSF in Figure 7. The amount of blur may vary significantly across the spectral channels [26]. The net result is energy from the spectra associated with each pixel, leaking into neighbouring pixels causing a nonphysical spectrum not corresponding to the actual object imaged. This is illustrated to the right in Figure 7, where the captured spectrum deviates significantly from the true spectrum of the polychromatic point source.

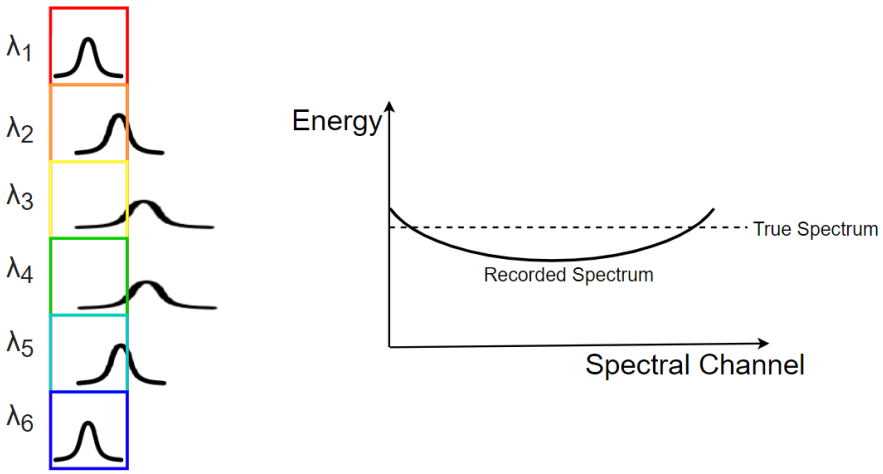


Figure 7: Spatial misregistration illustrated. Both keystone and wavelength-dependent smearing of the PSF result in a nonphysical spectrum not corresponding to the actual object imaged. Inspired by a similar illustration from [26].

2.2 Stereoscopic Imaging

As already touched upon in Section 1.2.1, stereo imaging addresses the subject of extracting 3D information from 2D images. When the 3D world is projected onto the 2D image plane the perception of depth is lost. However, by imaging a common scene from different points of view the depth of the scene may be recovered [2]. The subsequent chapter covers the basics of stereo vision, stereo image acquisition, how the depth of a scene can be recovered as well as stereo matching techniques, focusing mainly on the approaches relevant to the thesis.

2.2.1 Stereoscopic Vision and Human Stereopsis

As stated in Section 1.2.1, stereo imaging mimics the human approach to perceive the world in 3D. Retinal disparity, deriving from having horizontally separated eyes, results in our eyes perceiving the world slightly differently, from two marginally different points of view. Accordingly, all points incorporated in our vision will have a slight displacement, i.e disparity, when comparing the images formed on the left and right retina. The visual cortex processes the two images, determining the disparity of every point, resulting in a final image with an added dimension of depth [2]. The degree of depth is related to the disparity of the associated point, the greater the disparity, the closer that point lies in space [27]. This can easily be exemplified by holding one finger in the air, viewing it with one eye closed at the time, and watching how the position of the finger changes when closing and opening one eye at the time. Then, the same can be done when holding the finger further away, resulting in a much smaller change in position when opening and closing the eyes, suggesting the finger is located further away in space. The perception of depth due to the retinal disparity is known as stereopsis [27] and 3D vision based on comparing the relative position of objects viewed from a different position is known as stereo vision [27]. Below, in Figure 8 is an illustration of the process of stereopsis.

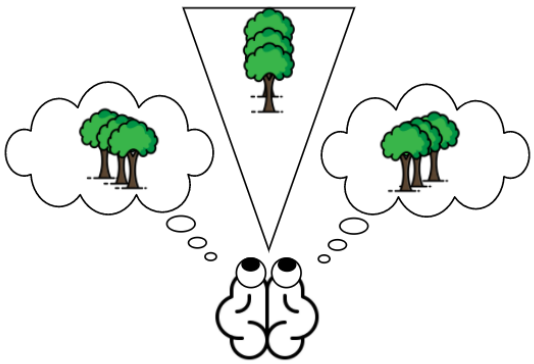


Figure 8: Human stereopsis. Slight retinal displacement leads to two different images forming on the left and right retina. The differences between the two images are used to construct a final image with an added dimension of depth. Inspired by a simmular illustration from [2].

2.2.2 Stereoscopic Image Acquisition

It is now evident, that by emulating human stereopsis, one can extract 3D information from 2D images captured of a common scene from different points of view, by determining the disparity of every point of the depicted scene. In terms of acquiring stereo image pairs, several procedures have been explored, the most traditional being the binocular setup presented in Figure 1 in Section 1.2.1. However, more complex setups can also be utilized, such as the use of arrays of multiple cameras or how a numerical stereo camera was developed in [28] by involving a laser, projector, and software for pattern analysis. Stereo image pairs can also be obtained using a single imaging system, simply by the movement of the system between imaging. This approach is common for satellite imagery, where the time between imaging can be days, months, or even years [29].

A principle aspect of stereo image acquisition is the process of stereo rectification. In a stereo-rectified image pair, both images share a common image plane, and all points located in the images can be found on the other image by searching along the same row [30]. An example of a rectified stereo pair is presented below in Figure 9. Here, all three points marked on one image can be found on the other by searching along the same row.

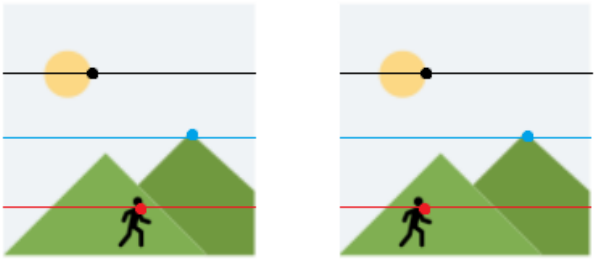


Figure 9: Rectified stereo pair. All points on the left image can be found on the right image by searching along the same row, and vice versa. The same illustration was also used in [1].

However, raw stereo images, and images in general, are commonly prone to distortions such as lens distortion causing a "fisheye" effect in the images, and they may not share a common image plane. These effects can all be corrected in the stereo rectification process, a mathematical procedure employed to achieve the ideal rectified stereo pair as illustrated in Figure 9 [3]. The procedure involves several steps, each correcting a particular feature. Figure 10 illustrates what a stereo rectification process could resemble. The process of stereo rectification will not be described in any further detail in this thesis, as it is not a part of the developed model, and a more detailed description can be found in [30]. However, the concept of a rectified stereo pair is brought to attention as it is an important aspect of stereo matching, as will be considered later, and most off-shelf stereo matching algorithms assume the stereo pair to be rectified.

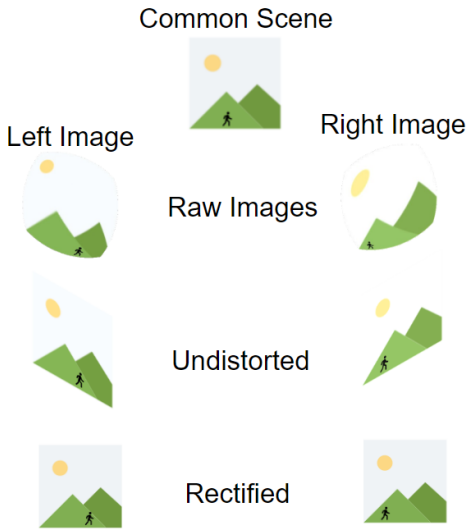


Figure 10: Stereo rectification process. The raw images contain "fisheye" distortions and the images do not share a common plane, while the final rectified image pair display no distortions and the two images share a common image plane.

2.2.3 Pushbroom Stereoscopic Image Model, Depth Estimation and Georeferencing

The fundamental principles of depth estimation following any stereo imaging system, are based on triangulation [2]. Triangulation denotes the concept of determining the unknown location of a point by forming triangles based on the known position of other points. The mathematical framework presented in this section was also employed in [1], the project report leading up to the thesis, and a similar description can also be found here.

Consider the situation presented below in Figure 11. A hyperspectral stereoscopic camera-rig consisting of two pushbroom scanners, shifted a distance B horizontally in the across-track direction wrt each other, images a common scan-line. The two imaging systems are shifted precisely horizontally, share a common image plane, have parallel optical axes, are triggered simultaneously, and the sensors share a common geometrical characterisation. Both cameras are equipped with an accurate INS such that the position, i.e the longitude, latitude, and altitude, of the left and right camera, P_L and P_R, when the scan-line is captured, is known. In addition, the geometrical characterisation of the sensors, i.e the sensor model, of both sensors is known, providing the angles θ. The objective is now to determine the position of the point P_1, imaged by both the left and right camera, in terms of longitude, latitude, and altitude.

It has already been well established how the depth of a point in space captured by a stereoscopic imaging system, denoted Z in Figure 11, is related to the disparity of the considered point. With this in mind, it can easily be verified that Z is given by

$$Z = \frac{B}{\tan\theta_1 - \tan\theta_2} \tag{2}$$

where B denotes the baseline, i.e the distance between the left and right camera [1]. In Eq. (2), Z's relation to the disparity can be identified in the denominator. The pixel coordinates of P_1 will differ in the scan-line captured by the left camera compared to the right camera. Hence, the angles θ_1 and θ_2 will not correspond, and their difference is given by the difference in x_1 and x_2, i.e the disparity. With Z known, the altitude of P_1 is simply determined by subtracting Z from the altitude of the left or right camera, provided by the INS data [1].

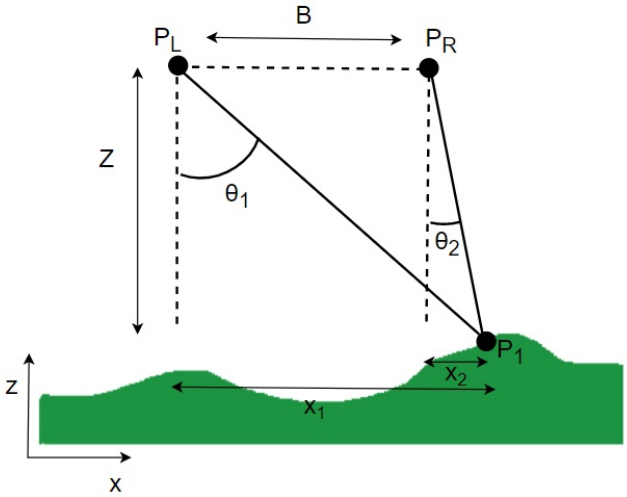


Figure 11: Pushbroom stereoscopic image model. Two cameras located at P_L and P_R image the point P_1. The coordinates of P_1 can be determined by triangulation wrt P_L or P_R. The same illustration was also used in [1].

With Z known, it is tempting to settle on the longitude and latitude of P_1 being given by simple trigonometric relations wrt P_L or P_R. However, since quantities such as longitude and latitude typically are given in units such as arc degrees, caution must be exercised. However, by assuming a

spherical earth and approximating its radius, Eq. (3) and Eq. (4) can be combined and re-arranged to obtain Eq. (5) and Eq. (6) which establish a relation between the longitudes and latitudes of P_L and P_1 [31] [1].

$$\alpha = \arctan 2(\sin \Delta\zeta \cdot \cos \phi_1, \cos \phi_L \cdot \sin \phi_1 - \sin \phi_L \cdot \cos \phi_1 \cdot \cos \Delta\zeta) \tag{3}$$

$$d = \arccos(\sin \phi_L \cdot \sin \phi_1 + \cos \phi_L \cdot \cos \phi_1 \cdot \cos \Delta\zeta) \cdot R \tag{4}$$

$$\phi_1 = \arcsin(\sin \phi_L \cdot \cos \delta + \cos \phi_L \cdot \sin \delta \cdot \cos \alpha) \tag{5}$$

$$\zeta_1 = \zeta_L + \arctan 2(\sin \alpha \cdot \sin \delta \cdot \cos \phi_L, \cos \delta - \sin \phi_L \cdot \sin \phi_2) \tag{6}$$

Here, ϕ symbolises latitude, ζ longitude, α bearing angle, δ angular distance, and R the radius of the earth. The bearing angle α is the angle of direction one has to move in order to get to P_1 from P_L while the angular distance δ is given by

$$\delta = \frac{d}{R} \tag{7}$$

where the distance d is the distance between P_L and P_1 if projected onto a common plane. If the disparity of P_1 has been determined, and Z calculated, d is given by Eq. (8) [1].

$$d = Z \cdot \tan \theta_1 \tag{8}$$

Thus, by employing Eq. (2), Eq. (5), Eq. (6), Eq. (7) and Eq. (8) the longitude, latitude and altitude of P_1 can be precisely determined. In fact, given that the position of P_L or P_R is recorded for each captured scan-line, and the disparity determined, the longitude, latitude, and altitude of the entire depicted scene can be determined. Consequently, determining the position of every pixel in the depicted scene involves two steps: first, establishing correspondence between the two images, i.e determining the disparity of every pixel, and secondly, precisely triangulating the position of every point in the depicted scene through the equations presented above [1].

It should be noted how Eq. (3), Eq. (4), Eq. (5) and Eq. (6) are all based on the assumption of a spherical globe with a constant radius. For larger scales this assumption can not be justified, as the earth has a shape more similar to an ellipsoid and its radius is far from constant [32]. Thus, the equations presented above can not be applied without introducing errors. With the radius of the earth approximated to be R=6 371 km, the error introduced when calculating the distance between two points introduced by the set of equations presented above, is according to [33], 0.334%.

Baseline versus Precision

The accuracy of the calculations based on the set of equations presented above is largely dependent on how well Z is estimated. In fact, any errors in Z lead to consequential errors in all forthcoming calculations. How well Z is estimated largely boils down to the disparity estimate, however, the imaging setup also plays a significant role through Eq. (2)'s dependence on B. This is exemplified below in Figure 12. Here, Eq. (2) is plotted considering two different stereo imaging setups. Both setups consist of two horizontally shifted pushbroom sensors, sharing the same geometrical characterization, with a sensor model varying between ± 0.17 radians over 620 spatial pixels. However, the left plot is based on a baseline of 0.075 m, while the right plot is based on a baseline of 1.5 m. Now consider depth estimates at around 60 m, marked as a blue dot in both plots. Taking into account the position of the blue dot in the two plots, it is evident how the precision of a narrow baseline stereo imaging setup is prone to errors in the disparity estimates. In fact, given the setup

with a baseline of 0.075 m, a disparity error of 0.1 pixels corresponds to an error in Z of 2.8 m, at 60 m. For the setup with a baseline of 1.5 m, a similar error would lead to an error in Z of only 0.13 m. Thus, given the case of a narrow baseline, accurate disparity determination, well beneath sub-pixel resolution, proves to be an absolute necessity for precise georeferencing.

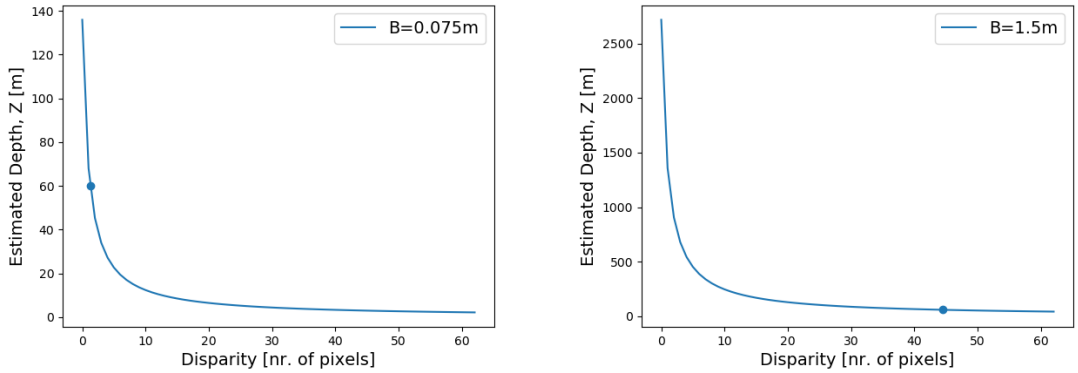


Figure 12: Eq. (2) plotted considering two identical stereo imaging setups, however, with different baselines. For a narrow baseline stereo rig, accurate sub-pixel disparity determination is necessary for precise georeferencing.

Conventional stereo imaging systems assure high precision by maximizing the B/Z ratio, and a B/Z ratio of around 0.6 is commonly chosen for aerial and satellite stereo imaging [34]. Setups consisting of one imaging system, which forms stereo pairs by movement in between imaging, enjoy the advantage of a fictitious baseline, making the depth estimations, in theory, independent of depth. However, blindly adopting as large of a baseline as possible, to maximize precision, also has drawbacks that should be considered. A large baseline increases the number of occlusions in the stereo pair. Occlusions are areas visible to only one of the images in stereo pair [35], and can therefore not have an associated disparity. With an increasing baseline, the overlapping region between a stereo pair decreases, increasing the occluded area. Additionally, with a larger baseline, taller objects tend to occlude smaller objects, a common issue seen in urban areas [34]. Both effects contribute to an increasing amount of occlusions and are illustrated below in Figure 13. Further, by increasing the baseline, the viewing geometry of a stereo pair deviates causing corresponding objects to possess a diverging reflectance profile, making the disparity more difficult to determine. Thus, despite a larger baseline having the advantage of not being dependent on sub-pixel disparity detection for appropriate precision, it still comes with a few downsides that also must be considered.

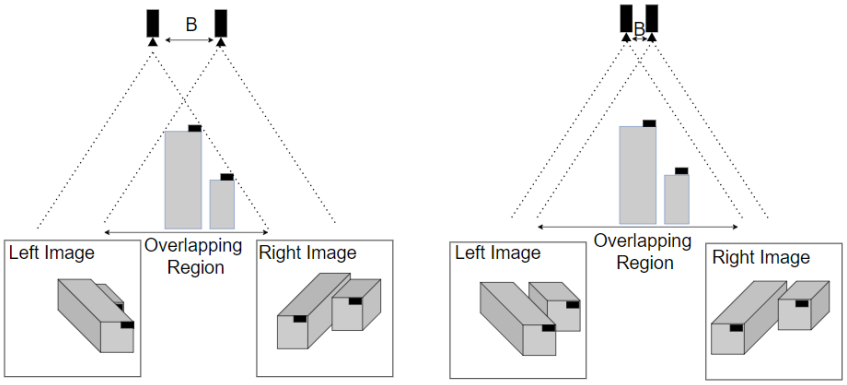


Figure 13: Wide baseline stereo configuration (left) and narrow baseline stereo configuration (right). A wide baseline results in a larger occluded area, both as a result of a smaller overlapping region, but also as a result of taller objects occluding smaller ones. Inspired by a similar illustration found in [34].

2.2.4 Stereo Matching

It is now evident, that for complete 3D reconstruction of a depicted scene, correspondence must first be established, and the disparity of every pixel must be determined. The correspondence process and associating each pixel with a disparity is referred to as stereo matching [3]. At first, this task may not sound particularly complex, however, the process of stereo matching is not straightforward. How can one really for sure know the position of two corresponding points in a stereo pair? This is referred to as the stereo correspondence problem, and is a topic studied for several decades [3]. Stereo matching is typically complicated by bad illuminated scenes, low texture, periodic patterns, radiometric differences, and occluded areas. Additionally, two separate imaging sensors may introduce stochastic signal variation due to having slightly different transfer functions [2], complicating matters even further. These effects may lead to several solid candidates for correspondence or non at all. To overcome the stereo correspondence problem, two types of algorithms are mainly employed, dense stereo matching algorithms and sparse stereo matching algorithms [3]. Where sparse stereo matching algorithms only consider the most certain of matches, typically based on very distinct features in a stereo pair, dense stereo matching algorithms address the issue of determining the disparity of the entire depicted scene. As the presented work employs stereo imaging for 3D reconstruction of entire depicted scenes, dense stereo matching is primarily addressed and described. Some of the mathematical framework presented in this section was also described in [1], particularly concerning intensity-based stereo matching.

Sparse Stereo Matching

When stereo matching, specific features of a depicted scene are easier to match than others. Robust matching features are typically very distinct in their environment, and corners, edges, or line segments of a depicted scene have proven to be solid features for stereo matching, and are considered by sparse stereo matching algorithms [2]. One approach to sparse stereo matching is to convert the intensity data of solid features, such as the ones just mentioned, to a set of attributes used to determine correspondence. The advantage of sparse stereo matching is the high confidence of the detected matches. However, as the features used for matching typically are unique in their environment, most of the depicted scene is left in the state "no feature present" and no correspondence detected [2]. Thus, fitting algorithms or interpolation methods may be applied to determine the correspondence of the remainder of the scene.

Dense Stereo Matching

As already emphasized, dense stereo matching addresses the issue of determining the disparity of every pixel in a depicted scene. In comparison to sparse stereo matching, dense stereo matching proves to be a complex task, as areas of the depicted scene inefficient for stereo matching must also be considered. Such areas typically exhibit a continuous nature, low texture, or bad lighting. The most basic dense stereo matching methods are based on comparing windows of pixels between the images in the stereo pair. Correspondence can be determined by finding the position of the best matching window on one image when compared to a similar window on the other image. By repeating the process over the entire image, the disparity of the entire depicted scene may be determined. In order to determine the best matching window, a measure of similarity must be implemented [1]. This roughly divides dense stereo matching methods into two categories: intensity-based approaches and phase-based approaches [36]. Where intensity-based stereo matching methods evaluate pixel intensities directly as a measure of similarity, phase-based approaches are based on first deriving Fourier-domain images from the pixel intensities. However, stereo matching algorithms only operating under one constraint, the similarity measure, have proven to be prone to false matches and struggle with natural and gradually changing terrain [37]. To encounter this, and also facilitate smooth transitions throughout the depicted scene, another constraint is commonly added, typically by considering the disparities of nearby and previous evaluated pixels. Both constraints can be integrated into a global energy function, as presented in Eq. (9) [1].

$$E(D) = \sum_{p} C(p, D_p) + \sum_{p,q \in N} R(p, D_p, q, D_q) \tag{9}$$

Here, the first term C(p, D_p) represents the matching cost and provides an indication of the similarity between the pixel p and all other viable matches on the other image, given by D_p. Ideally, the most probable matches yield the lowest matching cost. The second term, R(p, D_p, q, D_q) often referred to as the smoothness term, typically has the form

$$R(D_p, D_q) = \begin{cases} 0 & D_p = D_q \\ P_1 & |D_p - D_q| = 1 \\ P_2 & |D_p - D_q| > 1 \end{cases} \tag{10}$$

and is integrated to facilitate smooth transitions, by adding penalties of 0, P_1, or P_2 depending on the difference in disparity D_p-D_q, considering pixel p and pixel q within the range of N pixels. Commonly $P_2 > P_1$ to promote smooth disparity transitions, while discontinues are preserved by a constant P_2 [38]. The goal of some dense stereo matching algorithms now becomes finding the disparities, D, minimizing Eq. (9). The work presented in [39] summarizes this approach to stereo matching in four steps:

1. Matching cost computation. C(p, D_p) is calculated based on a similarity measure and can be both intensity-based and phase-based.
2. Cost aggregation. Cost aggregation is added through the term R(p, D_p, q, D_q) by adding penalties depending on the disparities of nearby pixels located within a support region of the pixel under consideration, both to reduce the effect of noise and promote disparity smoothness [3].
3. Disparity optimization. Disparity optimization includes the process of deciding upon the most likely disparity, based on the two previous steps.
4. Disparity refinement. At last, disparity refinement is included to optimize the results. This commonly involves noise reduction filters, filling of disparity holes, and detection of inconsistencies [40].

Intensity-based stereo matching methods are, as already stated, based on measuring pixel intensities directly. Some approaches include calculating the sum of absolute difference (SAD), the census transform, known for its robustness against luminance variations in stereo image pairs [41], the Birchfield–Tomasi (BT) measure as described in [42] or a more statistical approach based on mutual information as presented in [38]. The Open Source Computer Vision Library (OpenCV) provides an intensity-based stereo matching algorithm based on the work presented in [38]. The algorithm employs the BT similarity measure, an approach based on comparing interpolated intensity functions surrounding each pixel in the stereo pair [42]. Further, as explained in [38], Eq. (9) is minimized with dynamical programming, by approximating Eq. (9) as discrete segments, L, in several directions, r, throughout the image. For each segment and direction, an optimized cost L_r(p,d) for a given pixel p and disparity d is determined in compliance with Eq. (11) [38].

$$\begin{aligned} L_r(p,d) = C(p,d) + \min\{&L_r(p-r,d), L_r(p-r,d-1) + P_1, \\ &L_r(p-r,d+1) + P_1, \min_i(L_r(p-r,i)) + P_2\} - \min_k(L_r(p-r,k)) \end{aligned} \tag{11}$$

Here, C(p,d) is the same matching cost as in Eq. (9), calculated from the similarity measure, and the second term, accountable for cost aggregation, is added based on the minimum path cost to reach the previous pixel, considering the different disparities, d, d-1, d+1 and the entire range of disparities considered, i, with the appropriate penalties P_1 and P_2 added according to Eq. (10).

Finally, the minimum path cost to the previously evaluated pixel is subtracted. According to [38], the number of directions r considered, should at least be 8 for appropriate performance, including two vertical, two horizontal, and four diagonal directions. S(p,d), the accumulated cost, is summed for every r in compliance with

$$S(p,d) = \sum_r L_r(p,d) \tag{12}$$

and the disparity is determined by the winner-takes-all approach, according to Eq. (13) [38].

$$D(p) = \underset{d}{\operatorname{argmin}}(S(p,d)) \tag{13}$$

A more detailed description of this intensity-based approach to stereo matching can be found in [30], documentation provided by OpenCV, or in [1] the project report leading up to the thesis.

Phase-based stereo matching methods are based on deriving Fourier-phase images from the stereo pair based on their pixel intensities [2]. It has proven to be an approach robust against noise and radiometric differences [43], and ideal for achieving sub-pixel disparity resolution. It is based on the Fourier-shift property, stating the Fourier-transform of two images, translated wrt each other in the spatial domain, will be the same with an added phase term indicating the translation, given that the images are much larger than the translation.

The 2D discrete Fourier transform (DFT) of an image $f_1(x,y)$ is given by

$$F_1(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f_1(x,y) \exp\left(-2j\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right) \tag{14}$$

where x and y are the spatial domain components, ranging from x=0,1,2...,M-1 and y=0,1,2...,N-1, and u and v are the frequency domain components contained within the same range [43]. Considering another image $f_2(x,y)$ spatially translated a distance Δx and Δy wrt $f_1(x,y)$, its DFT will be given in compliance with Eq. (15).

$$F_2(u,v) = F_1(u,v) \exp\left(-2j\pi\left(\frac{u\Delta x}{M} + \frac{v\Delta y}{N}\right)\right) \tag{15}$$

From Eq. (15) it is evident how the two images share the same DFT, plus an additional phase-term containing information concerning the spatial translations Δx and Δy. Computing the normalized cross-power spectrum, Q(u,v) yields

$$Q(u,v) = \frac{F_1(u,v)F_2^*(u,v)}{|F_1(u,v)F_2^*(u,v)|} = \exp\left(-2j\pi\left(\frac{u\Delta x}{M} + \frac{v\Delta y}{N}\right)\right) \tag{16}$$

where the star indicates the complex conjugate. The phase correlation function, PC(x,y), can be computed by the 2D inverse discrete Fourier transform (IDFT) of Q(u,v), i.e

$$PC(x,y) = F^{-1}Q(u,v) = \delta(x + \Delta x, y + \Delta y) \tag{17}$$

where F^{-1} denotes the IDFT operation, and δ symbolises the Kronecker delta function, exhibiting a peak at the position (Δx, Δy) [43]. Thus, by computing PC(x,y) and determining its peak, the translations Δx and Δy between the two images can easily be determined.

The issue, however, with only using PC(x,y) to determine the translation between two images, is the fact that images are discrete functions, and the peak of PC(x,y) will always be located at an

integer pixel value. Hence, sub-pixel disparity resolution, an absolute necessity for a narrow baseline stereo-rig, is not obtainable only by considering Eq. (17). One solution to this is oversampling the images prior to computing PC(x,y). However, this approach is known to notably increase the computing load, introduce interpolation artifacts, and has limited precision [34]. Another approach to obtaining sub-pixel disparity resolution considering PC(x,y), is to fit another continuous function to the phase correlation data, and precisely determine its peak [44]. When noise components and other random artifacts are added to the images of consideration, their phase correlation function deviates from a perfect delta function [45]. Instead, PC(x,y) approaches a bell shape and can be approximated by a 2D polynomial function [44]. As far as what function to fit the PC(x,y) data, a simple parabola, Gaussian function or sinc function may all be employed. Determining the sub-pixel peak of PC(x,y) by function fitting have reported to reliably provide 0.1 pixels precision [34].

The work presented in [46] proposes a procedure to determine the translation between two images directly in the Fourier-domain, without calculating PC(x,y). The methodology is based on the phase-difference matrix, i.e the argument of Q(u,v) from Eq. (16), of two images translated Δx and Δy wrt each other. The phase-difference matrix gives rise to a 2D saw-tooth signal repeated Δx times along the u-axis and Δy times along the v-axis [46]. Below in Figure 14, the phase-difference matrix is plotted along the u-axis derived from two images where Δx=5 pixels. For circumstances where the translations are not given by an integer number of pixels, the saw-tooth signal will be repeated some integer number of times, plus a fraction of a period corresponding to the sub-pixel part of the translation.

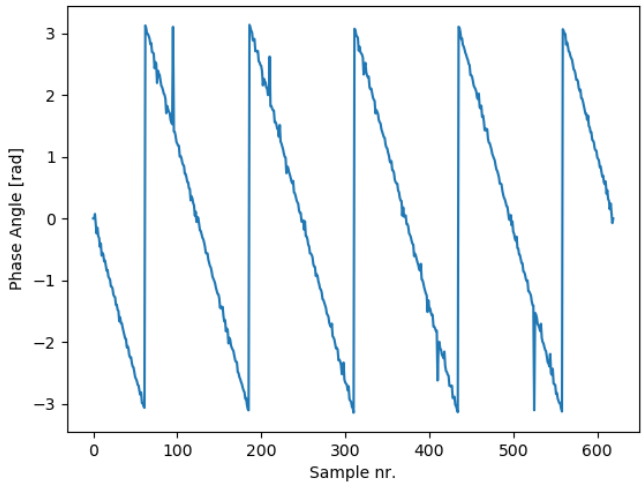


Figure 14: The phase-difference matrix of two images translated wrt each other gives rise to a 2D saw-tooth signal along both frequency axis. The translation corresponds to the number of times this signal is repeated along both frequency axes. Here plotted along the u-axis with Δx=5 pixels.

The question now becomes how this property can be applied efficiently and accurately to determine the translation between two images, without manually counting the number of cycles. The work presented in [34], approaches this by viewing the phase-difference matrix as a 2D plane in the u-v Cartesian coordinate system, given by

$$c = au + bv \tag{18}$$

where a and b simply is the slope of c in the u and v directions. With a and b known, the translations Δx and Δy can be determined from the relations presented in Eq. (19) and Eq. (20) [46].

$$\Delta x = \frac{M}{2\pi} a \tag{19}$$

$$\Delta y = \frac{N}{2\pi} b \tag{20}$$

The origin of the saw-tooth shape of the phase-difference matrix arises from the phase-angles being 2π wrapped. The phase-angles must therefore first be unwrapped in both u and v directions before a and b can be determined. The plot presented in Figure 15 illustrates the same phase-difference matrix as in Figure 14 after unwrapping in the u-direction. Evident from Figure 15 is how the slope, a, may effortlessly be determined, providing a solid framework for sub-pixel disparity estimation. In fact, phase-based stereo matching algorithms calculating the displacement directly in the Fourier-domain, such as the approach just described, have been reported to reliably provide 0.05 pixels resolution [34].

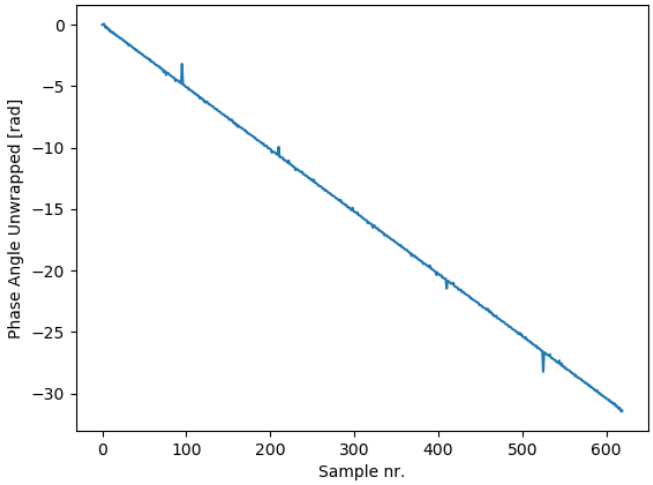


Figure 15: The unwrapped phase-difference matrix along the u-direction. By determining the slope, a, of the line segment, Δx can precisely be determined.

2.2.5 Stereo Matching Considerations

As already emphasized, stereo matching is a complex process and not straightforward. To maximize the performance of a stereo matching algorithm, and minimize the number of false matches some aspects must carefully be considered, as will be elaborated on next.

Stereo Rectification

With regards to stereo rectified image pairs, all corresponding points will be located on the same row, as stated in Section 2.2.2. This effectively reduces the search region of a stereo matching algorithm from two dimensions to one dimension, decreasing the amount of computing power needed, as well as the risk of detecting a false match considerably. Thus, for image pairs not stereo rectified, a stereo rectification step should be implemented prior to stereo matching.

Pre-Processing

A pre-processing step is typically implemented in order to optimize the matching processes. For intensity-based stereo matching algorithms, the pre-processing step traditionally involves noise

removal, deblurring, and contrast enhancement [47]. However, more advanced procedures can also be implemented, such as employing a horizontal Sobel-operator highlighting edges, normally a robust feature for stereo matching, crossing the rows in a stereo pair [3].

With regards to phase-based stereo matching algorithms, a common pre-processing step is to employ a Hamming window. A Hamming window mitigates the "wrap around" effect apparent at every edge of a 2D image DFT, causing discontinuities not apparent in the real world [45]. Further, the spatial frequency components of a natural image, tend to have most of their energy centered around the lower frequency components [45]. However, Eq. (16) values both low and high frequency components. The work presented in [45] addresses this by modifying Q(u,v) with a weighting function. With an image size of 251×251 pixels^2, the work presented in [45], achieves a pixel translation accuracy of 0.01 pixels, with a Gaussian-shaped weighting function centered around the zeroth spatial frequency, highlighting the importance of an appropriate pre-processing step.

Window Size

The window size regulates the number of pixels considered at a time when stereo matching. In terms of preserving detail in the depicted scene, its value plays a crucial role. While a larger window size makes the search for correspondence easier and more reliable, a smaller window size preserves detail to a greater extent with the downside of introducing a greater number of false matches. The level of detail in the depicted scene as well as how prone it is to false matches must therefore carefully be evaluated before an appropriate window size is determined [3]. In the ideal case, the window size is large enough to promote a reliable matching process, and its shape is such that every pixel within a window has the same ground truth disparity level. This way the number of false matches will to be kept a minimum, while the conservation of detail is maximized.

For images captured by a UAV-based pushbroom sensor, other aspects of the window size, not seen for more conventional imaging systems, must also be considered. As the image is built in a line-by-line fashion, the position of the UAV is continuously changing, including its distance to the scene. This way the disparity associated with a common plane will also vary in the along-track direction. This aspect must also be considered before employing pixels from a large number of scan lines.

Disparity Range

The disparity range determines the search range of a stereo matching algorithm and establishes what is known as the horopter in computer vision, meaning the 3D volume covered [30]. This value should be set based on the scene and within the range one would expect to detect matches. A range too limited could result in no potential matches, while a range too large may result in several good candidates. Figure 16 presented below, highlights the importance of an appropriate disparity range with regards to stereo matching algorithms based on minimizing Eq. (9). In Figure 16 there is a considerable disparity associated with the depicted person. In order to determine the actual disparity associated with the person, the disparity value minimizing the matching cost, the disparity range [Min. Disparity, Max. Disparity] must include the true disparity.

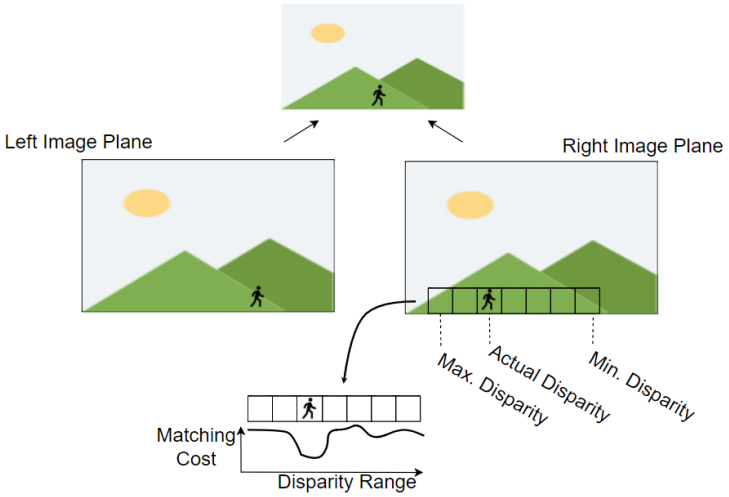


Figure 16: The disparity range established the search range of a stereo matching algorithm. The range should not be too large or too narrow, and must contain the true disparity. The same illustration was also used in [1].

An appropriate disparity range is also highly relevant considering other stereo matching algorithms not necessarily based on minimizing Eq. (9). Below presented in Figure 17 are two examples of what the phase correlation, PC, function may look like for two images translated horizontally wrt each other, where Δx=5 pixels. The left plot exhibits a very distinct peak located at sample nr. 5 and can be considered the ideal case for stereo matching. In the right plot, on the other hand, both images have been contaminated with random noise before the PC function is computed. This leads to several peaks in the PC function, many of them greater than the peak located at sample nr. 5. However, by setting the disparity range equal to for example [0,10] pixels, most of the PC function can be discarded, and the peak located at sample nr. 5 determined, even for the situation presented to the right in Figure 17. For PC function fitting, the disparity range can be made very narrow, as the PC function typically exhibits a very distinct peak and only a few data points are needed for precise function fitting [45].

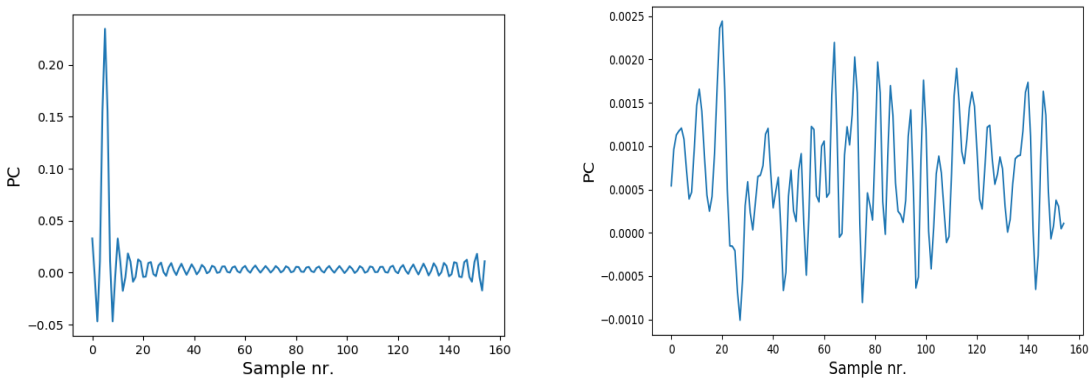


Figure 17: PC function calculated for two images translated 5 pixels horizontally wrt each other. The left plot exhibits a very distinct peak located at sample nr. 5 and can be considered the ideal case. In the right plot, both images have been contaminated with random noise and the PC function deviates from its ideal shape, and employing a suitable disparity range becomes critical.

Post-Processing

The post-processing step, also referred to as disparity refinement in Section 2.2.4, is all about optimizing the results and replacing unluckily matches with better and more likely alternatives [3]. This commonly involves interpolation of missing disparity areas and noise reduction filters [40]. Some algorithms also employ a left-right disparity constancy check which typically uncovers if the matching algorithm would yield the same corresponding matches if executed in reverse [3].

3 Imaging System Setup

The complete imaging setup is provided by NEO and consists of two separate units: the hyperspectral imaging system Hyspex Mjolnir VS-620 (Hyspex, Norsk Elektro Optikk, Oslo, Norway) and the UAV platform BFD XQ-1400S (BFD Systems, Pennsauken, New Jersey, USA). Both units are described in the upcoming sections. Given that the same imaging setup was employed in the project work leading to up the thesis, a very similar description of the setup can be found in [1].

3.1 Hyperspectral Camera Rig

The hyperspectral camera rig employed is Hyspex Mjolnir VS-620. The setup provides two co-aligned hyperspectral cameras, Mjolnir V-1240 and Mjolnir S-620. Mjolnir V-1240 covers the VNIR spectral region at 400 – 1000 nm, with a total of 200 bands and a spectral resolution of 3 nm, and Mjolnir S-620 covers the SWIR spectral region at 970 – 2500 nm, with a total of 300 bands and a spectral resolution of 5.1 nm. The cameras have an overlapping wavelength region between 970-1000 nm consisting of a total of 13 bands. Both cameras are mounted on a common chassis with B=0.075 m. The total weight of the setup is just under 6 kg, making it a suitable option for UAV-based imaging. A simplified illustration of Hyspex Mjolnir VS-620 is presented below in Figure 18, while the actual Hyspex Mjolnir VS-620 is presented in Figure 19.

The two cameras have a FOV ranging between ±0.17 radians, two parallel optical axes, and identical sensor models. Mjolnir V-1240 provides 1240 spatial pixels for each scan line, each pixel with a pixel FOV of 0.27/0.54 milliradian (mrad) in the across/along-track direction, while Mjolnir provides 620 spatial pixels for each scan line, each pixel with a pixel FOV of 0.54/0.54 mrad in the across/along-track direction. Thus, the combined number of spatial pixels for each scan line is 620, all contained within the FOV of ±0.17 radians. The sensor model varies within the FOV of ±0.17 radians in the across-track direction and is zero in the along-track direction. Additional details concerning Hyspex Mjolnir VS-620 can be found in the data sheet presented in [48]. For the remainder of the thesis, Mjolnir S-620 is regarded as Camera 1, while Mjolnir V-1240 is regarded as Camera 2.

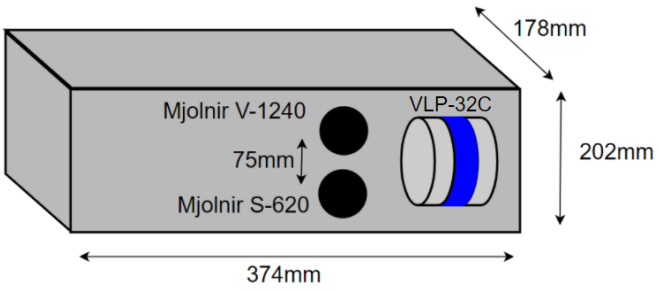


Figure 18: Hyspex Mjolnir VS-620. The configuration provides two hyperspectral imaging systems shifted 75 mm wrt each other, in the across-track direction. An additional LiDAR, VLP-32C is mounted to the setup. The same illustration was also used in [1].

Both imaging systems are incorporated with the PicoITX i7 computer and the Applanix APX-15 UAV INS. The Applanix APX-15 can provide positional data with accuracy down to 2 cm [22], providing a solid framework for accurate georeferencing. Additional details concerning the Applanix APX-15 UAV INS can be found in [22].

As illustrated in Figure 18, Hyspex Mjolnir VS-620's external input/output connector is utilized to integrate the system with an additional LiDAR, the VLP-32C. VLP-32C provides a range of up to 200 m, generation of approximately 1,200,000 data points every second with an accuracy down to 3 cm over a FOV of 40°, when mounted to Hyspex Mjolnir VS-620. The laser operates at a wavelength of approximately 903 nm, in the near-infrared region of the electromagnetic spectrum. Additional details considering VLP-32C can be found in its datasheet, presented in [49].

3.2 UAV Platform

The UAV platform utilized for imaging is the octocopter BFD XQ-1400S. This octocopter provides complete control of all flight parameters such that the altitude, speed, and direction may easily be adjusted, providing a large degree of flexibility in terms of frame rate and integration time, two important aspects of pushbroom scanning. To compensate for vibrations and high frequency rolling and pitching, characteristic of an octocopter, the setup is equipped with a high-quality gimbal, the gStabi H16. Provided the Hyspex Mjolnir VS-620 payload, BFD XQ-1400S allows for a flight time of around 30 minutes. The complete imaging setup, including the UAV platform and the hyperspectral stereoscopic camera rig, is presented below in Figure 20.

Figure 19: Hyperspectral imaging system Hyspex Mjolnir VS-620. The same illustration was also used in [1].

Figure 20: Complete hyperspectral stereoscopic imaging setup, including the octocopter BFD XQ-1400S and the hyperspectral camera rig Hyspex Mjolnir VS-620. The same illustration was also used in [1].

4 Method and Implementation

4.1 Hyperspectral Imaging and Pushbroom Scanning

The imaging setup including both the HSI system Mjolnir VS-620 and the UAV platform BFD XQ-1400S is operated by UAV specialists from NEO. The location selected for imaging as well as important aspects of how the scene is scanned are considered next.

4.1.1 Scene and Location

The location chosen for imaging is Losby shooting range, a site in proximity of Lørenskog. The scene is presented below in Figure 21, and exhibits a mixture of level ground, gradually changing elevation, trees, and bushes, as well as more distinct objects such as a house and a container. The diversity of the scene makes it a suitable location to fully evaluate the robustness of the developed model, and how it manages different types of terrain. It is also worth highlighting the amount of waste lying on the ground due to the site normally being used for clay target shooting. Below presented in Figure 22, is a map of the depicted area with contours demonstrating the change in elevation, while Figure 23 illustrates the scene depicted from 60 m UAV flight altitude.

Figure 21: Losby shooting range. The scene exhibits a mixture of level ground, gradual changing elevation as well as more distinct objects. The same illustration was also used in [1].

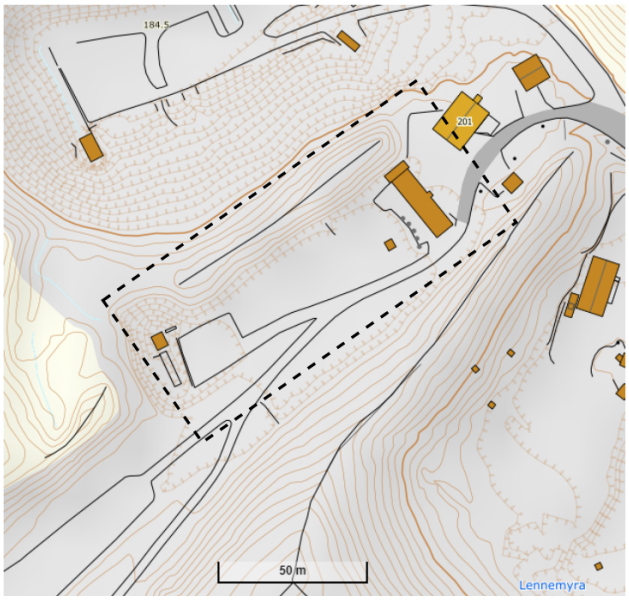


Figure 22: Map of Losby shooting range. The dashed rectangle highlights the imaged area. Map downloaded from [50].

Figure 23: Losby shooting rang from an aerial point of view. The scene is scanned in three flight lines, flight line 1, flight line 2, and flight line 3 from left to right.

4.1.2 Pushbroom Scanning

The scene is scanned with the imaging setup described in Section 3, including the HSI system Hyspex Mjolnir VS-620 integrated with the VLP-32C LiDAR, mounted to the UAV-platform BFD XQ-1400S. The scene is imaged with three different UAV flight altitudes, 20 m, 40 m, and 60 m above ground, to fully test the robustness of the developed model. The scene is scanned in three different flight lines for each of the considered altitudes, resulting in three separate images. This is illustrated in Figure 23, where the images resulting from flight lines 1, 2, and 3, given a UAV altitude of 60 m above ground are presented.

Each scan line captured by Hyspex Mjolnir VS-620 has a 25% overlap with the neighbouring flight lines, while the overlap considering VLP-32C is 50% due to its two times greater FOV. The frame rate of Hyspex Mjolnir VS-620 is kept constant for all UAV flight altitudes, with the frame rate of Camera 2 kept double the rate of Camera 1 due to its two times smaller pixel FOV in the across-track direction. The frame rates are kept at approximately 92.592 frames/s and 46.297 frames/s for Camera 2 and Camera 1 respectively. With a constant frame rate, the UAV along-track velocity is adjusted for the different flight altitudes according to Eq. (1). For each captured scan line, INS data regarding both Camera 1 and Camera 2 is registered. This includes frame number, longitude, latitude, altitude (wrt some reference), pitch, roll, and heading.

4.2 Stereo Matching and Disparity Calculation

Prior to stereo matching, the hyperspectral images captured by Camera 1 and Camera 2 are processed and calibrated radiometrically in Hyspex RAD, software provided by NEO. Additionally, the data is corrected for both smile and keystone distortions and the images from Camera 2 are down-sampled to match the image size of Camera 1, all while maintaining their stereoscopic relation. Commonly, images resolving from pushbroom scanning are also straightened out to remove distortions seen as oscillations in the images due to rolling and pitching of the UAV platform during image acquisition. However, this would remove the stereoscopic relationship between Camera 1 and Camera 2 and is therefore not included. It is assumed that Camera 1 and Camera 2 are perfectly co-aligned with parallel optical axes, sharing a common image plane, and have equal geometrical characterisation. Thus, the images are considered to be stereo rectified with only horizontal disparity. Additionally, Camera 1 is considered the left camera of the two.

For stereo matching and disparity calculation, certain considerations are taken into account:

Accurate subpixel disparity resolution is an absolute necessity due to the narrow baseline, only 75 mm, of the imaging system. This becomes evident when comparing the B/Z ratios for the considered UAV flight altitudes, presented in Table 1, to the B/Z ratio of more conventional stereo rigs which typically lies around 0.6 [34].

Table 1: B/Z ratios for the considered UAV flight altitudes.

UAV Flight Altitude [m above ground]	B/Z
20	0.00375
40	0.001875
60	0.00125

Radiometric differences are expected in the stereo image pairs. This is due to the different spectral bands covered by Camera 1 and Camera 2. However, radiometric differences should also be expected in the overlapping wavelength region, 970 - 1000 nm, as the no channels are exactly overlapping due to the two cameras' different spectral resolutions.

Periodic patterns due to the images not being corrected for oscillations due to rolling and pitching of the UAV platform, in combination with all the waste lying on the ground of the depicted scene, are expected to be present in the data. One example of such a pattern is presented below in Figure 24.

Figure 24: Oscillations of the UAV platform in combination with the amount of waste lying on the ground makes the images prone the periodic patterns running horizontally.

To cope with periodic patterns and radiometric differences, the robust phase-based PC function fitting approach as described in Section 2.2.4 and [43] is employed. Additionally, to increase the precision the important principles with regards to window size and weighting of the different spatial frequency components, as touched upon in Section 2.2.5 and described in detail in [45], are employed. At last, to maximize the disparity accuracy and resolution the principles of calculating the disparity directly in the Fourier-domain, as explained in Section 2.2.4 and also utilised in [34], are implemented. The flowchart of the complete stereo matching algorithm is presented below in Figure 25 and will be described in detail next.

4.2.1 Developed Stereo Matching Algorithm

The main stages of the developed stereo matching algorithm are outlined below in Figure 25 and consist of the following:

1. Start: Both hyperspectral cubes from Camera 1 and Camera 2 with desired spectral channels are loaded. Further, corresponding spatial windows of identical size, M×N pixels^2, from the two cubes are extracted, based on one spectral channel from Camera 1, λ_n, and one spectral channel from Camera 2, λ_m.

2. Pre-Processing: This step is all about facilitating the forthcoming PC function fitting and involves two steps: one in the spatial domain and one in the Fourier-domain. First, a Hamming window is applied to both considered windows from Camera 1 and Camera 2 to mitigate the "wrap around" effect at the edges of 2D image DFTs. Secondly, the cross-power spectrum, Q(u,v), derived from the DFT of the two windows, is modified with a rectangular low-pass filter of the form

$$H(u,v) = \begin{cases} 1 & u \leq U, v \leq V \\ 0 & otherwise \end{cases} \tag{21}$$

in order to highlight the lower spatial frequency components associated with natural images. U and V are set based on the work presented in [45], where values satisfying $\frac{U}{M}=\frac{V}{N}=0.5$ gave the most accurate results for rectangular low-pass filters.

3. PC Function Fitting: The peak of the PC function is approximated by fitting the function to a continuous function of a similar shape. For this, step it is assumed that the stereo pair is perfectly rectified and Δy from Eq. (17) is set to zero, and the PC function only has to be fitted in one dimension. The initial continuous fitting function of choice is the bell-shaped Gaussian function of the form

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2) \tag{22}$$

where σ denotes the standard deviation of the function, indicating the spread of the bell-shape, and μ the expected value, the value the peak is centered around and effectively the value of interest to approximate. For circumstances where the PC function can not be fitted to Eq. (22), a second effort is carried out. This time the sinc function of the form

$$sinc(x-\mu) = \frac{\sin(\pi(x-\mu))}{\pi(x-\mu)} \tag{23}$$

is employed, shifted a distance μ from the origin. To optimize the process of fitting the PC function to either Eq. (22) or Eq. (23), an initial guess of μ is provided. The initial guess of μ is the peak of the PC function within the expected disparity range. The disparity range is determined based on the values presented in Table 2, and the expected disparities of the scene for the specific UAV flight altitude. The number of data points from the PC function employed to fit either Eq. (22) or Eq. (23) is 7, three points to the left of the peak of the PC function and three points to the right. 7 data points for each dimension yielded the most accurate results considering the work presented in [45]. After the PC function is fitted to either Eq. (22) or Eq. (23), the location of its peak is determined as a decimal number and is regarded as the disparity level of the entire M×N pixels^2 window. For circumstances where the PC function neither can be fitted to Eq. (22) nor Eq. (23) and the disparity can not be determined, the window is assigned a disparity value of -1 and is regarded as a disparity hole. The windows is also regarded as a disparity hole for circumstances where the disparity is estimated to be outside the disparity range.

If the PC function is successfully fitted to either Eq. (22) or Eq. (23) and the disparity determined, a PC_{score} is calculated. The PC_{score} is simply a measure of how likely the disparity estimate resulting from PC function fitting is to be accurate and is part of the disparity optimization step of the algorithm. It is calculated by comparing the initial peak of the PC function located at pixel nr. μ to the nearby values, i.e Eq. (24).

$$PC_{score} = \frac{|PC(\mu)|^2}{|\sum_{x=\mu-5}^{\mu+5} PC(x)|^2} \tag{24}$$

With the PC_{score} calculated, a new pair of spectral bands of the same spatial window is loaded, and the process just described is repeated. This is repeated such that each spectral channel from Camera 1 and Camera 2 undergoes the PC Function Fitting step, and the pair maximizing PC_{score} is employed for the next steps of the stereo matching algorithm.

4. Window Alignment: The spatial windows from Camera 1 and Camera 2 consisting of the pair of spectral bands maximizing Eq. (24) now undergoes a window co-registration step. This includes horizontal shifting of the left window a certain number of pixels to the right. The amount the left window is shifted is the same number of pixels disparity calculated in the previous step. Thus, the two windows would now precisely overlap if the disparity calculated from PC Function Fitting step was completely accurate.

5. Fourier Domain Plane Fitting: However, as stated in Section 2.2.4, disparity calculation based on fitting of the phase-correlation function is not error-free and errors of around 0.1 pixels are common. Hence, a second matching procedure of the two now aligned windows is implemented. This is the Fourier Domain Plane Fit procedure based on determining the slope of the phase-difference matrix after unwrapping, as described in Section 2.2.4. Disparity calculation directly in the Fourier domain is known to be particularly effective considering small disparity variations, which now should be the case as the two windows are almost aligned. The total disparity of the considered window is the disparity calculated from the PC Function Fitting step, plus the disparity calculated from the Fourier Domain Plane Fitting step. After the Fourier Domain Plane Fitting step, a new spatial window from Mjolnir S-620 and Mjolnir V-1240 is loaded and the steps described so far are repeated, while the total disparity of the previously considered window is stored in the raw disparity map of the scene.

6. Post-Processing: To further refine the results, a post-processing step of the raw disparity map is implemented. This first includes disparity hole-filling, i.e interpolation of the disparity holes based on nearby disparity values that are not -1. The disparity map is then filtered and denoised by first an infinite impulse response (IIR) filter and then a total variation denoise filter. The idea behind these filters is to get smooth disparity variations while preserving edges. As the disparities are calculated for windows of several pixels and not individual pixels, the size and dimensions of the raw disparity map do not match the size and dimensions of the images employed in the stereo matching algorithm. The disparity map is, therefore, up-sampled to match the spatial dimensions of the two hyperspectral cubes. To avoid the final disparity map having a profile similar to a staircase after up-sampling, a low-pass filter with cutoff frequencies equal to the inverse of sampling rate in the horizontal and vertical directions is also implemented.

7. Output: The final output of the developed stereo matching algorithm is the final processed disparity map of the depicted scene.

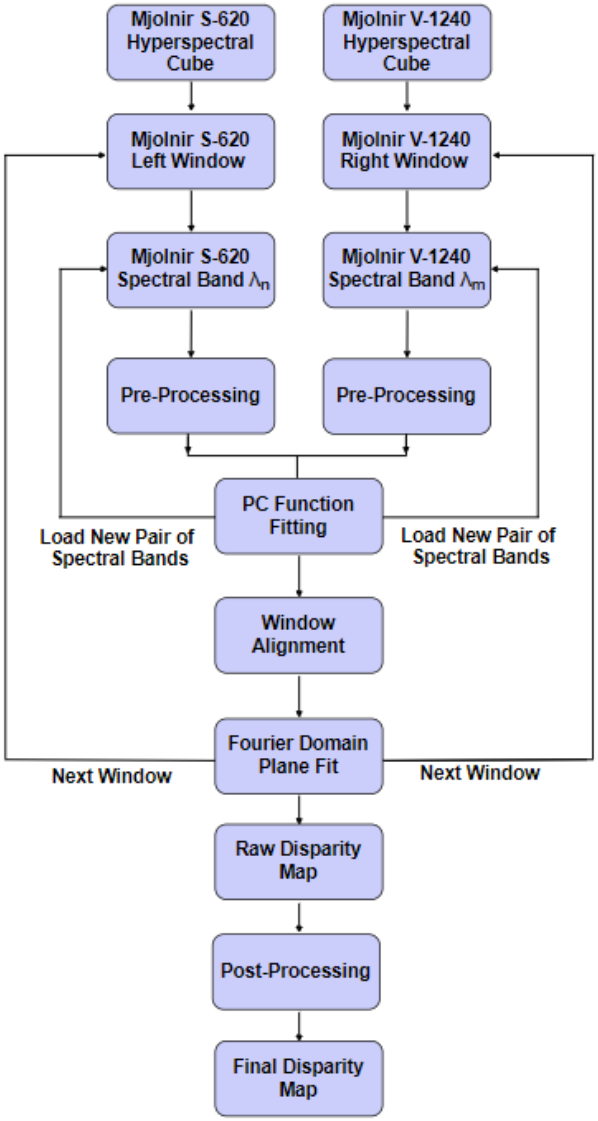


Figure 25: Flowchart of the developed stereo matching algorithm. The algorithm includes a pre-processing step, two disparity calculation steps as well as a post-processing step.

Yet to be touched upon is the size of the spatial windows and how they are extracted from the hyperspectral cubes. This is done in the simplest of manners, namely by extracting corresponding rectangles of width M and length N from the data captured by Camera 1 and Camera 2. The size of M and N, on the other hand, requires a bit more consideration. As explained in Section 2.2.5, it is desirable to perform stereo matching on large windows as long as the true disparity remains somewhat constant within a window. This limits the window size N in the along-track direction, as the UAV platform may change in altitude between scan lines, introducing a varying disparity to a constant plane. Thus, the size of N is kept somewhat limited and constant with N=20 pixels for all images employed in the developed model. The size of M, on the other hand, may be adjusted with fewer constraints. However, the changing elevation in the across-track direction of the scene must carefully be taken into consideration. This roughly divides the images into two groups: images with significant elevation variation and images with much less variation. Figure 23 presented in Section 4.1.1 above demonstrates both circumstances: the image resulting from flight line 1 with threes, bushes, and a gradually changing hill, and the two images resulting from flight lines 2 and 3 consisting of mostly the level ground. To preserve detail in the depicted scene, M=62 pixels for the images with a significantly changing elevation, while M=155 pixels for the images with much

less change in elevation.

The complete stereo matching algorithm is implemented in Python version 3.7.9, and all functionalities can be found in the Appendix Section A.

Table 2: Estimated depth based on a disparity range of 1-16 pixels. The presented values are utilized to determine a suitable disparity range for the proposed stereo matching algorithm and are calculated based on the stereo rig at hand. The same table was also utilised in [1].

Disparity [nr. of pixels]	Estimated depth, Z [m]
1	133.440
2	66.715
3	44.473
4	33.352
5	26.679
6	22.231
7	19.054
8	16.671
9	14.817
10	13.334
11	12.121
12	11.110
13	10.255
14	9.521
15	8.885
16	8.329

4.3 3D Terrain Mapping and Georeferencing

With the disparity of the entire scene determined, Eq. (2) is used to calculate the distance from the image plane to the considered pixel, denoted Z in Figure 11. To account for sub-pixel disparity values, the sensor model, providing the angles θ in Eq. (2), is interpolated.

For georeferencing of the depicted scene, Eq. (5), Eq. (6), Eq. (7) and Eq. (8), all presented in Section 2.2.3, are employed. INS data from both Camera 1 and Camera 2 are recorded for each captured scan line. The INS data may therefore be modeled as a continuously moving origin, where the 3D coordinates of each pixel in each scan-line can be triangulated based on this origin, as explained in Section 2.2.3. Here, only INS data from Camera 1 is utilized. This is simply because the number of frames recorded by Camera 1 matches the number of scan lines in the two hyperspectral cubes, whereas the number of frames recorded by Camera 2 is two times greater due to having double the frame rate. The flowchart presented in Figure 26 below, illustrates the process of determining the 3D coordinates of a pixel p, and the complete developed model. Here, the same notation as presented in Section 2.2.3 and utilized in Eq. (5), Eq. (6), Eq. (7) and Eq. (8) applies, where the bearing angle α is determined based on ± 90 degrees of the heading angle ρ. Earth's radius is approximated to be 6 371 km.

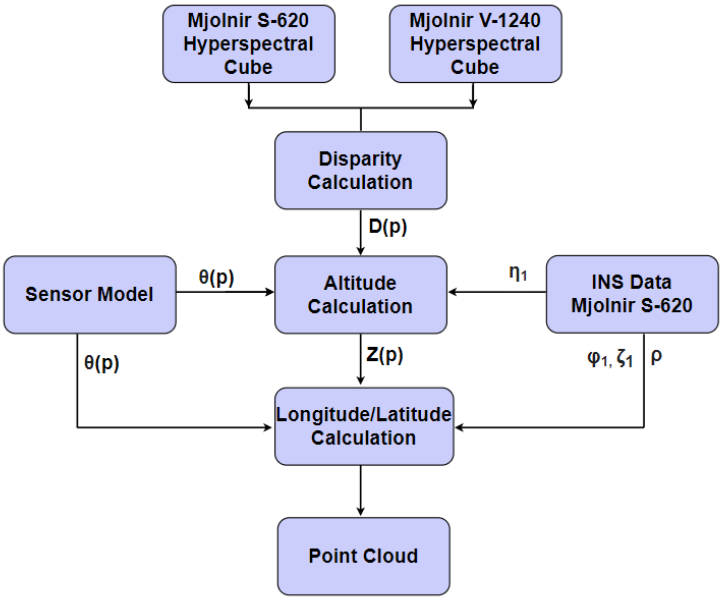


Figure 26: Flowchart of the complete developed model. The 3D coordinates of a pixel p are calculated based on its disparity, the sensor model, and the INS data provided by Camera 1.

4.3.1 Elevation Refinement - Trend Adjusting

Any variation in the UAV flight altitude for a specific flight line is expected to be compensated for in the disparity calculation step. Meaning if the UAV altitude decreases, the disparity calculated should increase correspondingly. However, very rapid and significant changes in the UAV flight altitude may introduce inaccuracy in the disparity calculations, causing deviations from the actual elevation of the terrain. This will particularly be evident for circumstances where the UAV undergoes a rapid increase or decrease in flight altitude within the along-track window size of N=20 pixels. The resulting deviations will typically be sudden increases or decreases in the elevation.

To address this, the elevation of the depicted scene is calculated in two steps. First, the elevation is determined by subtracting Z from the INS altitude, as explained in Section 2.2.3. Second, areas in the elevation profile of the terrain, in the along track-direction, fulfilling the condition where the rate of change has the opposite sign as the INS altitude and the calculated disparity, are modified with the rate of change of the INS altitude multiplied with some weight. In essence, this is the same as applying the shape of the INS altitude to the calculated elevation profile, and the weight regulates the degree to which it is applied. The idea behind this is to flatten out the areas of sudden increases or decreases in the calculated elevation of the terrain due to sudden changes in UAV flight altitude. It should be noted that this type of elevation refinement is most suitable to a level terrain profile and can not blindly be applied to significantly varying terrain.

The flowchart presented in Figure 26 as well as the trend adjusting algorithm are implemented in Python version 3.7.9, and all functionalities can be found in the Appendix Section B. Here, the function calculatePointCloud(disparity, elevation) has been reused from [1].

4.3.2 Point Cloud Visualisation and Analysis

The calculated point clouds are plotted and visualized in Global Mapper version 23.1, a geographic information system software provided by Blue Marble Geographics. However, prior to this, the point clouds are converted to .las file format, the industry-standard file format of LiDAR point

clouds. Global Mapper's "Fit Point Clouds" functionality is employed to combine the point clouds derived from the three different flight lines together into one point cloud of the entire scene. Additional smoothing of the point clouds is also performed in Global Mapper.

To evaluate the robustness of the developed model, the calculated point clouds resulting from the different UAV flight altitudes are compared to the LiDAR point clouds. Different types of terrain are considered, both simple and complicated. To quantify the results, the root means square error (RMSE) of the elevation associated with the derived point clouds is determined under the assumption of the LiDAR point clouds being ground truth. However, prior to this, the LiDAR point clouds and the point clouds resulting from the developed model are converted to elevation grids with Global Mapper's "Create Elevation Grid" tool. This is performed due to the LiDAR point clouds and the calculated point clouds having different point densities and the individual points are not precisely overlapping. With the point clouds represented as elevation grids, corresponding areas from the LiDAR data and the data from the developed model may easily be extracted and analyzed.

4.4 Benchmarking

Prior to applying the developed stereo matching algorithm described in Section 4.2.1 to the hyperspectral images of Losby shooting range, the considerations taken into account during the development of the algorithm are tested with both synthetic and spatially misregistered data.

4.4.1 Synthetic Data

Accurate sub-pixel disparity resolution, arguably being the most important aspect of the developed stereo matching algorithm, is tested by generating synthetic stereo pairs. It is desirable to map the performance of the two different fitting functions, i.e Eq. (22) and Eq. (23), and the Fourier-domain plane fit method, both individually and when implemented cooperatively as described in Section 4.2.1. The accuracy of the different approaches is tested on images with both constant and varying disparities, and their capacity to detect both very small and large disparities is uncovered. Additionally, the window size and how it affects the disparity precision of the different approaches are also looked into.

Radiometric differences and how they affect the stereo matching process, is evaluated by modifying the synthetic data with brightness differences, modeling what illumination differences in the stereo pair could resemble.

Periodic patterns, particularity known to complicate stereo matching, are also added to the synthetic data by a sinusoidal pattern running horizontally and should give additional insight into the robustness of the stereo matching approaches considered.

At last, to fully test the robustness of the different approaches to stereo matching, the generated stereo pair is contaminated with random artifacts modeled as Poisson distributed noise according to Eq. (25).

$$P(x) = \frac{\beta^x}{x!} e^{-\beta} \tag{25}$$

Here, β, the expected rate of occurrences, can be incrementally increased, adding more noise to the images to really test the robustness of the different stereo matching approaches.

The off-shelf intensity-based semi-global block matching (SGBM) algorithm provided by OpenCV is also implemented and executed on synthetic data. The algorithm was briefly introduced in Section 2.2.4. In short, the SGBM algorithm is intensity-based and minimizes the global energy function, Eq. (9), by dynamic programming through Eq. (11), Eq. (12) and Eq. (13), and displays a somewhat different approach to stereo matching. It is therefore interesting to uncover how this approach to stereo matching holds up against the phase-based approaches implemented in the

developed stereo matching algorithm. A more detailed description of OpenCV's SGBM algorithm can be found in [30] and [1]. The synthetic data is generated in Python 3.7.9 and all functionalities can be found in the Appendix Section C, while the implementation of the SGBM algorithm is located in the Appendix Section A and is the same implementation used in [1].

4.4.2 Spatially Misregistered Data

Hyperspectral data not corrected for spatial distortions exhibits horizontal disparity due to the difference in keystone between spectral channels of the same hyperspectral cube. This type of disparity is most significant for spectral channels located far away from each other in the electromagnetic spectrum, and the spatial pixels located towards the beginning and end of the scan line. Figure 27 presented below, illustrates the keystone effect of the hyperspectral cube resulting from Camera 1 for the first pixel of its scan line. The plot has a similar shape to the one presented in Figure 7 and showcases how the difference in keystone across the spectral channels results in horizontal disparity.

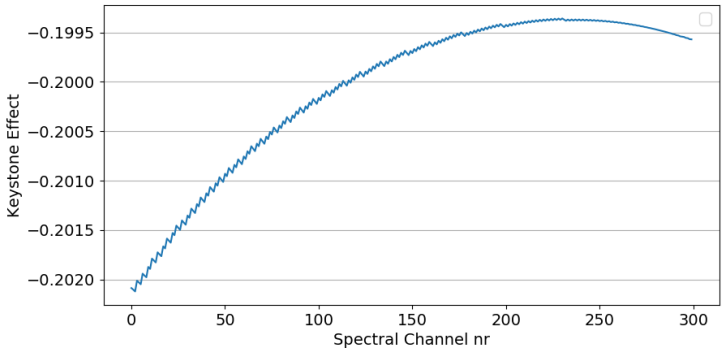


Figure 27: Keystone effect resulting from Camera 1 for the first spatial pixel of its scan line. The difference in keystone between spectral channels results in horizontal disparity.

Estimating the disparity resulting from the differences in keystone between spectral channels is not only beneficial in terms of testing the different stereo matching approaches on actual hyperspectral data, but the disparity profiles due to this effect are also gradually changing along each scan line. They, therefore, serve as a suitable representation of what the disparity profile of a natural and gradually changing terrain may look like. The accuracy of these disparity estimates may also be quantified as the keystone effect for each spectral band of Camera 1 and Camera 2 has been quantified by NEO, and are available in the keystone maps of Camera 1 and Camera 2. The disparity due to spatial misregistration is typically very small and is therefore estimated with the Fourier-domain plane fit method.

5 Results and Discussion

5.1 Synthetic and Spatially Misregistered Data

The performance of the different stereo matching approaches is first demonstrated in terms of accuracy, robustness, and how well the disparity of spatially misregistrated data can be estimated. The stereo matching techniques considered are PC function fitting, with both Gaussian and sinc functions, Fourier-domain plane fitting, and the SGBM algorithm provided by OpenCV. The results are first presented and discussed consecutively without context to the developed model. Section 5.1.4, however, addresses the relevance of the results derived from the synthetic and spatially misregistered date to the developed model, in terms of expectations and measures to improve the performance, and the key points from Section 4.4 are revisited.

5.1.1 Stereo Matching Accuracy

Scene of Constant Disparity

A scene of constant disparity is emulated by employing the image presented in Figure 28, and a second identical image, incrementally shifted 0-7 pixels wrt the initial image as input to the different matching approaches. Corresponding windows of 62×20 pixels^2, randomly distributed across the two images, are extracted and the disparity determined over 700 samples, with an incrementally increasing disparity of 0.01 pixels between each sample.

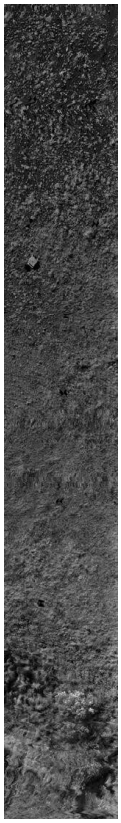


Figure 28: Image used to determine the accuracy of the different stereo matching techniques. A second identical image shifted 0-7 pixels wrt the initial image is also employed.

The plots presented below in Figure 29 and Figure 30 showcase the different stereo matching approaches and their estimated disparities for an incrementally increasing true disparity. In Figure 29, PC Initial Estimate is calculated based on the initial pixel maximizing the PC function, and

will always be an integer, as explained in Section 2.2.4 and seen from its staircase-shaped plot. Further, evident from Figure 29 is how both the PC Gaussian fit and PC Sinc fit is more accurate than the SGBM algorithm. This is especially clear from the zoomed-in plot to the right in Figure 29 where both the PC Gaussian fit and the PC Sinc fit almost overlap the true disparity. However, for the first 100 samples, in the disparity range of 0-1 pixels, the PC Gaussian fit deviates somewhat from the true disparity value.

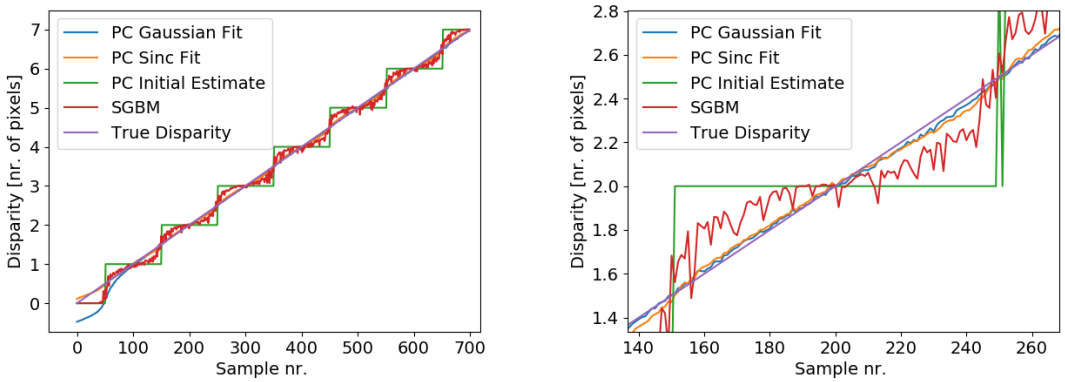


Figure 29: Calculated disparity based on PC function fitting and the SGBM algorithm for an incrementally increasing disparity. PC Gaussian fit demonstrates the highest accuracy, neglecting the first 100 samples.

The plots presented below in Figure 30, showcase the performance of the Fourier-domain plane fit approach to stereo matching for an incrementally increasing true disparity, in the same manner as presented in Figure 29. Evident from the left plot is how the calculated disparities resemble a shape similar to a staircase. This suggests that the Fourier-domain plane fit approach to stereo matching becomes inaccurate for sub-pixel disparity values. Further inspection reveals how the saw-tooth shape of the phase-difference matrix becomes less predictable for sub-pixel disparities. This is illustrated in the plots presented in Figure 31, where the left plot exemplifies the phase-difference matrix given a disparity of 1 pixel, while the right plot illustrates the same given a disparity of 1.3 pixels. Due to the phase-difference matrix deviating from a perfect saw-tooth signal, its slope after unwrapping, used in Eq. (19), is also altered.

However, by only considering the first third of the samples in the phase-difference matrix, the area introducing deviations to the saw-tooth signal is discarded, while still keeping enough samples to accurately estimate the slope. This approach is showcased to the right in Figure 30. At first glance the estimated disparities may appear inaccurate, however, this is mostly true when the disparities exceed 3 pixels. For smaller disparities, especially below 1 pixel, this approach achieves very high accuracy, as can be seen for the first 100 samples. This modification of the Fourier-domain plane fit method is applied to all further calculations.

Table 3 presented below, summarizes the accuracy of the different methods for a scene of constant disparity. Evident from Table 3 is how the Fourier-domain plane fit approach achieves the lowest RMSE, given disparities smaller than 0.5 pixels. With regards to the PC fitting approaches, the Gaussian fit accomplishes the lowest RMSE, given disparities larger than 1 pixel with the sinc fit not far behind. All phase-based approaches, neglecting the initial estimate of the PC function, yielded more accurate results than the intensity-based SGBM algorithm.

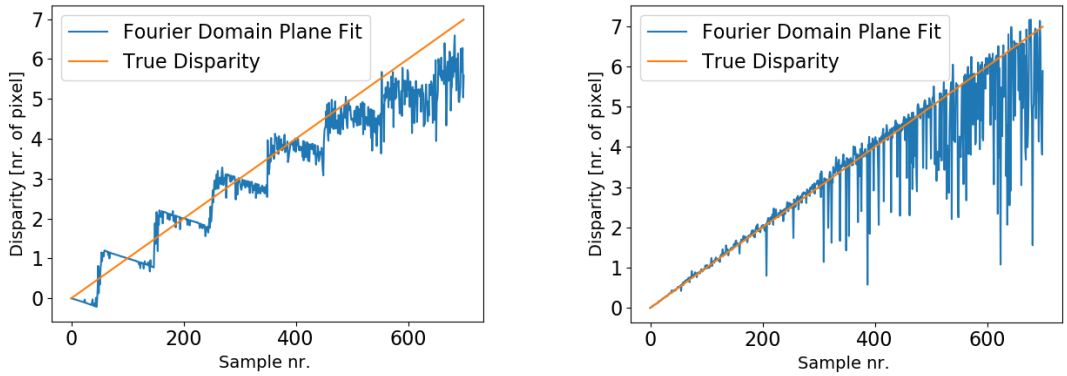


Figure 30: Calculated disparity based on Fourier-domain plane fitting. The left plot is calculated based on the entire phase-difference matrix and struggles with sub-pixel disparities. The right plot achieves very high accuracy for small disparities, by only considering the first 20 samples of the phase-difference matrix.

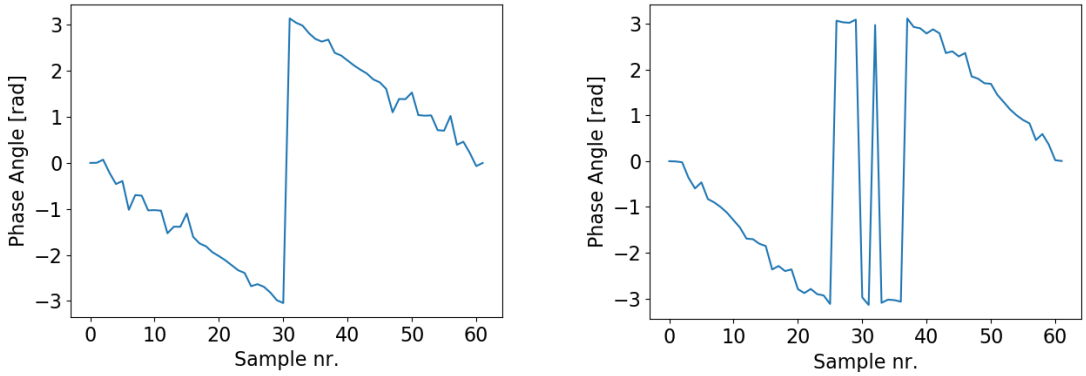


Figure 31: Phase-difference matrix corresponding to a disparity of 1 pixel (left) and 1.3 pixels (right). Sub-pixel disparities cause the phase-difference matrix to deviate from an ideal saw-tooth shape.

Table 3: RMSE of the different stereo matching approaches. The Fourier-domain plane fit technique achieves the lowest RMSE score given disparities smaller than 0.5 pixels. For PC function fitting, the Gaussian fit yields the lowest RMSE score given disparities larger than 1 pixel. All phase-based approaches, excluded the initial PC estimate, yielded more accurate estimates than the intensity-based SGBM approach.

Method	RMSE [nr. of pixels]
PC Initial Estimate	0.289
PC Gaussian Fit	0.149
PC Gaussian Fit (last 600 samples)	0.026
PC Sinc Fit	0.039
PC Sinc Fit (last 600 samples)	0.039
SGBM	0.152
Fourier Domain Plane Fit (first 50 samples)	0.012

Scene of Varying Disparity

To fully demonstrate the benefits of calculating the disparity in two steps, the approach explained in Section 4.2.1, images with varying disparity levels are used for input. The image presented to the left in Figure 32 is used as input with a second image shifted 3.67 pixels horizontally, where rectangle 1 is shifted additional 0.19 pixels, rectangle 2 additional 0.27 pixels, and rectangle 3 additional 0.12 pixels. The ground truth disparity map is presented to the right in Figure 32.

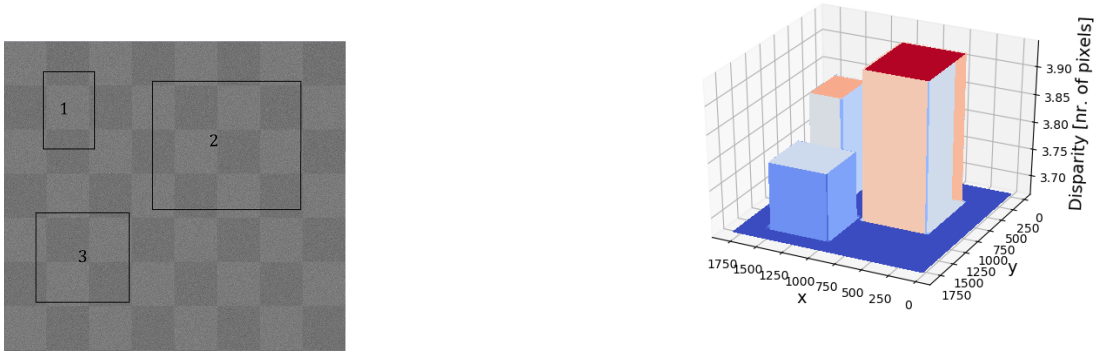


Figure 32: Image of varying disparity used as input for stereo matching (left). An identical image is shifted 3.67 pixels horizontally with rectangle 1 shifted additional 0.19 pixels, rectangle 2 0.27 pixels, and rectangle 3 0.12 pixels. The ground truth disparity map is presented to the right.

The disparity is estimated with PC function fitting individually, with a Gaussian fitting function, the two-step approach based on PC function fitting and the Fourier-domain plane fit, and the SGBM algorithm. The window size employed is 62×20 pixels^2. The disparity map resulting from the SGBM algorithm is presented to the left in Figure 33, while the disparity map resulting from the two-step approach based on PC function fitting combined Fourier-Domain Plane Fitting, is presented to the right. Here, the disparity map presented has been post-processed with an IIR filter and a total variation denoise filter, both part of the post-processing step of the developed stereo matching algorithm, as explained in Section 4.2.1. The disparity map resulting from PC function fitting individually showed very little visual deviation from the disparity map presented to the right in Figure 33, and is presented in Figure 74 located in the Appendix Section D.

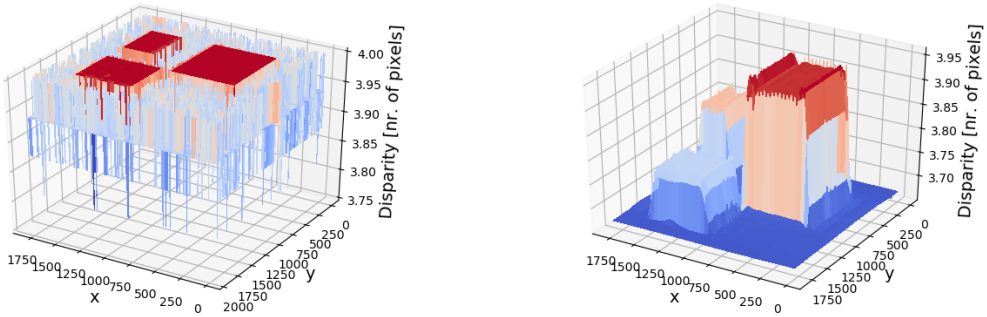


Figure 33: Disparity map resulting from the SGBM algorithm (left), and disparity map resulting from PC function fitting combined with Fourier-domain plane fitting (right), the same approach employed in the complete developed stereo matching algorithm.

From Figure 33 it is evident how the SGBM algorithm struggles with small disparity variations. It is also evident how the lack of distinct features in the images used as input to the algorithm leads to a significant amount of noise in the disparity map calculated by the SGBM algorithm, most

likely due to the intensity-based nature of the algorithm. The disparity map resulting from PC function fitting combined with Fourier-domain plane fitting, on the other hand, is a much more accurate representation of the ground truth disparity map presented in Figure 32. However, when compared to the ground-truth disparity map, sharp edges are somewhat smoothed out and the areas of changing disparity levels are prone to errors.

The RMSE values of the calculated disparity maps are presented below in Table 4. Apparent from Table 4 is how combining PC function fitting with Fourier-domain plane fitting leads to a slight increase in accuracy, here corresponding to approximately 0.01 pixels, when compared to only PC function fitting. It is also evident how this phase-based approach to stereo matching is over 10 times more accurate compared to the intensity-based SGBM algorithm. Filtering also reduces the errors slightly.

Table 4: RMSE values of the different disparity maps based on the image presented to the left in Figure 32. The phase-based approach of combing PC function fitting with Fourier-domain plane fitting yielded the most accurate results.

Method	RMSE [nr. of pixels]
PC Fitting Combined With Fourier-Domain Plane Fitting	0.0224
PC Fitting Combined With Fourier-Domain Plane Fitting (filtered)	0.0206
PC Gaussian Fitting	0.0314
PC Gaussian Fitting (filtered)	0.0292
SGBM	0.2448

It is also of interest to uncover how the phase-based stereo matching approaches respond to even smaller window sizes. The same image as presented in Figure 32 is used as input, and the disparity maps are calculated with an incrementally decreasing window size. The results are presented below in Table 5, where all results have been filtered with an IIR filter and total variation filter. From Table 5 it is evident how the RMSE error increases with decreasing window size. The RMSE increase is most significant for PC Gaussian fitting individually, highlighting the benefit of implementing an additional Fourier-domain plane fit step to the stereo matching algorithm for smaller window sizes. Further, the increase in RMSE value is most significant when the width, M, of the window size is reduced.

Table 5: RMSE values of the disparity maps calculated with an incrementally decreasing window size. The RMSE values increase as the window size decreases, however, most notably for the disparity maps calculated based on PC function fitting individually.

Method	Window Size [pixels^2]	RMSE [nr. of pixels]
PC Fitting Combined With Fourier-Domain Plane Fitting	62x20	0.0206
PC Gaussian Fitting	62x20	0.0314
PC Fitting Combined With Fourier-Domain Plane Fitting	62x10	0.0247
PC Gaussian Fitting	62x10	0.0344
PC Fitting Combined With Fourier-Domain Plane Fitting	31x10	0.0513
PC Gaussian Fitting	31x10	0.0781
PC Fitting Combined With Fourier-Domain Plane Fitting	31x5	0.0654
PC Gaussian Fitting	31x5	0.1256

To fully test the capacity of the stereo matching approach combing PC function fitting and Fourier-domain plane fitting, images with very small disparities are employed. The image presented in Figure 32 is once again employed, however, now the rectangles of the second identical image are shifted 0.02, 0.04, and 0.06 pixels. The disparity map resulting from PC function fitting individually is presented to the left in Figure 34, while the disparity map resulting from combining PC function fitting with Fourier-domain plane fitting is presented to the right. Evident from Figure 34 is how the disparity maps now become significantly smeared out. However, this effect is much more apparent in the disparity map calculated from PC function fitting individually, again

highlighting the increase in performance when PC function fitting is combined with Fourier-domain plane fitting.

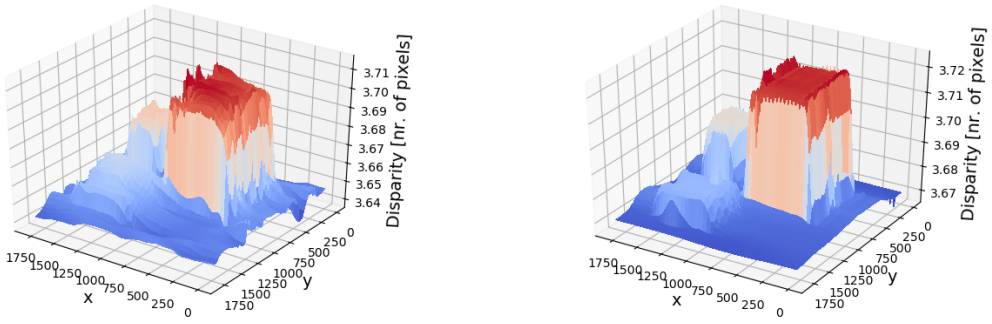


Figure 34: Disparity map resulting from PC function fitting individually (left), and PC function fitting combined with Fourier-domain plane fitting (right), for very small disparity variations.

5.1.2 Stereo Matching Robustness

Radiometric differences are emulated by brightness enhancing one of the images in the stereo pair. The image presented in Figure 28 is once again utilized, with a second image shifted 5.63 pixels horizontally. The disparity is determined based on corresponding windows of 62×20 pixels^2 randomly distributed across the two images, with an incrementally increasing brightness enhancement of the second image in the stereo pair. The calculated disparities for increasing brightness differences in the stereo pair are presented to the left in Figure 35. From the plot, it is clear how the SGBM algorithm struggles with brightness differences, while the PC function fitting approaches remain unaffected with little deviation from the true disparity value. However, it should be noted how these calculations are based on one of the images simply being multiplied with a brightness enhancement factor. Thus, the spatial frequency content of the image is not altered, and its DFT remains very similar. For cases where the two images have very different spatial frequency content due to illumination differences, such as dark spots in one image appearing very bright in the other, the performance of the PC function fitting approaches would most likely also decrease significantly. The image used for input after being brightness enhanced by a factor of 2.75 is presented in Figure 37.

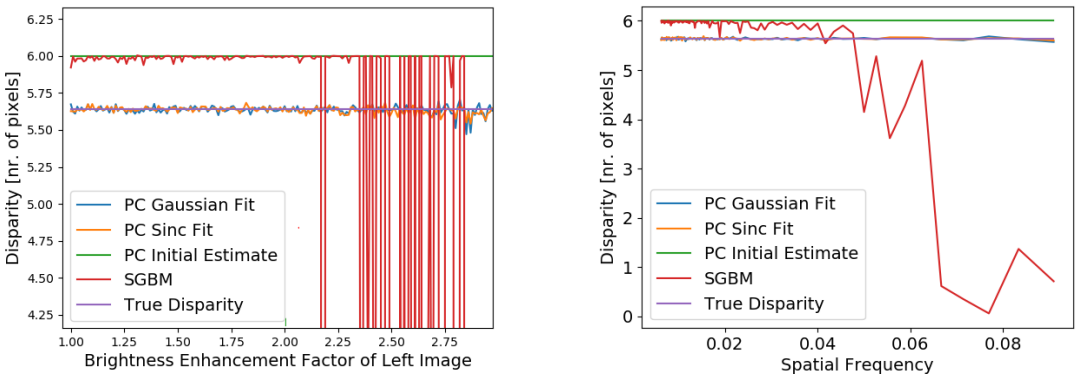


Figure 35: The effect of brightness enhancement (left) and periodic patterns (right). The SGBM algorithm struggles with input images of different brightness and periodic patterns running horizontally, while the PC function fitting calculations showed no significant decrease in performance.

Periodic patterns running horizontally are also added to the image presented in Figure 28 to test

the robustness of the different stereo matching approaches. The disparity is calculated in the same manner as described in the paragraph above, and the spatial frequency of the sinusoidal pattern incrementally increased. The results are presented to the right in Figure 35. Here it is evident how the SGBM algorithm struggles with increasing frequencies, especially apparent when the spatial frequency approaches the inverse of the window size. The PC function fitting approaches, on the other hand, showed no apparent reduction in performance, also when the spatial frequency was increased beyond what is presented in Figure 35. The input image, when applied a sinusoidal pattern of frequency 0.08 oscillations/pixel is shown in Figure 37.

At last, to fully test the robustness of PC function fitting, the images used as input are contaminated individually with Poisson distributed shot noise, according to Eq. (25). The results, for an incrementally increasing β are presented below in Figure 36. Compared to the plots presented in Figure 35, the PC function fitting approaches start to deviate more from the true disparities. Considering how random noise components are added individually to the input images, this is also expected as the spatial frequency components of the two images, used to calculate the PC function, deviate from each other as β grows. Evident from the right plot, illustrating the same as the left plot, excluding the results from the SGBM algorithm and the initial PC estimate, is how PC function fitting with a Gaussian fit tends to be more accurate than the sinc fit. However, also evident from the right plot, is the existence of instances where the Gaussian-shaped function could not be fitted to the data, resulting in a disparity of -1, and the PC function fitting with a sinc function provided an accurate estimate. Once again, the phase-based approaches to stereo matching proved to be more robust than the intensity-based SGBM algorithm. The input image when applied Poisson distributed noise where β=1000 is presented in Figure 37.

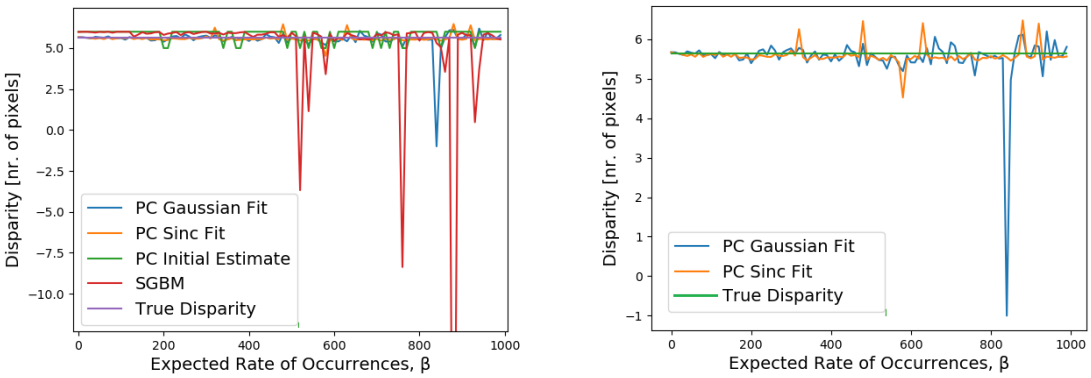


Figure 36: Calculated disparities for an increasing amount of noise added to the input images. The right plot illustrates the same as the left plot, excluding the results from the SGBM algorithm and the initial PC estimate.

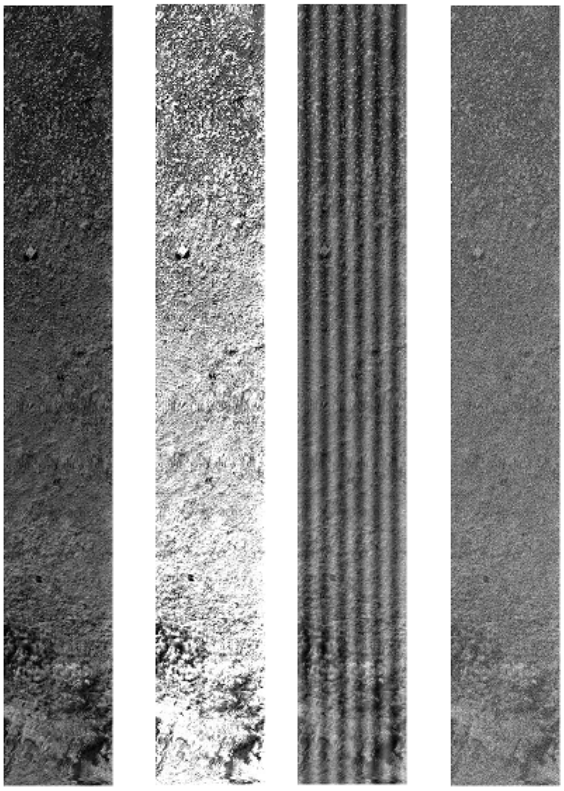


Figure 37: Images used as input to evaluate the robustness of the stereo matching approaches. The first image is the same as the one presented in Figure 28, the second is after being brightness enhanced by a factor of 2.75, the third is after a sinusoidal pattern is applied, while the fourth is after being contaminated with shot noise where β=1000.

5.1.3 Spatially Misregistered Data

The horizontal disparity resulting from keystone distortions is calculated by using the center pixel of the scan line as the origin, the pixel where the keystone is approximately zero across all spectral channels. The disparity is then determined as the difference in keystone between two spectral channels, at a given pixel, plus the contributions from the pixels located closer to the center pixel within the scan line. This is executed for the pixels to the left of the center pixel and the right of the center pixel independently. The disparity profile due to keystone differences between spectral channels is presented below in Figure 38, and is calculated from the keystone map of Camera 1, where spectral channel nr. 0 and 10 and spectral channel nr. 0 and 20 are considered.

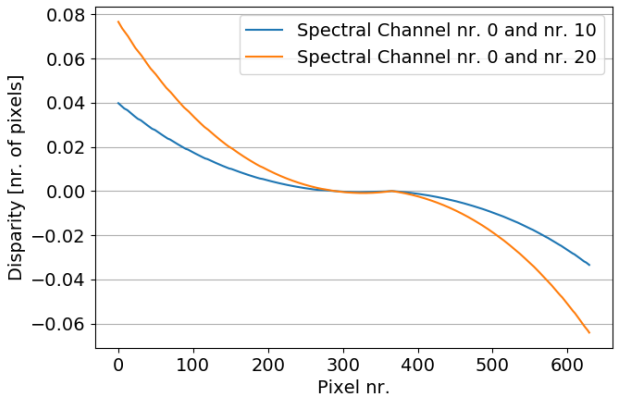


Figure 38: Disparity profile due to keystone differences between spectral bands.

Modelling of Gradually Changing Elevation

The disparity profiles presented above in Figure 38 illustrate what the disparity profile of a gradually changing terrain may look like, where each of the pixels are shifted differently. As the resulting disparities are very small, the Fourier-domain plane fitting approach to stereo matching is considered.

Spatially misregistered data from Camera 1 captured at a UAV flight altitude of 20 m above ground is used as input. An image of the spatially misregistered scene from the first band at 970 nm is presented in Figure 75 located in the Appendix Section D, and is very similar to the image presented in Figure 28. Horizontal disparity resulting from keystone differences between spectral channel nr. 0 and 10 as well as spectral channel nr. 0 and 20, are considered. The disparity is calculated with the Fourier-domain plane fit approach, first with a window size of 62×20 pixels^2. The results are presented below in Figure 39, where spectral channel nr. 0 and 10 are considered to the left and spectral channel nr. 0 and 20 are considered to the right.

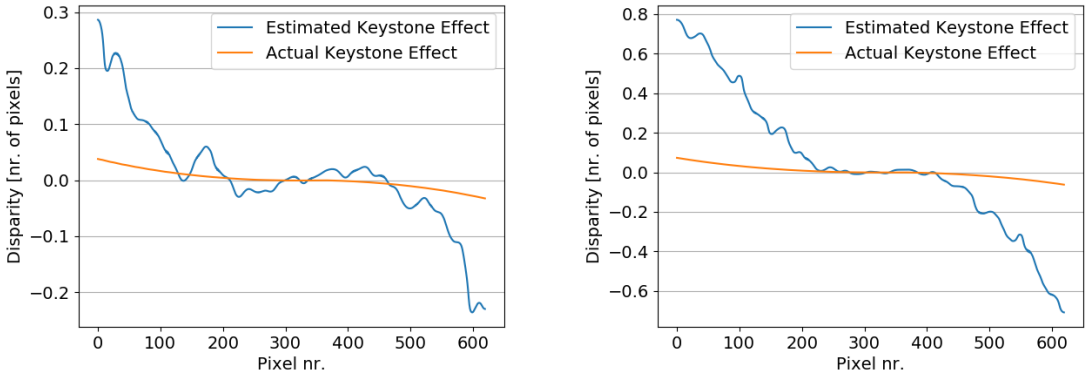


Figure 39: Estimated and actual horizontal disparity resulting from keystone. Spectral channel nr. 0 and 10 from Camera 1 are considered to the left, while spectral channel nr. 0 and 20 are considered to the right. Calculated with a window size of 62×20 pixels^2.

From Figure 39 it is clear how the calculations become significantly prone to errors for the areas where the disparity increases most rapidly. How the window size affects the calculations is revealed by changing the dimensions of the window size to 20×62 pixels^2. The results from using a smaller window size in the along scan line direction are presented below in Figure 40, where spectral channel nr. 0 and 10 are considered to the left and spectral channel nr. 0 and 20 are considered to the right.

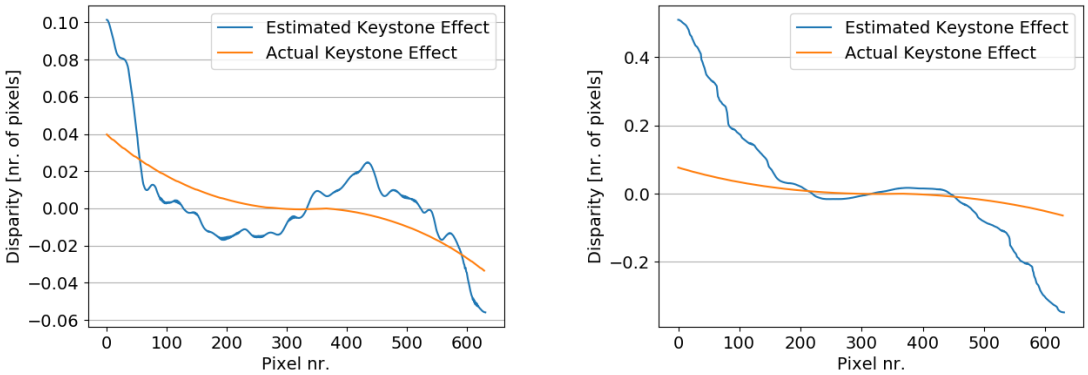


Figure 40: Estimated and actual horizontal disparity resulting from keystone. Spectral channel nr. 0 and 10 from Camera 1 are considered to the left, while spectral channel nr. 0 and 20 are considered to the right. Calculated with a window size of 20×62 pixels^2.

When comparing Figure 39 and Figure 40 it becomes clear how a smaller window size in the direction of varying disparity is necessary to preserve the details of a gradually changing disparity. To further exemplify the effect of reducing the window size to preserve detail, the same calculations are repeated, however, now with a window size of only 4 pixels in the along scan line direction. The results are presented below in Figure 41, and showcase even more accurate calculations.

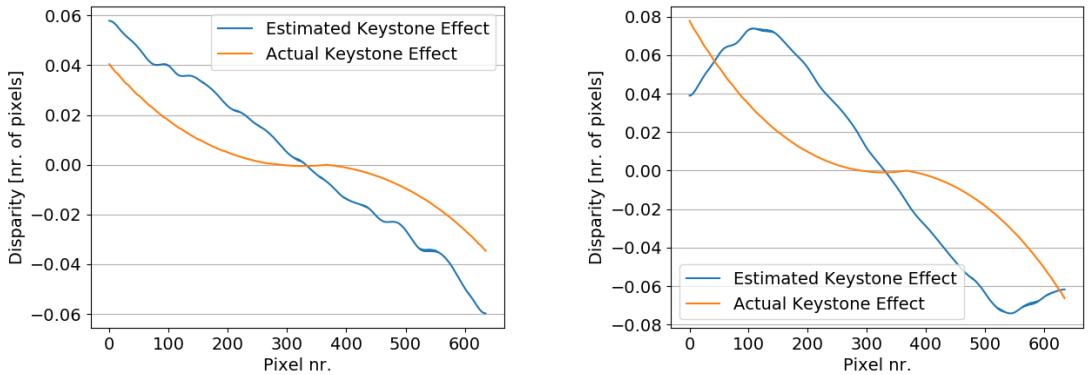


Figure 41: Estimated and actual horizontal disparity resulting from keystone. Spectral channel nr. 0 and 10 from Camera 1 are considered to the left, while spectral channel nr. 0 and 20 are considered to the right. Calculated with a window size of 4×62 pixels^2.

The results presented in Figure 39, Figure 40 and Figure 41 suggest that the the disparity calculations for a terrain of changing disparity is more prone to errors, compared to terrain of constant disparity, as was considered in Section 5.1.1. The errors also appear to be largest in the areas where the disparity changes more rapidly. However, despite a decrease in accuracy, the trends of the disparity profiles are still present in the majority of the calculations, mainly those with a smaller window size in the direction of changing disparity.

5.1.4 Remarks and Considerations

In terms of accurate sub-pixel disparity resolution, the results revealed in Section 5.1.1, demonstrate the advantage of phase-based stereo matching in comparison to the intensity-based SGBM algorithm. It was also demonstrated an increase in accuracy by performing the disparity calculations in two steps, first by PC function fitting and then Fourier-domain fitting, also implemented in the developed stereo matching algorithm. Fitting the PC function with a Gaussian function proved to yield to most accurate results, given disparities above 1 pixel. Based on the values presented in

Table 2, errors in the disparity calculations resulting from disparities below 1 pixel should not be an issue when the algorithm is implemented on real hyperspectral data of the imaged scene. Further, the importance of the Fourier-domain plane fitting step became particularly evident for decreasing window sizes and very small disparity variations. However, this was only evident when the first third of the phase-difference matrix was utilized, a consideration also added to the developed stereo matching algorithm. Additionally, it was also demonstrated how this Fourier-domain plane fit approach to stereo matching only was accurate for very small disparities, ideally below 0.5 pixels. To mitigate errors from larger disparities, only disparity values calculated to be less than 0.2 pixels are accepted from this step of the developed stereo matching algorithm. If the calculated disparity is greater, only the value derived from the PC function step will be considered. However, given the accuracy of PC function fitting presented in Section 5.1.1, the disparity between the left and right window should be less than 0.2 pixels after the window-alignment step of the developed stereo matching algorithm.

With regards to window size, it was highlighted in Section 4.2.1 how it is desirable to maintain a rather limited window size in the along-track direction of the developed stereo matching algorithm, due to the variation in UAV flight altitude. The results presented in Table 5 suggest that high accuracy can be maintained even with a small along-track window size. Additionally, the across-track window size of 62 pixels, utilized for more complicated areas of the scene, should be more than enough pixels to provide accurate calculations.

Concerning the robustness of the developed stereo matching algorithm, radiometric differences and periodic patterns should not be an issue, given that the spatial frequency content of the stereo pairs remains somewhat equal. This is attempted to ensure by primarily considering the overlapping wavelength region of Camera 1 and Camera 2 at 970 – 1000 nm in the developed stereo matching algorithm. However, deviations in the spatial frequency content of the stereo pairs can not be completely neglected, the cameras employed are after all different with separate spectral resolutions. Despite this, evident from Figure 36, is how PC function fitting still proves to be robust even when random artifacts are added separately to the stereo pair, altering their spatial frequency content. Additionally, the approach of fitting the PC function to a sinc function if the Gaussian fit fails should also improve the robustness of the stereo matching process, based on Figure 36.

For terrain of varying disparity, the accuracy of the developed model is expected to decrease, based on the results presented in Section 5.1.3. The error is expected to be largest in the direction where the window size is largest, the across-track direction in the developed model. However, despite some inaccuracies, trends in the elevation profile of a gradually changing terrain should be present in the calculations.

Using the results presented in Table 4 and a disparity error of approximately 0.02 pixels, Table 6 presented below, showcases the maximum expected elevation accuracy obtainable by the developed stereo matching algorithm, for UAV flight altitudes of 20 m, 40 m, and 60 m.

Table 6: Maximum expected elevation accuracy of the developed model. The values are derived from Eq. (2), the stereo rig at hand, and a disparity error of approximately 0.02 pixels.

UAV Flight Altitude [m above ground]	Elevation Accuracy [m]
20	0.06
40	0.24
60	0.55

5.2 Hyperspectral Stereoscopic Data

In the following section, the results derived from real hyperspectral stereoscopic data of the imaged scene are presented. First, the results from each of the UAV flight altitudes are presented and discussed separately. Then follows a comparison of the results from the different UAV altitudes, before the entire system, including the developed model and the imaging system, are evaluated. It should be noted that, if not stated otherwise, all error estimates are based on the LiDAR data being ground truth and a constant UAV flight altitude. Additionally, all elevation values presented are wrt the same reference.

5.2.1 20 m UAV Flight Altitude

The imaged scene captured by Camera 1 at 986.36 nm, given a UAV flight altitude of 20 m above ground, is presented below in Figure 42. In Figure 42, all three flight lines are showcased and the images have been brightness enhanced for better visualisation.

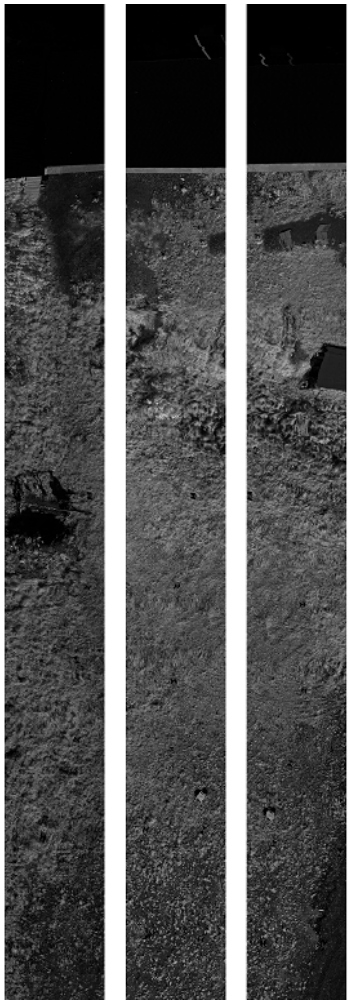


Figure 42: Imaged scene captured by Camera 1 from 20 m UAV flight altitude. The images have been brightness enhanced for better visualisation.

For stereo matching, the developed algorithm described in Section 4.2.1 is employed. The images resulting from a UAV flight altitude of 20 m mostly consist of the level ground and the top of the house and exhibit little elevation variations. A larger window size in the across-track direction may therefore be employed and the window size utilized is 155×20 pixels^2 for all flight lines. The disparity range is determined based on expected disparity values and is set to 5-8 pixels. The raw

disparity profile in the along-track direction resulting from approximately the center of the scene captured in flight line 2, is presented below in Figure 43.

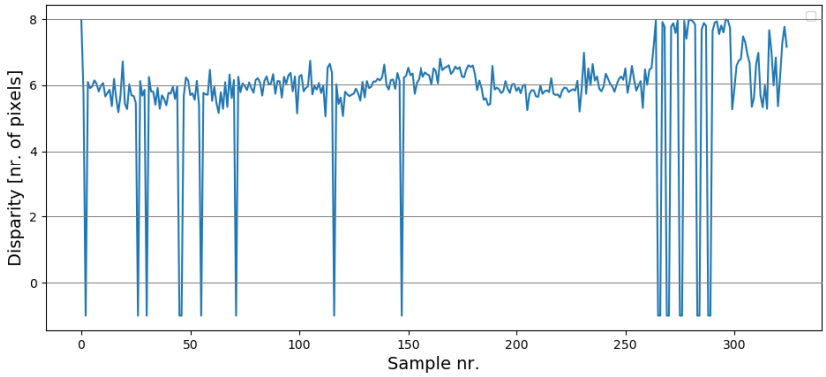


Figure 43: Raw disparity profile in the along-track direction based on images captured at 20 m UAV flight altitude.

From Figure 43 it is clear how the disparity mainly varies between two values, the disparity associated with the almost level ground and the disparity associated with the top of the depicted house. The disparity holes, the samples in Figure 43 with disparity value -1, are the areas where neither Eq. (22) nor Eq. (23) could be fitted to the PC function or the calculated disparity was estimated to be outside the disparity range. The top of the house is particularly prone to the latter as its ground true disparity is most likely close to 8 pixels. The corresponding processed and refined disparity profile presented in Figure 43 is showcased in Figure 44 below.

The disparity profile presented in Figure 44 displays a few of the same tendencies uncovered in Section 5.1.1, particularly how sharp edges and abrupt changes in the disparity become smeared-out, resulting in a reduction of detail. This is apparent between samples 250 and 300 covering the overlap between the two distinct disparity levels of the level ground and the top of the house. The disparity does not change instantaneously but is rather spread over several samples.

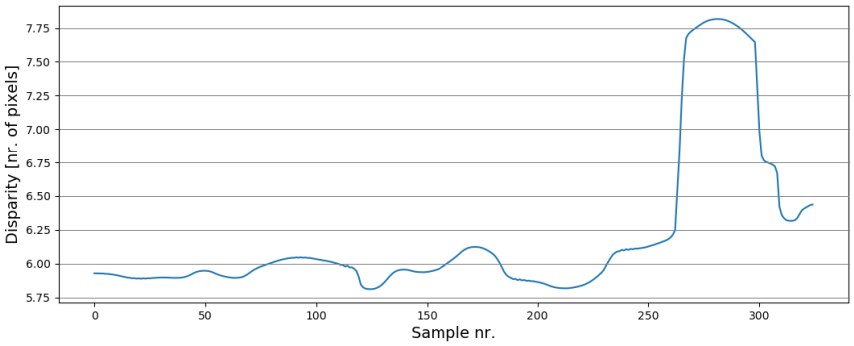


Figure 44: Processed and refined disparity profile resulting from the raw disparity profile presented in Figure 43.

The benefit of employing several spectral channels from Camera 1 and Camera 2 and using Eq. (24) to determine the most suitable pair for stereo matching, becomes evident when comparing the disparity profile resulting from the entire overlapping wavelength region of Camera 1 and Camera 2, to the disparity profile derived from only two spectral channels. This is presented below in Figure 45, where two along-track disparity profiles from flight line 3 are illustrated. Here, the left disparity profile is the result of limiting the developed stereo matching algorithm to only two spectral channels, the 199th band at 986.36 nm from Camera 1 and the 4th band at 985.35 nm from

Camera 2, while the right disparity profile is from the entire overlapping region of 970 - 1000nm.

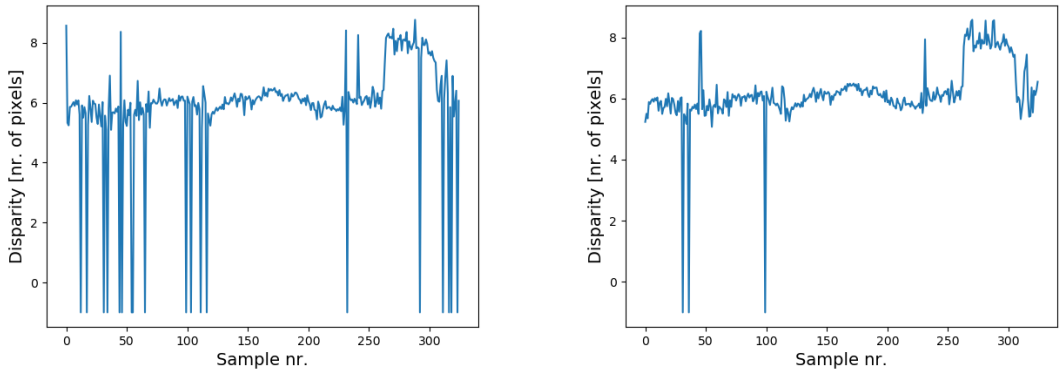


Figure 45: The disparity profile derived from one pair of spectral channels from Camera 1 and Camera 2 (left), and the disparity profile resulting from the entire overlapping wavelength region of Camera 1 and Camera 2. By employing several spectral channels for stereo matching, the number of disparity holes decreases significantly.

Evident from Figure 45 is how the number of disparity holes decreases significantly by employing several spectral channels and implementing a disparity optimization step by calculating PC_{score} in the developed stereo matching algorithm. This is particularly true in low illuminated areas of the depicted scene. One example of such an area is located behind the depicted house in Figure 21, corresponding to the very top of Figure 42. In these areas, the shape of the PC function may vary considerably between spectral channels, exemplified below in Figure 46. Here, the PC function is plotted based on two corresponding spatial windows, however, two different pairs of spectral channels are considered. After PC function fitting, the left plot resulted in a disparity of 5.88 pixels, while neither Eq. (22) nor Eq. (23) could be fitted to the right plot resulting in a disparity hole. Thus, despite the spectral channels within the overlapping wavelength region may be containing very similar reflectance profiles, bad illuminated areas of the depicted scene benefit notably from including multiple spectral channels from Camara 1 and Camera 2.

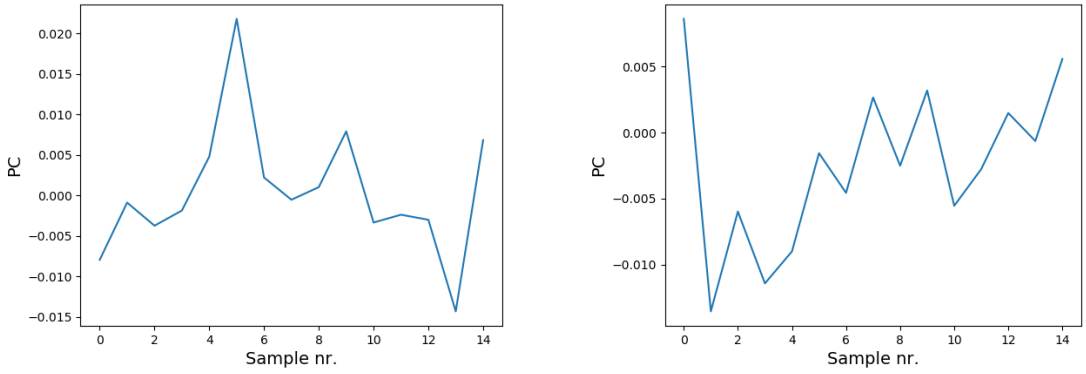


Figure 46: PC functions for two corresponding areas, however, for different pairs of spectral channels. The left plot resulted in a disparity of 5.88 pixels, while the right plot resulted in a disparity hole, highlighting the importance of including multiple spectral channels in the developed stereo matching algorithm.

At lower UAV flight altitudes, such as 20 m above ground, it is expected to see a significant correlation between the variations in the along-track disparity profiles and variations in UAV flight altitude, particularly for areas with sudden and significant variations in UAV altitude. Below, plotted in Figure 47, is the along-track disparity profile resulting from the right of the scene captured in flight line 1 and the corresponding INS altitude data.

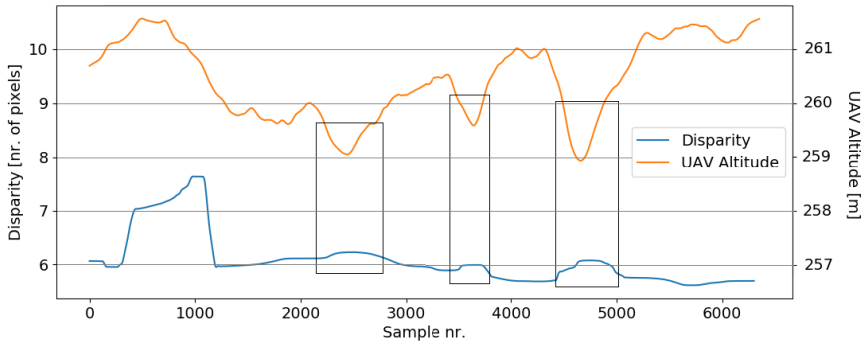


Figure 47: Along-track disparity profile based on the scene captured in flight line 1 and the corresponding INS altitude data. Sudden and significant UAV altitude variations lead to unexpected variations in the disparity profile.

From Figure 47 it is notable how the disparity profile changes inversely with the UAV altitude variations, particularly for the most significant UAV altitude variations, marked with black rectangles. These variations are expected, as the calculated disparity should compensate for variations in UAV flight altitude. However, as uncovered in Section 5.1.3, the disparity calculations become prone to errors for areas where the true disparity varies significantly within a window.

This also becomes evident in Figure 48, where the blue plot is the elevation profile calculated from the disparity profile and INS data presented in Figure 47. The elevation profile exhibits sudden increases in elevation at the corresponding samples marked in Figure 47. These variations are located in an area of the imaged scene where no significant variations are expected, and may therefore be regarded as a consequence of the sudden variations in UAV altitude. However, by trend adjusting the elevation profile, as described in Section 4.3.1, with a weight of -0.85, the elevation profile becomes a much more accurate representation of the actual terrain, as seen in the orange plot in Figure 48.

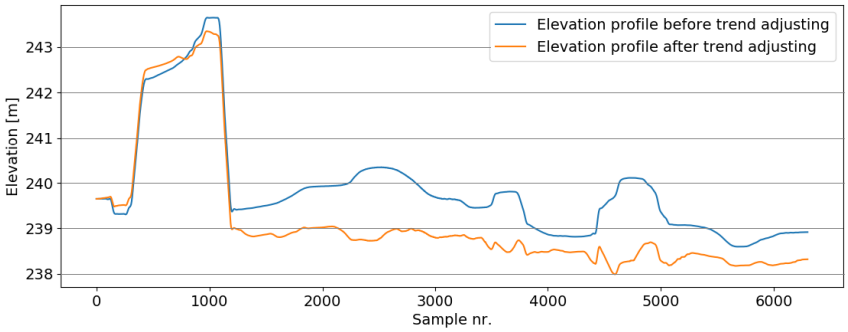


Figure 48: Elevation profile before and after trend adjusting. The elevation profile before trend adjusting displays unexpected increases in elevation, due to sudden variations in UAV altitude. These are removed by trend adjusting the elevation profile.

Point Clouds

The point cloud resulting from the captured LiDAR data with a 20 m above ground UAV flight altitude is presented below in Figure 49, while Figure 50 showcases the point cloud derived from the developed model. In Figure 49, the dashed rectangle highlights the area of the scene included in the point cloud derived from the developed model, while the dashed lines feature cross-sections for comparisons between the two point clouds.

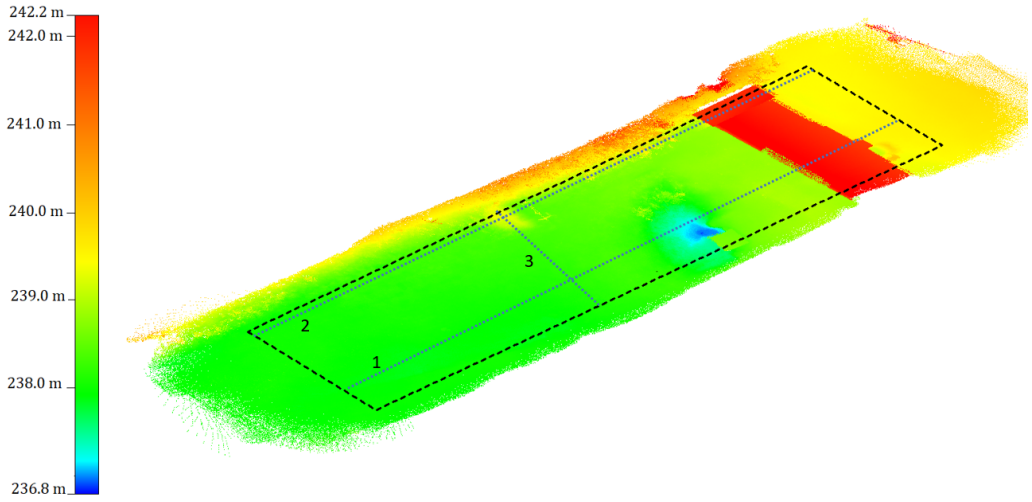


Figure 49: LiDAR point cloud from 20 m UAV flight altitude. The dashed rectangle highlights the area also considered by the developed model, while the dashed lines feature cross-sections used for evaluation. The colors indicate elevation.

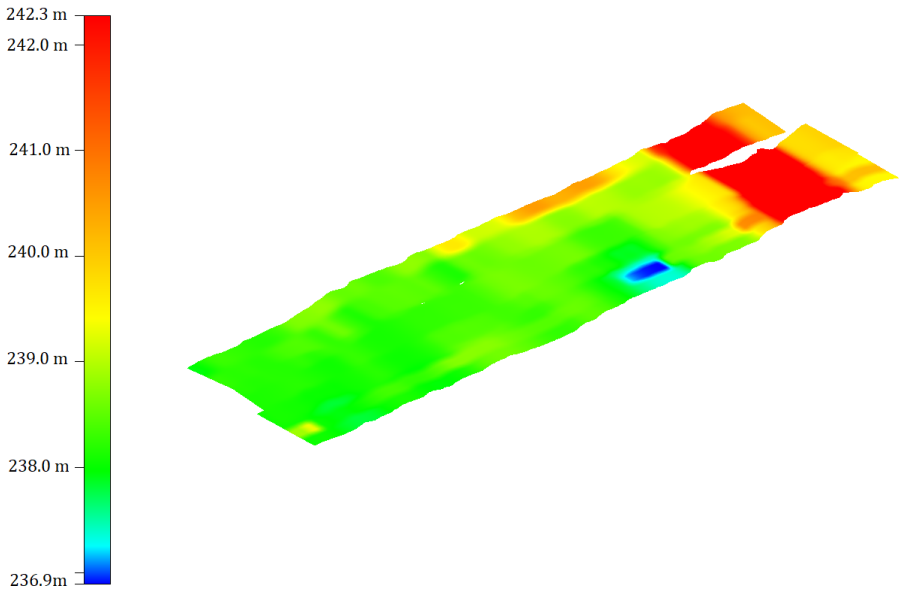


Figure 50: Point cloud derived from the developed model based on hyperspectral data captured at 20 m UAV flight altitude. The colors indicate elevation.

At first glance the two point clouds appear very similar, they both mainly consist of two areas of different elevation, the top of the house at around 242 m and the level ground at approximately 238 m. In fact, the point cloud derived from the developed model comes across as a smoothened version of the LiDAR point cloud. The smoothing effect is the result of how disparity details are lost in the transitions between two disparity levels, already discussed above and in Section 5.1.1. However, the "Fit Point Clouds" functionality in Global Mapper, used to combine the three point clouds from the flight lines, also contributes to this smoothing effect. The total elevation RMSE of the entire point cloud presented in Figure 50 is calculated to be 0.4096 m, with the assumption of the LiDAR point cloud being the ground truth. At 20 m UAV flight altitude, 0.4096 m elevation error corresponds to a disparity RMSE of 0.1352 pixels.

Figure 51 presented below, illustrates the elevation profile of cross-section 1 from both point clouds.

This area of the depicted scene is considered less complicated and the elevation profile derived from the developed model is analogous to the one resulting from the LiDAR point cloud. The elevation RMSE of cross-section 1 is 0.3277 m, corresponding to a disparity RMSE of 0.1086 pixels. Here, most of the errors are in the overlapping region between the level ground and the house, consistent with the errors in the disparity map in Figure 33 derived from synthetic data.

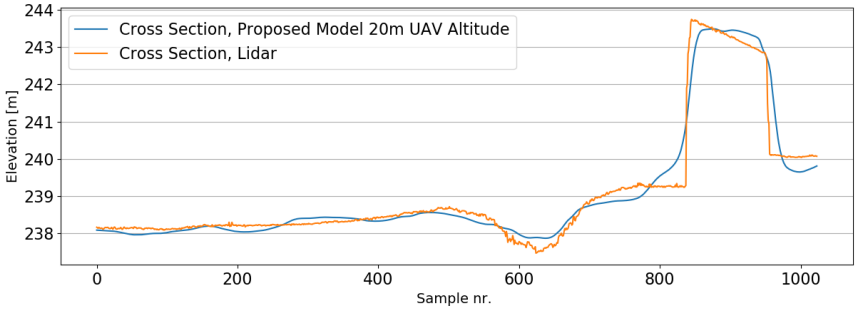


Figure 51: Elevation profiles of cross-section 1, resulting from both LiDAR data and the developed model.

The elevation profiles from cross-section 2, is presented below in Figure 52. This part of the scene can be considered more complicated with more elevation variations and some vegetation. Thus, the corresponding elevation RMSE increases somewhat and is calculated to be 0.4689 m, corresponding to a disparity RMSE of 0.1543 pixels. Despite an error increase, the elevation profile resulting from the developed model still follows the trend of the variations in the elevation profile of the LiDAR data to a large degree.

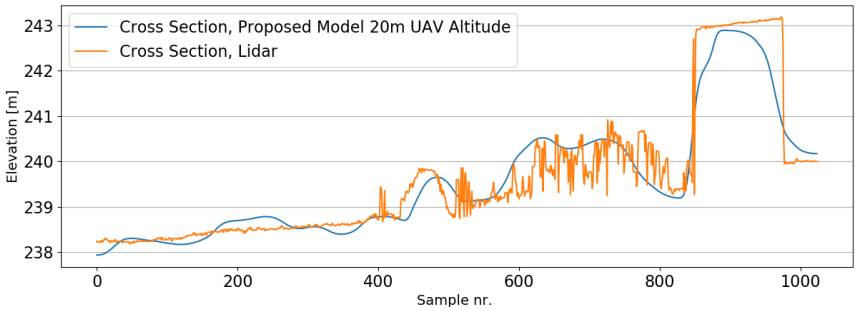


Figure 52: Elevation profiles of cross-section 2, resulting from both LidAR data and the developed model. Despite more complicated terrain, the developed model stills follow the trend in elevation to a large degree.

Both cross-sections 1 and 2 are parallel to the along-track direction, the direction where the window size is smallest, 20 pixels. In the across-track direction, on the other hand, the window size is 155 pixels, and less detail in the elevation profile is expected. This is also evident from Figure 53, where elevation data from cross-section 3 is presented. Here, the issue of utilising a window size too large is well illustrated. For the first 155 samples, the elevation is calculated to be within approximately 0.2 m off the LiDAR data. However, for the next 155 samples, the true elevation of the terrain decreases rapidly to a lower level. Hence, the window of pixels from this area consists of two disparity levels with a very rapid transition between the two. Thus, the resulting elevation also becomes inaccurate. The elevation RMSE of cross-section 3, derived from the developed model, is calculated to be 0.3156 m, corresponding to a disparity RMSE of 0.1046 pixels.

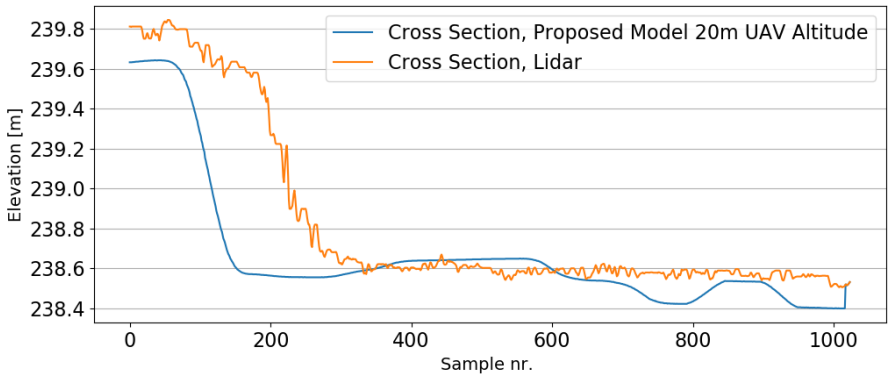


Figure 53: Cross-section 3 from Figure 49 and the elevation profiles from the LiDAR data and the developed model. The large window size of 155 pixels in the across-track direction causes some inaccuracies for areas of rapidly changing elevation.

Table 7 presented below sums up the elevation accuracy of the developed model derived from hyperspectral images captured at 20 m above ground UAV flight altitude, and the corresponding disparity accuracy derived from the assumption of a constant UAV altitude.

Table 7: Elevation errors and the corresponding disparity errors from a UAV flight altitude of 20 m above ground. All disparity errors are determined with the assumption of a constant 20 m UAV flight altitude.

Area	Elevation RMSE [m]	Disparity RMSE [nr. of pixels]
Entire Point Cloud	0.4096	0.1352
Cross-Section 1	0.3277	0.1086
Cross-Section 2	0.4689	0.1543
Cross-Section 3	0.3156	0.1046

Any errors in the elevation lead to consequential errors in all longitude and latitude calculations trough Eq. (8) and its linear dependence on Z. The longitude and latitude errors introduced by an elevation error of 0.4096 m is varying between $2.012{\cdot}10^{-9}$-$8.375{\cdot}10^{-7}$ degrees and $1.136{\cdot}10^{-9}$-$4.730{\cdot}10^{-7}$ degrees, with the pixels located at the edge of each scan line most prone to longitude and latitude errors. These errors correspond to approximately 0.000169 - 0.0703 m when calculating the distance between two points with coordinates that coincide with the location of the depicted scene.

5.2.2 40 m UAV Flight Altitude

The imaged scene captured by Camera 1 at 986.36 nm from a UAV flight altitude of 40 m above ground, is presented below in Figure 54. The images are once again brightness enhanced for better visualisation.

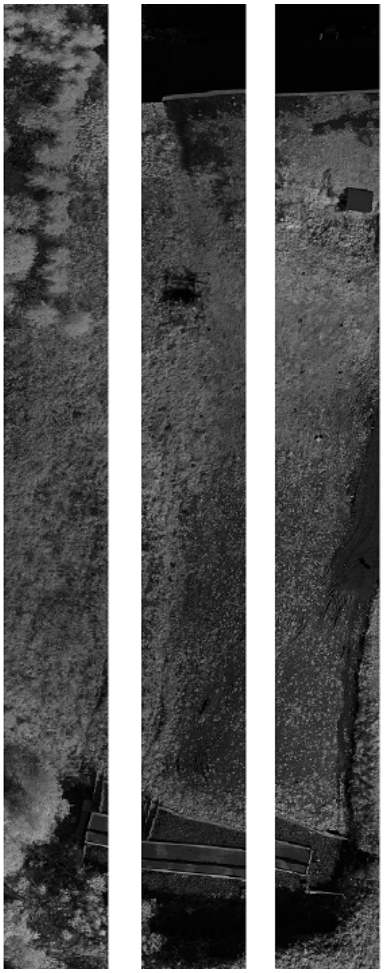


Figure 54: Imaged scene captured by Camera 1 from 40 m UAV flight altitude. The images have been brightness enhanced for better visualisation.

The images resulting from flight lines 2 and 3, in Figure 54, share much of the same scenery as the images captured from 20 m UAV flight altitude. The disparity is also determined similarly with a window size of 155×20 pixels^2. However, the disparity range is adjusted to 3-5 pixels. The data from flight line 1, on the other hand, is an area with much vegetation and a varying elevation. To preserve detail from this area, the window size is reduced to 62×20 pixels^2 and the disparity range employed is 3-7 pixels. Figure 55 presented below, illustrates the along-track disparity profile from the center of the image resulting from flight line 2.

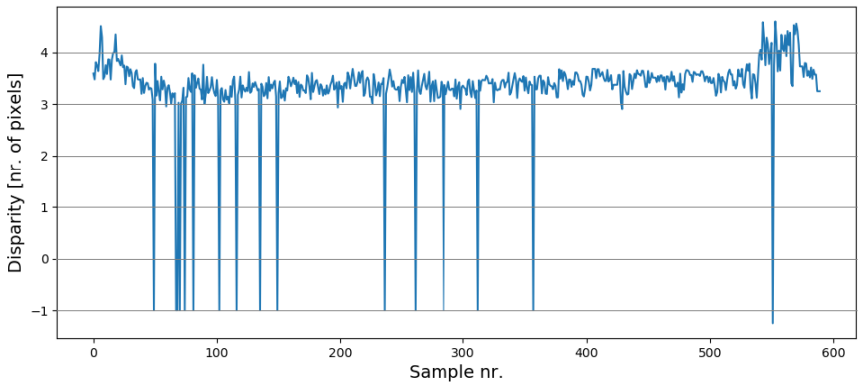


Figure 55: Raw disparity profile in the along-track direction from flight line 2, based on images from 40 m UAV flight altitude.

The disparity profile presented in Figure 55 includes much of the same area as the one presented in Figure 43 derived from 20 m UAV flight altitude, and they both have similar shapes, consisting mainly of the level ground and the top of the house. However, Figure 55 also includes more gradually changing elevation in the first 50 samples of the disparity profile. The first portion of the disparity profile showcased in Figure 55 also includes a number of disparity holes. This area suffers significantly from low illumination, as can be seen from the bottom of the image resulting from flight line 2 in Figure 54. Thus, also including spectral channels beyond the overlapping wavelength region of Camera 1 and Camera 2 would most likely be beneficial for this area of the depicted scene. Below, presented in Figure 56 is the same disparity profile as in Figure 55 after being processed and refined.

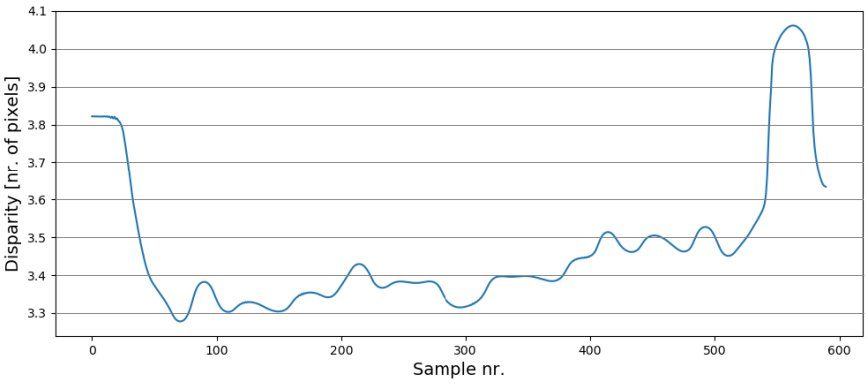


Figure 56: Processed and refined disparity profile resulting from the raw disparity profile presented in Figure 55.

The raw along-track disparity profile of a more complicated part of the imaged scene is presented below in Figure 57. This is the most left area of the image in Figure 54 from flight line 1 and consists of vegetation as well as changes in elevation. This area is evaluated with a pixel size of 62 pixels in the across-track direction, which is expected to reduce the accuracy of the calculation to some degree, however, necessary to preserve the detail of the scene. The amount the accuracy of the disparity calculations is reduced is difficult to manifest by Figure 57 alone, however, by comparison to Figure 55, the disparity profile in Figure 57 seems to fluctuate more from one sample to the next, suggesting a larger degree of uncertainty. This is also supported by an increasing disparity fluctuation for corresponding spatial areas but calculated from different spectral channels. The processed and refined disparity profile in Figure 57 is presented below in Figure 58.

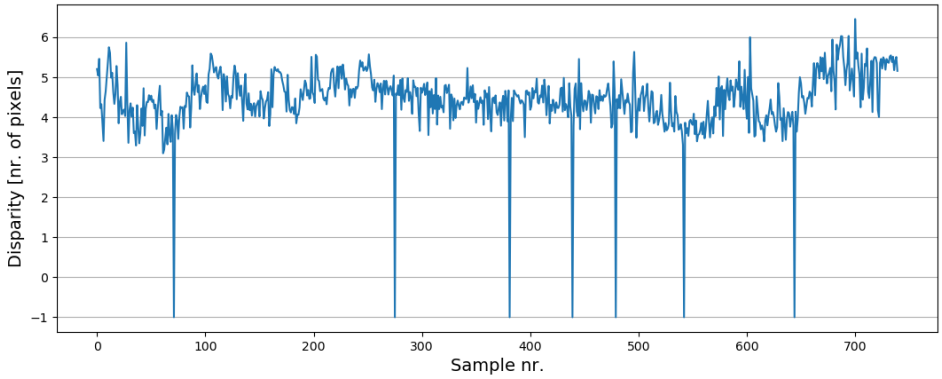


Figure 57: Raw disparity profile in the along-track direction of a more complicated area of the scene, based on images from 40 m UAV flight altitude.

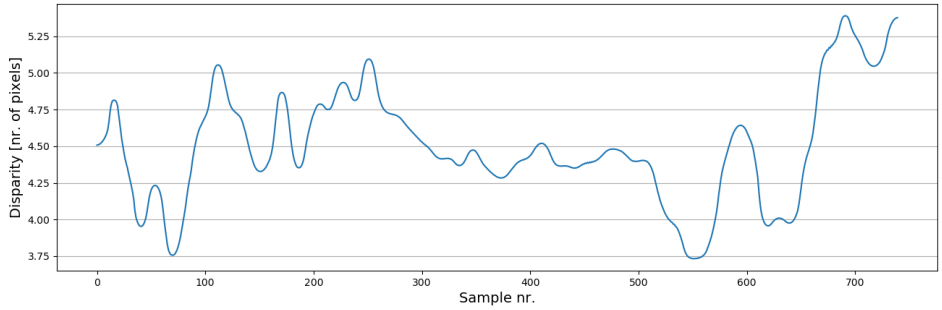


Figure 58: Processed and refined disparity profile from the raw disparity profile presented in Figure 57.

Visually, the along-track disparity profiles showed little variations in accordance with changes in UAV altitude. Figure 76 located in the Appendix Section D highlights this, where an along-track disparity profile from flight line 2 and the INS altitude data are plotted. Less degree of correlation between the disparity and UAV altitude is somewhat expected considering how UAV altitude variations at 40 m cause very small variations in disparity. Despite this, trend adjusting the elevation profiles of less complicated areas in the depicted scene still improved the elevation accuracy. This is demonstrated in Figure 59, where the along-track elevation profile of a less complicated area is plotted before and after trend adjusting with a weight of -0.5. Considering how the ground truth elevation between sample 1000-5000 is between 238-239 m and how the house located between sample 5000-6000 is somewhere in the middle of 243 m and 244 m, trend adjusting the elevation profiles of less complicated areas improved performance in terms of elevation accuracy.

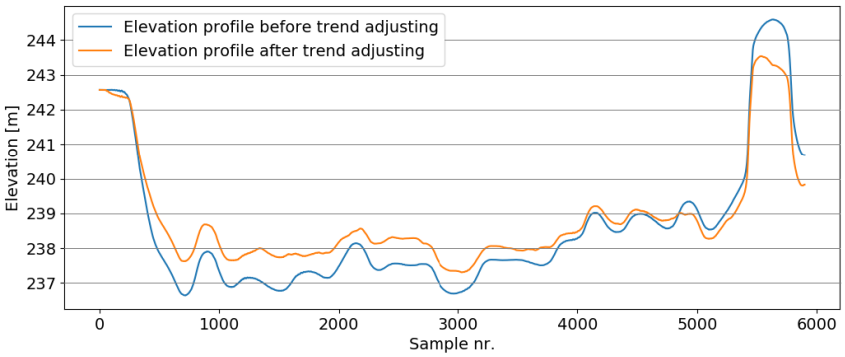


Figure 59: Elevation profiles before and after trend adjusting. Despite the disparity profiles in the along-track direction being little affected by sudden variations in UAV flight altitude, trend adjusting the elevation profiles still improved accuracy.

Point Clouds

The point clouds resulting from the recorded LiDAR data at 40 m UAV flight altitude as well as the point cloud derived from the developed model are presented below in Figure 60 and Figure 61 respectively. In Figure 60 the dashed rectangle highlights the area considered by the developed model, while the dashed lines once again feature the cross-sections used for comparison of the two point clouds. Here, cross-section 1 is the same cross-section marked in Figure 49.

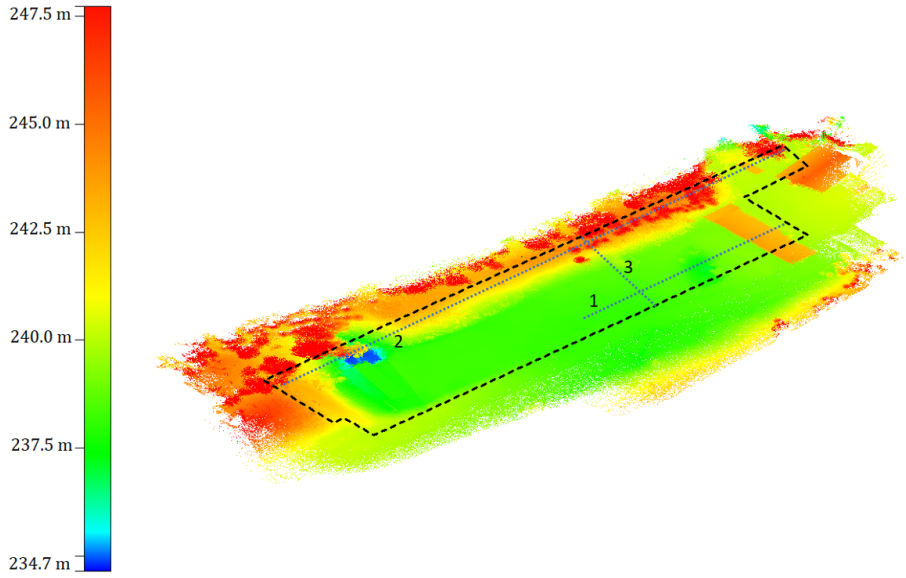


Figure 60: LiDAR point cloud resulting from 40 m UAV flight altitude. The dashed triangle highlights the area also considered by the developed model, while the dashed lines are cross-sections used for comparison. The colors indicate elevation.

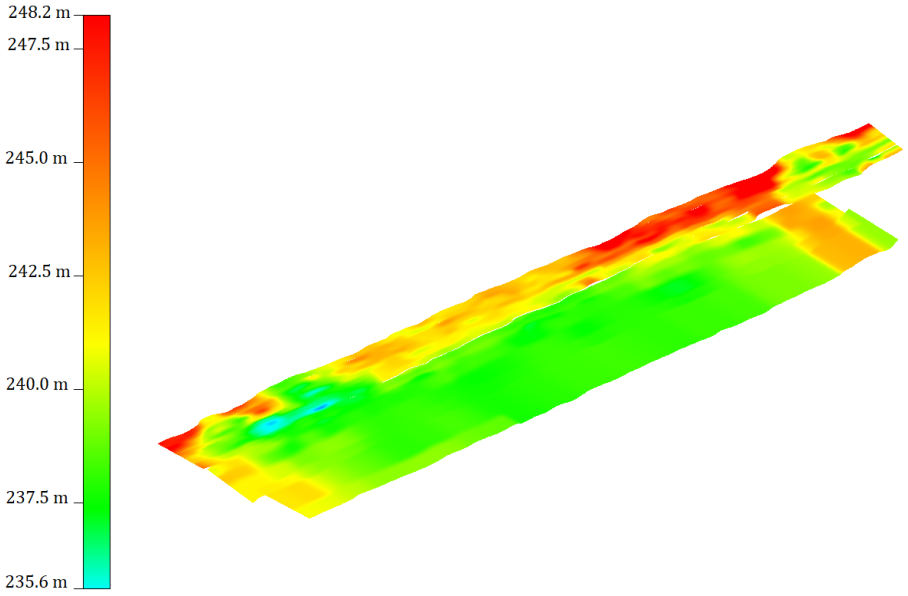


Figure 61: Point cloud resulting from the developed model, based on hyperspectral data from 40 m UAV flight altitude. The colors indicate elevation.

Similar to the point clouds presented in Section 5.2.1 resulting from 20 m UAV flight altitude, the point cloud resulting from the developed model appears as a smoothened and smeared-out version of the LiDAR point cloud. This is especially apparent in the more complicated part of the scene, the most elevated area in Figure 61. The total elevation RMSE of the entire point cloud is calculated to be 1.2049 m with the assumption of the LiDAR point cloud being ground truth. This corresponds to a disparity RMSE of approximately 0.0986 pixels assuming a constant UAV flight altitude of 40 m.

The elevation profiles of cross-section 1 is presented below in Figure 62, and is the same cross-section presented in Figure 51 derived from 20 m UAV altitude. Similar to the plot presented in Figure 51, the elevation profile resulting from the hyperspectral images taken at 40 m UAV altitude, is very analog to the LiDAR elevation profile. However, the decrease in elevation at around sample nr. 600 is not registered by the elevation profile derived from the developed model. This change in elevation was present in the elevation profile in Figure 51 and illustrates how detail is lost when the UAV flight altitude is increased. The elevation RMSE of cross-section 1, presented below in Figure 62, is calculated to be 0.6075 m corresponding a disparity RMSE of 0.0504 pixels.

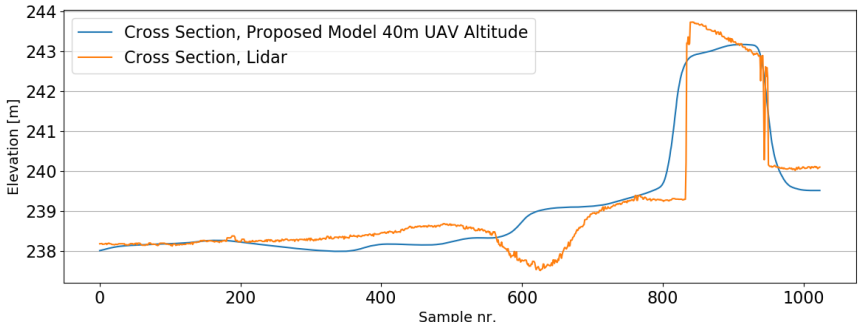


Figure 62: Cross-section 1 from Figure 60 and the elevation profiles from LiDAR data and the developed model. Most elevation detail is preserved by the developed model, however the decrease in elevation around sample nr. 600 is not registered.

A more complicated area of the depicted scene, cross-section 2, is presented below in Figure 62.

When compared to Figure 63, a less complicated area of the depicted scene, it is evident how the accuracy of the developed model decrease for areas with significantly fluctuating elevation, also expected from the results uncovered in Section 5.1.3. However, also anticipated from Section 5.1.3, is how the trends in the elevation profile still are preserved even though the accuracy is somewhat degraded. The elevation RMSE of the elevation profile presented in Figure 63 is calculated to be 1.6932 m, corresponding to a disparity RMSE of 0.1369 pixels.

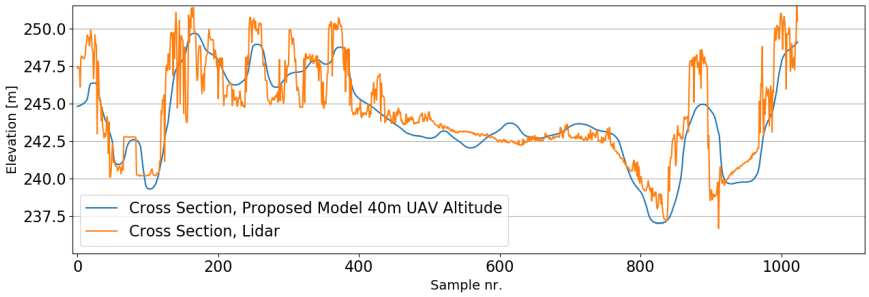


Figure 63: Elevation profiles of cross-section 2 from Figure 61. More complicated terrain degrades the elevation accuracy of the developed model.

The elevation profiles of the across-track cross-Section 3 from Figure 60, is presented below in Figure 64. From Figure 64 the benefit of reducing the across-track window size to 62 pixels is evident as the elevation profile derived from the developed model follows the trend of the LiDAR profile to a large extent. This is true also for the first part of the cross-section, an area with rapidly changing elevation. The elevation RMSE of cross-section 3 is calculated to be 1.0839 m, corresponding to a disparity RMSE of 0.0889 pixels.

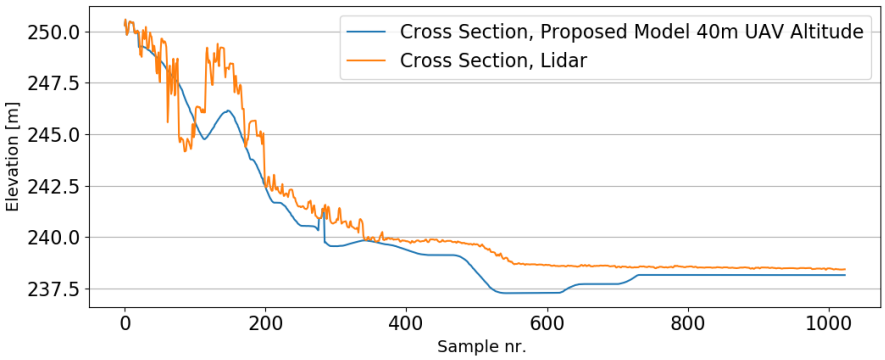


Figure 64: Elevation profiles of cross-section 3 from Figure 60. Reducing the across-track window size preserves detail of the imaged scene for the first 400 samples.

Table 8 presented below sums up the elevation accuracy and disparity accuracy of the developed model, for the areas considered above, based on a UAV flight altitude of 40 m.

Table 8: Elevation errors and the corresponding disparity errors, resulting from 40 m UAV flight altitude. All disparity errors are based on the assumption of a constant 40 m UAV flight altitude.

Area	Elevation RMSE [m]	Disparity RMSE [nr. of pixels]
Entire Point Cloud	1.2049	0.0986
Cross-Section 1	0.6075	0.0504
Cross-Section 2	1.6932	0.1369
Cross-Section 3	1.0839	0.0889

The longitude and latitude errors due to an elevation error of 1.2049 m corresponds to $5.220 \cdot 10^{-9}$-$2.173 \cdot 10^{-6}$ degrees and $3.623 \cdot 10^{-9}$-$1.508 \cdot 10^{-6}$ degrees, respectively. This yields an error of 0.000497-0.2069 m when determining the distance between two points, derived from the set of equations presented in Section 2.2.3.

5.2.3 60 m UAV Flight Altitude

The imaged scene captured by Camera 1 at 986.36 nm from a UAV flight altitude of 60 m above ground, is presented below in Figure 65.

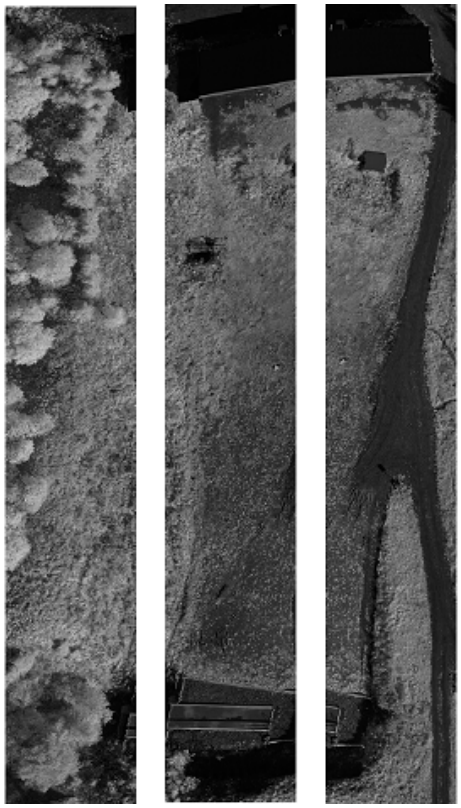


Figure 65: Imaged scene captured by Camera 1 from 60 m UAV flight altitude. The images have been brightness enhanced for better visualisation.

The disparity of the images is determined in the same manner as the images captured from a UAV flight altitude of 40 m above ground, namely with a window size of 62×20 pixels^2 for flight line 1, and 155×20 pixels^2 for flight line 2 and 3. The disparity range employed is 2-4 pixels for all images. Figure 66 presented below, showcases the raw disparity profile in the along-track direction of the center of the image resulting from flight line 2.

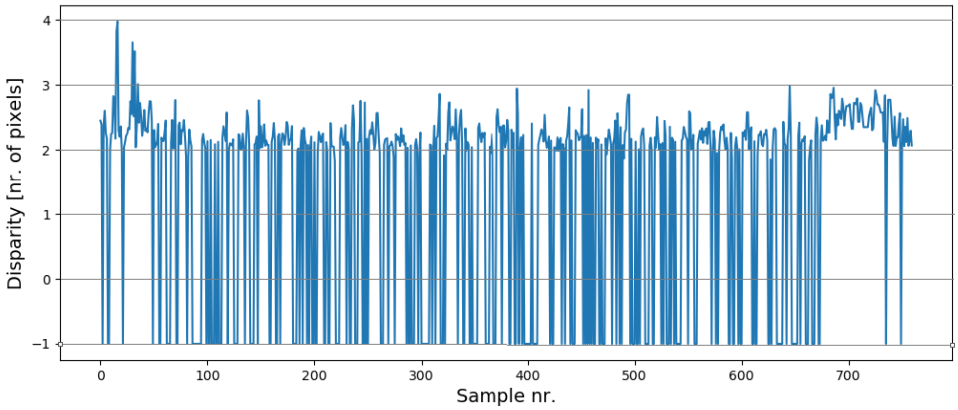


Figure 66: Raw disparity profile in the along-track direction based on the center of the images captured in flight line 2.

Particularly apparent from Figure 66 is the significant number of disparity holes. Further inspection reveals that the majority of the disparity holes are the result of the calculated disparities being outside the disparity range, and not the result of failure to fit the PC function to Eq. (22) or Eq. (23). The initial disparity values associated with the disparity holes typically lie around 1.9 pixels and would result in elevation errors of many meters according to Table 2. All disparity profiles resulting from flight lines 1, 2, and 3 demonstrated a large number of disparity holes. It is difficult to pin-point the exact reason behind the large number of disparity holes, however, one explanation could be how an increasing on-ground pixel size leads to more disparity variations within each window used for stereo matching, which again reduces the accuracy, as uncovered in Section 5.1.3. On the other hand, then by reducing the window size, one would expect the accuracy to increase, however, this was not observed, as can be seen in the plot presented in Figure 77 calculated with a window size of 62×20 pixels^2, located in the Appendix Section D. Expanding the spectral range of Camera 1 and Camera 2 to fifteen additional bands outside the overlapping wavelength region also resulted in no increase in performance, which also can be seen in Figure 77. It should be noted, however, that a disparity range as narrow as 2-4 pixels leaves very little room for errors before the calculations end up outside this range and are assigned a value of -1. This most likely contributes to the significant number of disparity holes.

Neglecting the disparity holes, the profile illustrated in Figure 66 shares many similarities to the one presented in Figure 55 derived from 40 m UAV altitude, consisting mainly of the level ground and the top of the house located at around sample nr. 700. However, the disparity profile presented in Figure 66 exhibits significant fluctuations from one sample to the next, indicating a large degree of uncertainty. This was also observed in other parts of the imaged scene, one example is presented in Figure 78 located in the Appendix Section D. Below presented in Figure 67, is the processed and refined version of the disparity profile in Figure 66.

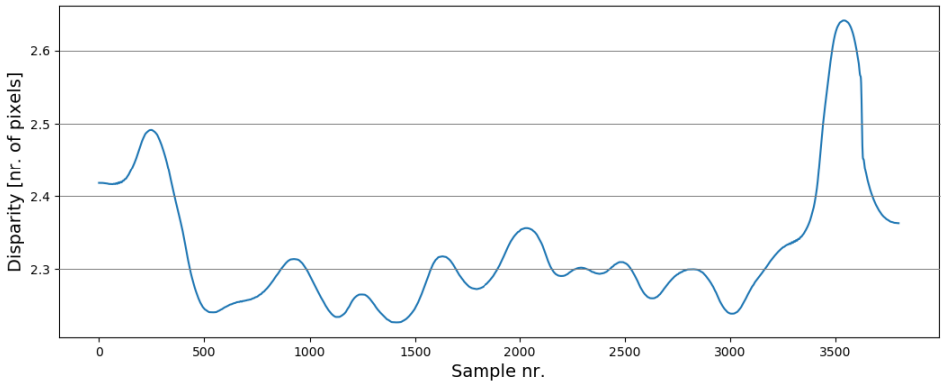


Figure 67: Processed and refined disparity profile resulting from the raw disparity profile presented in Figure 66.

The elevation profile resulting from Figure 67 is presented below in Figure 68. From Figure 68 it becomes clear how the fluctuating disparity profile causes serious variations in the elevation profile in areas regarded as constant. Trend adjusting the elevation profile resulting from 60 m UAV flight altitude showed no notable increase in performance, as exemplified in Figure 79 located in the Appendix Section D. One very apparent aspect of Figure 67 Figure 68, is their almost identical shape. Every variation in the disparity profile is also present in the elevation profile. This showcases how prone the developed model is to disparity errors at the higher UAV altitudes, and variations in the disparity of only 0.02 pixels yield variations of 0.5 m in the final elevation profile.

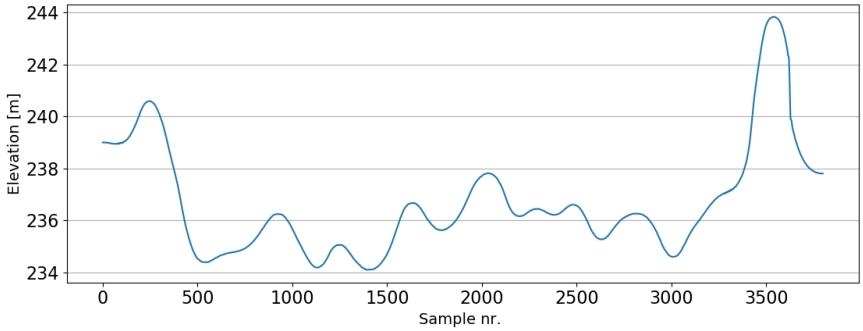


Figure 68: Elevation profile calculated from the disparity profile in Figure 67.

Point Clouds

The point cloud resulting from the recorded LiDAR data as well as the point cloud derived from the developed model is presented below in Figure 69 and Figure 70. The dashed rectangle in Figure 69 highlights the area also considered by the developed model, while the dashed lines are cross-sections used for comparison. Here, cross-section 1 is the same area already considered for UAV flight altitudes of 20 m and 40 m.

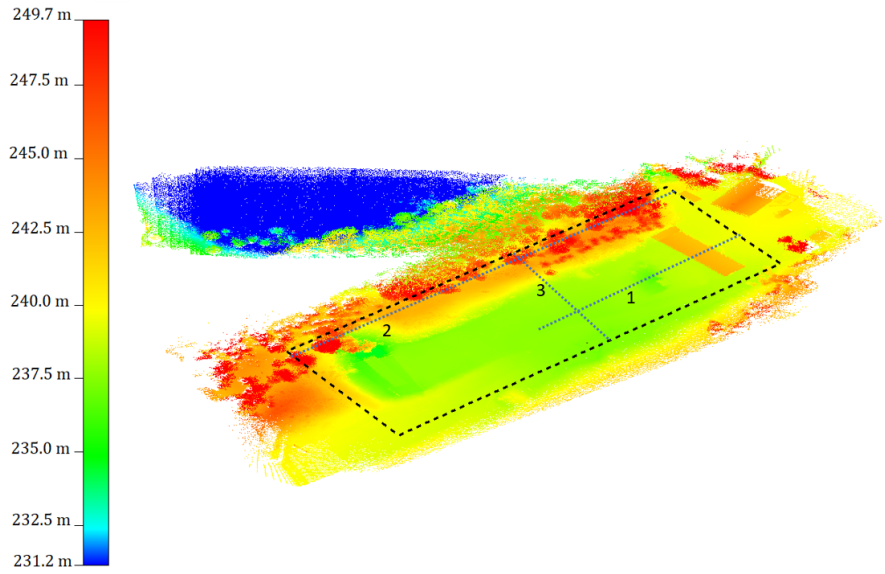


Figure 69: Lidar point cloud resulting from 60 m UAV flight altitude. The dashed triangle highlights the area also considered by the developed model, while the dashed lines are cross-sections for comparison. The colors indicate elevation.

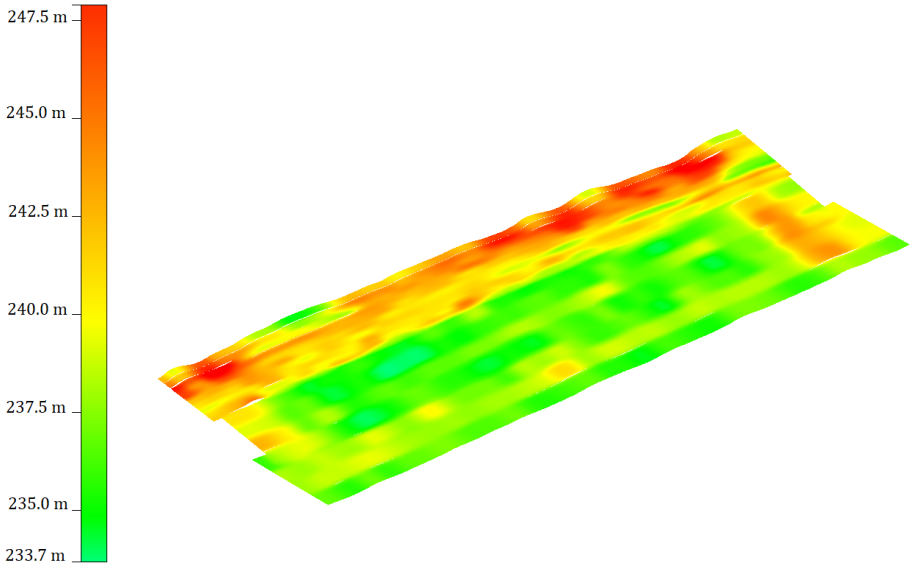


Figure 70: Point cloud resulting from the developed model, based on hyperspectral images captured from 60 m UAV flight altitude. The colors indicate elevation.

Similar to the point clouds resulting from 20 m and 40 m UAV flight altitude, the point cloud derived from the developed model presented in Figure 70 appears as a smeared-out and smoothened version of the LiDAR point cloud. At 60 m UAV flight altitude this smoothing effect is expected to be particularly noticeable due to the small disparity variations. Thus, the significant smoothing effect seen in Figure 70 is expected and agrees well with what was uncovered in Section 5.1.1, particularly Figure 34. The elevation RMSE of the derived point cloud is calculated to be 2.4918 m, which corresponds to a disparity RMSE of 0.0896 pixels.

The elevation profiles of cross-section 1 are presented below in Figure 71. From Figure 71 it becomes clear how the inaccurate and fluctuating disparities cause a significant reduction in the accuracy of the point cloud. Further, although there is a clear increase in elevation associated with the house around sample 800-1000, the shape of the elevation profile shows little resemblance to a

house when compared to the LiDAR elevation profile, due to the severe smoothing. The elevation RMSE of cross-section 1, plotted in Figure 71, is calculated to be 2.3872 m, which corresponds to a disparity RMSE of 0.0859 pixels.

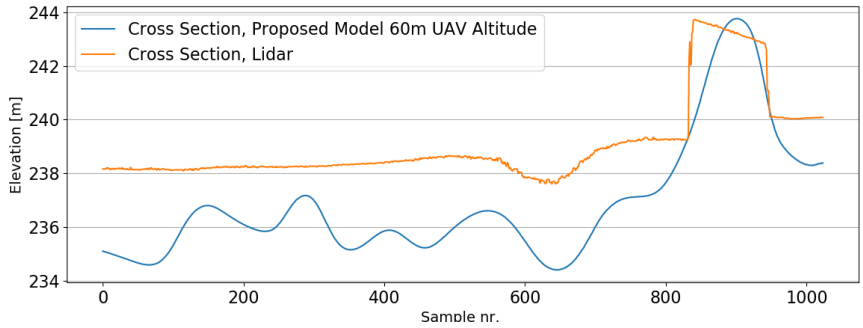


Figure 71: Elevation profiles of cross-section 1 from Figure 69 from both LiDAR data and the developed model.

Similar tendencies can also be seen in Figure 72 where the elevation profiles of cross-section 2 are plotted. This part of the scene can be considered quite complicated with much change in elevation resulting in even more inaccuracy. The elevation RMSE of cross-section 2 is calculated to be 2.8616 m, corresponding to a disparity RMSE of 0.1023 pixels.

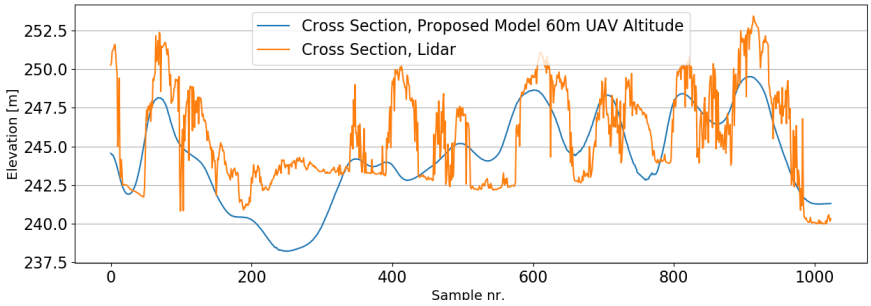


Figure 72: Elevation profiles of cross-section 2 from both LiDAR data and the developed model.

The elevation profile derived from the developed model of cross-section 3, also showed a large degree of inaccuracy, as seen from the plots in Figure 73. The elevation RMSE of cross-section 3 is calculated to be 1.8011 m, corresponding to a disparity RMSE of 0.0654 pixels.

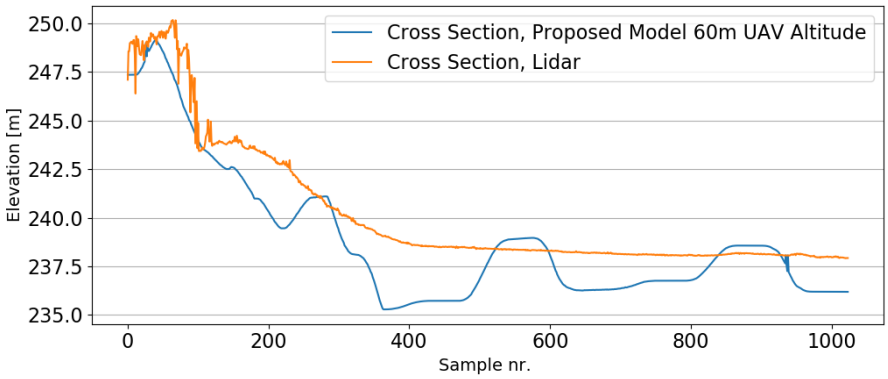


Figure 73: Elevation profiles of cross-section 3 from both LiDAR data and the developed model.

Table 9 presented below sums up the elevation accuracy of the developed model, given a UAV flight altitude of 60 m above ground.

Table 9: Elevation errors and the corresponding disparity errors, resulting from 60 m UAV flight altitude. All disparity errors are calculated based on the assumption of a constant 60 m above ground UAV flight altitude.

Area	Elevation RMSE [m]	Disparity RMSE [nr. of pixels]
Entire Point Cloud	2.4918	0.0896
Cross-Section 1	2.3872	0.0859
Cross-Section 2	2.8616	0.1023
Cross-Section 3	1.8011	0.0654

The longitude and latitude errors due to an elevation error of 2.4918 m correspond to $9.645 \cdot 10^{-9}$-$4.015 \cdot 10^{-6}$ degrees and $7.876 \cdot 10^{-9}$-$3.279 \cdot 10^{-6}$ degrees, respectively. This again yields an error of 0.001028-0.428 m when calculating the distance between two points.

5.2.4 Point Clouds Accuracy and Comparison

The overall elevation RMSE of the different point clouds derived from the developed model was calculated to be 0.4096 m, 1.2049 m, and 2.4918 m for UAV flight altitudes of 20 m, 40 m, and 60 m respectively. Considering the inverse relation between Z and the disparity, plotted in Figure 12, the exponential increasing elevation error with UAV altitude is also expected. Further, the elevation errors were lowest for the less complicated areas of the depicted scene, i.e areas with less change in elevation. Additionally, most elevation details were preserved in the along-track direction, the direction with the smallest window size. This was observed for all derived point clouds and agrees well with the expectations stated in Section 5.1.4.

For less complicated terrain and lower UAV flight altitudes, such as cross-section 1, an area considered by all UAV flight altitudes, the point clouds demonstrated accurate representations of the actual terrain. This was particularly evident given a UAV flight altitude of 20 m above ground, as illustrated in Figure 51 where the elevation profile derived from the developed model almost overlaps the LiDAR data. However, also at 40 m UAV altitude, this area of the depicted scene showed impressive accuracy, although some detail was lost. Given a UAV altitude of 60 m, however, the accuracy of the developed model decreased significantly and the elevation profile plotted in Figure 71 showed very little resemblance to the LiDAR data.

For more complicated areas, marked as cross-section 2 in all point clouds, a drop in elevation accuracy was observed. At 20 m UAV altitude the elevation error increased by 0.06 m, 0.49 m at 40 m UAV altitude, and 0.40 m at 60 m UAV altitude wrt the remainder of the point cloud. Here, the small error increase seen in the point cloud derived from 20 m UAV altitude is largely due to the complicated area considered being much less complicated compared to the areas evaluated in the point clouds derived from 40 m and 60 m UAV flight altitude. However, despite this, one could still expect the point cloud derived from 20 m UAV altitude to have the least significant increase in elevation error if the same terrain was considered. Further, at first glance, it may appear somewhat surprising how the elevation error for complicated terrain increases less given a UAV altitude of 60 m compared to 40 m. However, this is most likely the result of how the elevation error of the entire point cloud derived from 60 m UAV altitude is rather significant in the first place and complicated terrain does not stick out in this manner. Despite an increase in elevation error for higher UAV altitudes, the point cloud derived from a UAV altitude of 40 m above ground, demonstrated a satisfactory accuracy also for complicated terrain. This, however, does not apply to the point cloud derived from a UAV altitude of 60 m, where most details of complicated terrain were lost.

The errors in latitude and longitude, introduced by the elevation errors, vary significantly within each scan line, with the pixels located at the edges of the scan lines most susceptible. Considering the point cloud derived from 20 m UAV altitude these errors are quite insignificant, at most around

0.07 m. For the point clouds derived from a UAV altitude of 40 m and 60 m, these errors can be expected to be more significant. Particularly the calculations based on the hyperspectral data from flight line 1 are prone to errors. The images from flight line 1 contain most of the complicated area of the imaged scene and pixels located at the edges of the scan line are vulnerable to errors in longitude and latitude corresponding to approximately 0.2 m and 0.4 m, given a UAV altitude of 40 m and 60 m respectively.

Yet to be discussed is how all error estimates presented above, are based on the captured LiDAR data. Although, up to this point being regarded as ground truth, the LiDAR data is not completely flawless and precise. However, based on VLP-32C's datasheet, presented in [49], the LiDAR can provide accuracy down to 3 cm at a distance of 200 m, suggesting a minuscule amount of errors being introduced to the calculations as the result of errors in the LiDAR data. The same conclusion can also be drawn wrt the INS system Applanix APX-15, which provides positional data down to 2 cm according to its datasheet, located in [22]. With regards to the equations presented in Section 2.2.3, based on a spherical earth and utilised to calculate the longitudes and latitudes of the point clouds, an error of 0.334% is introduced when calculating the distance between two points. For the point clouds derived, this error would be most significant given a UAV altitude of 60 m and is calculated to be 0.034 m and be regarded as negligible. Thus, in terms of accuracy, the errors in derived point clouds are primarily the result of errors in the elevation estimates, not errors introduced by the LiDAR data, INS system, nor the set of equations based on a spherical earth.

5.3 System Evaluation, Limitations and Possibilities

Overall, the stereoscopic aspect of the imaging system combined with the developed model worked well for UAV-based 3D terrain mapping. Next, the different features of the system are evaluated, the developed model and the imaging setup, with some of the aspects presented in Section 1.4 revisited, and the system is also compared to literature inspiring much of the presented work.

5.3.1 Developed Model

Arguably, the most important aspect of the developed model is the stereo matching algorithm. During the development of the algorithm, the main focus was to achieve accurate sub-pixel disparity resolution, ideally below 0.1 pixels, while at the same time being robust against radiometric differences and periodic patterns.

With regards to accurate sub-pixel disparity resolution, the results derived from the synthetic generated data demonstrated an impressive potential for sub-pixel disparity resolution. By implementing a two-step phase-based approach, disparity accuracy down to 0.02 pixels was achieved in a scene of varying disparity. 0.02 pixels disparity resolution is in the same ball-park as the inspiring results presented in [34] and [45], where the majority of inspiration behind the developed stereo matching algorithm was taken. The stereo matching algorithm also performed well on spatially misregistered hyperspectral data, given small window sizes.

The sub-pixel accuracy of the developed stereo matching algorithm decreased somewhat when applied to real stereoscopic hyperspectral data of the imaged scene. The maximum expected elevation accuracy, presented in Table 6, based on a disparity error of 0.02 pixels was not achieved. Instead, the disparity error fluctuated between 0.05-0.15 pixels, depending on the type of terrain considered. The larger portion of the disparity errors is likely to be the result of varying disparity within the windows of pixels considered, both due to elevation and UAV altitude variations. At first, it may be surprising how the data captured from 20 m UAV altitude yielded the highest disparity RMSE. However, at this altitude, the developed stereo matching algorithm is also most prone to variations in the UAV altitude inducing a varying disparity in the along-track direction. It should also be noted how the disparity errors presented in Table 7 are based on a constant 20 m UAV flight altitude. In reality, the UAV altitude was closer to 23 m above ground, making the true disparity errors similar to the ones calculated from 40 m and 60 m UAV altitude. Further, the significant increase in elevation error seen from the data captured at 60 m UAV altitude might initially be unexpected, however, judging by the disparity errors, the point cloud derived from

a UAV altitude of 60 m is consistent with the point clouds derived from 40 m and 20 m UAV altitude. To achieve an elevation accuracy between 0.4-1.2 m also at 60 m UAV flight altitude, with the imaging system at hand, a disparity accuracy of 0.015-0.044 pixels is necessary. This is more precise than any of the disparity estimates presented in Section 5.2 and may therefore be regarded as beyond the limits of what to expect from the developed stereo matching algorithm. However, the disparity accuracy of 0.1 pixels, mentioned in Section 1.4, was for the most part achieved also given a UAV altitude of 60 m above ground.

One tweak that possibly could make the stereo matching algorithm more precise, is to a larger degree exploit how the Fourier-domain plane fitting approach to stereo matching remains accurate for very small window sizes. This became clear from Section 5.1 when the synthetic and spatially misregistered data was evaluated, and could have been utilised in the developed stereo matching algorithm by dividing the spatial windows into smaller sub-windows after the window alignment step. Fourier-domain plane fitting could be performed on the sub-windows for higher accuracy and greater detail, particularly beneficial for areas with varying disparity. This would most likely also reduce the smoothening effect seen in the point clouds especially evident at higher UAV altitudes.

With regards to the robustness of the developed stereo matching algorithm, the results revealed in Section 5.1.2, where the synthetic data was contaminated with artifacts known to complicate stereo matching, demonstrated the superior reliability of phase-based stereo matching in comparison to the intensity-based SGBM algorithm. This also translated to the real hyperspectral data of the depicted scene, as neither radiometric difference nor oscillations causing periodic patterns in the images appeared to have much impact on the results. One aspect that may introduce errors to the developed model, is how the imaging system consists of two different hyperspectral cameras. It was demonstrated in Section 5.1.2 how noise components, altering the spatial frequency content of the images, reduce the accuracy of phase-based stereo matching. Although the two imaging sensors are very similar, differences in the spatial frequency content of the images, also within the overlapping wavelength region, can be expected. This also most likely contributes to errors in the calculations.

It was also demonstrated how including multiple spectral channels from both Camera 1 and Camera 2 and calculating PC_{score} increased the robustness of the developed stereo matching algorithm. This especially became apparent in low illuminated areas of the scene. The number of bands included from both Camera 1 and Camera 2 could potentially be increased to both entire hyperspectral cubes for even more robustness. However, spectral channels located too far from each other in the electromagnetic spectrum may become redundant in the developed stereo matching algorithm, also indicated by calculations on data from 60 m UAV altitude.

Although the precision of the developed model increased by implementing an elevation refinement step, by trend adjusting the elevation profiles in the along-track direction, this part of the developed model may seem a little out of place and lacks standardisation. As of now, the weights employed for this step are based on trial and error for each along-track elevation profile individually and set based on knowledge of the desired result. This step of the implementation could certainly benefit from a re-visit and a more standardised operation.

In fact, the entire developed model could be re-designed and standardised to a much larger degree. As of now, every step of the model is executed individually, and parameters are adjusted along the way for optimal results. Ideally, the developed model should be one function call, taking some input parameters, and yielding a point cloud as output. As far as input parameters, the hyperspectral cubes, which bands to consider, INS data, window size, disparity range, and some filter parameters should provide a solid foundation and also parameters utilised in OpenCV's SGBM algorithm.

5.3.2 Imaging system

Given how the stereoscopic aspect of the imaging system is a byproduct of achieving the optimal co-registered hyperspectral cube, the imaging setup comes with a few unconventional assets. The narrow baseline of only 75 mm puts a fundamental limit on the achievable accuracy of the developed model. It was mentioned in Section 1.2.1, how in [4] a similar experiment was conducted, however with a baseline of 1.5 m. Elevation accuracy of 0.56-0.65 m was reported for a UAV altitude of 40 m. Modifying Eq. (2) with a baseline of 1.5 m, and using the disparity errors presented in Table 8, 0.0504-0.1369 pixels, from the point cloud derived at 40 m UAV altitude, the corresponding elevation errors turn out to be 0.03-0.08 m. This demonstrates how the unconventional narrow baseline limits the accuracy of the developed model, while at the same time showcasing an impressive potential given the circumstances presented in [4]. Despite this, even with a wide baseline, small details such as electrical wires and branches showcased in a LiDAR point cloud, would likely not be present in point clouds derived from stereoscopic imaging.

The hyperspectral aspect of the imaging system also improved the performance of the developed model by considering several spectral channels. However, given the two very different spectral regions covered by Camera 1 and Camera 2, only a very limited part of the hyperspectral cubes was employed. Ideally, both cameras should have been indistinguishable, covering the same spectral regions, such the entire hyperspectral cubes could have been utilised, improving the performance further.

Additionally, the imaging technique of pushbroom scanning complicates matters further. As the images are built in a line-by-line fashion and the UAV is constantly moving, disparity variations are induced to the images in the along-track direction. This complicates the stereo matching process, as revealed by the results, degrading the performance. A more conventional imaging setup, based on "normal" perspective imaging, such as the one presented in [4] enjoys the advantage of imaging the entire scene in one go. The disparity is then only the result of elevation variations, making the stereo matching process less complicated. On the other hand, pushbroom scanning provides an easy and intuitive scheme for precise georeferencing, especially true given the impressive INS for the imaging system at hand. However, the developed model would most likely benefit from an even more stable UAV platform.

5.3.3 Final Remarks

Based on the results, it is evident how the imaging system at hand may be employed to extract 3D information from a depicted scene, despite the stereoscopic aspect being a byproduct of achieving the optimal co-registered hyperspectral cube and the imaging setup not being specifically tailored for 3D terrain mapping. As far as limitations, the unconventional narrow baseline limits the achievable accuracy for even the most precise stereo matching algorithms. On the other hand, the hyperspectral aspect of the imaging system improves the robustness, while pushbroom scanning allows for an easy and intuitive scheme for precise georeferencing. However, with regards to terrain mapping, achieving high accuracy is essential and the elevation accuracy of 0.4 m achieved from 20 m UAV altitude is not that impressive compared to the accuracies down 0.02 m, attainable with LiDAR technology. Precision down below 0.1 m is obtainable, however, with a wider baseline, as demonstrated by using Eq. (2) with B=1.5 m. Another crucial aspect of terrain mapping is how well details are preserved. It was mentioned in Section 1.2.1, how the LiDAR configuration in [9] generated point clouds so dense and detailed that electrical wires could be identified at a distance of 100 m in an urban environment. This amount of detail would most likely never be preserved by a stereoscopic imaging system, even if the imaging setup is tailored specifically for stereoscopic usage.

Thus, for detailed and accurate 3D terrain mapping, the imaging system at hand may not be the most suitable. In fact, for circumstances where a large amount of detail must be preserved, other sensing technologies such as LiDAR are more much applicable. However, for applications with less demand for precision, stereoscopic imaging, also with the imaging system at hand, becomes more suitable. Nevertheless, these would be the more conventional applications for the technology, such as providing rough distance estimates [11], obstacle detection as in [8] or navigation under stable

conditions as in [12].

6 Conclusion and Future Work

UAV-based hyperspectral stereoscopic imaging shows an encouraging potential for accurate 3D terrain mapping. However, given that the stereoscopic aspect of the imaging system at hand is a byproduct of achieving the optimal co-registered hyperspectral cube and the system is not tailored specifically for stereoscopic use, the system comes with a few unconventional traits, some limiting and others providing new aspects to an already well-tested technology.

The baseline of only 75 mm puts a significant limit to the achievable accuracy, even for the most accurate stereo matching algorithms. The developed stereo matching algorithm demonstrated a disparity accuracy down to 0.02 pixels on ideal synthetic data, an accuracy consistent with other unconventional narrow baseline stereo rigs in the litterateur. Despite this, Table 6 reveals how even in the most ideal circumstances, only UAV altitudes of 20 m above ground would yield elevation accuracy comparable to LiDAR data. For circumstances with less demand on the accuracy, data from both UAV altitudes of 20 m and 40 m above ground can provide enough accuracy to reconstruct a rough representation of the imaged scene. A UAV altitude of 60 m, on the other hand, is beyond the limit of rough reconstructions of an imaged scene, with the proposed and developed model.

The hyperspectral characteristics of the imaging system provide new and less explored aspects to stereoscopic imaging. Pushbroom scanning combined with the impressive accuracy of the INS allows an intuitive scheme for precise georeferencing. Further, including several spectral channels from both cameras increased the robustness of the developed model significantly, especially apparent in the low illuminated areas of the imaged scene. The accuracy, however, did not increase notably by including spectral channels outside the overlapping wavelength region of the two cameras, revealed by calculations on data from 60 m UAV flight altitude.

For stereo matching, the phased-based approach of PC function fitting and Fourier-domain plane fitting proved to be both robust and accurate, especially compared to the intensity-based off-shelf SGBM algorithm. Calculating the disparity in two steps, first by PC function fitting, then by Fourier-domain plane fitting, increased the accuracy, as demonstrated when tested with synthetic data. Although the disparity error increased somewhat when applied to real hyperspectral data of the imaged scene, an accuracy of 0.1 pixels, and better, was for the most part maintained for the less complicated areas of the scene at all UAV altitudes.

There are several measures to improve the system. First of all, the baseline must be increased significantly for the developed model to provide accurate elevation estimates comparable to LiDAR technology for UAV altitudes any higher than 20 m above ground. This proves to be necessary even for the most accurate stereo matching algorithms. Concerning the two cameras, the model would most likely benefit from being based around two identical HSI systems with overlapping spectral channels, especially wrt robustness. For stereo matching, phase-based approaches also seem to be the way moving forward. However, the benefits of the Fourier-domain plane fitting approach to stereo matching, in terms of window size, should be exploited to a much larger degree to preserve detail in the scene to a larger extent. At last, the entire developed model should be re-designed and standardised such that it only requires one function call with a set of input parameters, and yields a complete and processed point cloud of the imaged scene as output.

To conclude, the stereoscopic aspect of the imaging system at hand shows potential for UAV-based 3D terrain mapping. As of now, the unconventional narrow baseline puts a fundamental limit on the performance of the system, and the developed model does not replace state-of-the-art LiDAR technology. However, for applications with fewer constraints on the accuracy, the developed model is capable of producing a rough representation of the depicted scene, given UAV altitudes of 20 m and 40 m, in the form of a point cloud where each pixel is georeferenced.

References

[1] Andreas Horpedal Bugge. *UAV-Based Hyperspectral Stereoscopic Imaging for Accurate 3D Terrain Mapping*. Project report in TFE4580. Department of Electronic Systems, NTNU – Norwegian University of Science and Technology, Dec. 2021.

[2] Raju Shrestha. 'Conceiving a Fast and Practical Multispectral-Stereo System'. MA thesis. Gjøvik University College, 2010.

[3] Stian Selbek. 'Multi-Objective Optimisation of Stereo Vision Algorithms'. MA thesis. University of Oslo, 2015.

[4] K. V. Stefanik et al. 'UAV-Based Stereo Vision for Rapid Aerial Terrain Mapping'. In: *GIScience Remote Sensing* (2012).

[5] Vinay Chamola et al. 'A Comprehensive Review of Unmanned Aerial Vehicle Attacks and Neutralization Techniques'. In: *Ad Hoc Networks* (2021).

[6] W. Zang et al. 'Investigating small-scale water pollution with UAV Remote Sensing Technology'. In: *World Automation Congress* (2012).

[7] Tulldahl H. M and H. Larsson. 'Lidar on small UAV for 3D mapping'. In: *Electro-Optical Remote Sensing, Photonic Technologies, and Applications VIII* (2016).

[8] N. Appiah and N. Bandaru. 'Obstacle detection using stereo vision for self-driving cars'. In: 2015.

[9] Brent Schwarz. 'LIDAR Mapping the world in 3D'. In: *Nature Photon 4* (2010).

[10] M. Fahle. 'Why two eyes?' In: *Die Naturwissenschaften* (1987).

[11] M. Okutomi and T. Kanade. 'A Multiple-Baseline Stereo'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1993).

[12] J. Biesiadecki J and M. W. Maimone. 'The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition'. In: *6 IEEE Aerospace Conference Proceedings* (2006).

[13] C. T. Theodore and M. B. Tischler. 'Precision Autonomous Landing Adaptive Control Experiment (PALACE)'. In: *Army Science Conference* (2006).

[14] M. Abdo, V. Badilita and J. Korvink. 'Spatial scanning hyperspectral imaging combining a rotating slit with a Dove prism'. In: *OPTICS EXPRESS* (2019).

[15] H. E. Torkildsen and T. Skauli. 'Full characterization of spatial coregistration errors and spatial resolution in spectral imagers'. In: *Optics Letters* (2014).

[16] Hyspex. *The keystone effect and its influence on classification results*. 2019. URL: https://www.hyspex.com/hyperspectral-imaging/key-quality-parameters/keystone/ (visited on 31/03/2022).

[17] S. Blaaberg et al. 'HySpex ODIN-1024: a new high-resolution airborne HSI system'. In: *Optical Engineering* (2014).

[18] G. Lu and B. Fei. 'Medical hyperspectral imaging: a review'. In: *Journal of Biomedical Optics* (2014).

[19] N. Heide et al. 'REAL-TIME HYPERSPECTRAL STEREO PROCESSING FOR THE GENERATION OF 3D DEPTH INFORMATION'. In: *25th IEEE International Conference on Image Processing (ICIP)* (2018).

[20] G. Lu and B. Fei. 'Medical hyperspectral imaging: a review'. In: *Biomed. Opt. 19* (2014).

[21] J. Qin et al. 'Hyperspectral and multispectral imaging for evaluating food safety and quality'. In: *Journal of Food Engineering* (2013).

[22] *APX-15 UAV*. Trimble. URL: https://www.applanix.com/downloads/products/specs/APX15_UAV.pdf.

[23] T. O. Opsahl, T. V. Haavardsholm and I. Winjum. 'Real-time georeferencing for an airborne hyperspectral imaging system'. In: *Proceedings Volume 8048* (2011).

[24] N. Yokoya N. Miyamura and A. Iwasaki. 'Preprocessing of hyperspectral imagery with consideration of smile and keystone properties'. In: *Proceedings Volume 7857* (2010).

[25] N. Yokoya, N. Miyamurab and A. Iwasakib. 'Preprocessing of hyperspectral imagery with consideration of smile and keystone properties'. In: *Proceedings of SPIE - The International Society for Optical Engineering* (2010).

[26] G. Høye, T. Løke and A. Fridman. 'Method for quantifying image quality in push-broom hyperspectral cameras'. In: *Optical Engineering 54* (2015).

[27] R. Patterson and W. L. Martin. 'Human Stereopsis'. In: *Human Factors* (1992).

[28] A. Martin et al. 'The Numerical Stereo Camera'. In: *Three-Dimensional Machine Perception* (1981).

[29] K. Gong and D. Fritsch. 'DSM Generation from High Resolution Multi-View Stereo Satellite Imagery'. In: *Photogrammetric Engineering Remote Sensing* (2019).

[30] A. Kaehler G. Bradski. *Learning OpenCV 3*. O'Reilly Media, 2016.

[31] Chris Veness. *Calculate distance, bearing and more between Latitude/Longitude points*. URL: https://www.movable-type.co.uk/scripts/latlong.html. (accessed: 10.03.2022).

[32] G. B. West. 'Mean Earth Ellipsoid Determined From SEASAT 1 Altimetric Observationss'. In: *Journal of Geophysical Research* (1982).

[33] H. Mahmoud and N. Akkari. 'Shortest Path Calculation: A Comparative Study for Location-Based Recommender System'. In: *World Symposium on Computer Applications Research* (2016).

[34] G. Llewellyn et al. 'Precise Subpixel Disparity Measurement From Very Narrow Baseline Stereo'. In: *IEEE Transactions on Geoscience and Remote Sensing* (2010).

[35] X. Lin, Y. Liu and W. Dai. 'Study of Occlusions Problem in Stereo Vision'. In: *2008 7th World Congress on Intelligent Control and Automation* (2008).

[36] D. Li and H. Zhao. 'Fast Phase-based Stereo Matching Method for 3D Shape Measureme'. In: *International Symposium on Optomechatronic Technologies* (2010).

[37] L. T. Sach et al. 'A Robust Stereo Matching Method for Low Texture Stereo Images'. In: *IEEE-RIVF International Conference on Computing and Communication Technologies* (2009).

[38] H Hirschmüller. 'Stereo Processing by Semiglobal Matching and Mutual Information'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008).

[39] D. Zha, X. Jin and T. Xiang. 'A real-time global stereo-matching on FPGA'. In: *Microprocessors and Microsystems* (2016).

[40] S. Birchfield and C. Tomasi. 'Local Stereo Matching with Improved Matching Cost and Disparity Refinement'. In: *IEEE MultiMedia* (2014).

[41] J. Ko and Y. Ho. 'Stereo Matching using Census Transform of Adaptive Window Sizes with Gradient Images'. In: *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (2016).

[42] S. Birchfield and C. Tomasi. 'A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling'. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* (1998).

[43] N. Ma et al. 'A Subpixel Matching Method for Stereovision of Narrow Baseline Remotely Sensed Imagery'. In: *Mathematics in Utilizing Remote Sensing Data for Investigating and Modelling Environmental Problems* (2017).

[44] M. Debella-Gilo and A. Kääb. 'Sub-pixel precision image matching for measuring surface displacements on mass movements using normalized cross-correlation'. In: *Remote Sensing of Environment* (2010).

[45] K. Takita et al. 'High-accuracy subpixel image registration based on phase-only correlation'. In: *IEICE Transactions. Fundamentals* (2003).

[46] M. Balci and H. Foroosh. 'Estimating sub-pixel shifts directly from the phase difference'. In: *International Conference on Image Processing* (2005).

[47] Deepa and K. Jyothi. 'A robust and efficient pre processing techniques for stereo images'. In: *2017 International Conference on Electrical, Electronics* (2017).

[48] *HySpex Mjolnir VS-620*. Norsk Elektro Optikk. URL: https://www.hyspex.com/media/2jbdxard/hyspex_mjolnir_vs-620.pdf.

[49] *VLP-32C*. Velodyne Lidar. URL: https://www.mapix.com/wp-content/uploads/2018/07/63-9378_Rev-D_ULTRA-Puck_VLP-32C_Datasheet_Web.pdf.

[50] Kartverket. *Norgeskart*. 2021. URL: http://norgeskart.no (visited on 31/05/2022).

Appendix

A Stereo Matching Python Scripts

```
import spectral.io.envi as envi
import numpy as np
import math
import scipy
import statistics as stats
from skimage.filters import window
from scipy.optimize import curve_fit
import cmath
import cv2
from skimage.restoration import denoise_tv_chambolle
from scipy.signal import lfilter
from scipy.signal import butter, lfilter
from scipy import signal

def loadBands(path,SWIR_start,SWIR_stop,VNIR_start,VNIR_stop):
    print("Loading bands ..")
    SWIR=[]
    VNIR=[]
    for i in range(SWIR_start,SWIR_stop):
        SWIR_envi=envi.open(path+'headerFile.hdr', path+'imgFile.img').read_band(i)
        SWIR.append(SWIR_envi)
        print("SWIR band nr: ",i, " loaded")
    print("Loading VNIR cube ..")
    for i in range(VNIR_start,VNIR_stop):
        VNIR_envi=envi.open(path+'headerFile.hdr', path+'imgFile.img').read_band(i)
        VNIR.append(VNIR_envi)
        print("VNIR band nr: ",i, " loaded")
    return SWIR, VNIR

def shiftImage(img, xShift):
    try:
        img=scipy.ndimage.shift(img, (0, xShift))
        return img
    except RuntimeError:
        img=scipy.ndimage.shift(img, xShift)
        return img

def adjustImage(imgL,imgR, row_nr,ins, expected_alt):
  sensorModel=np.loadtxt('sensormodelPath',usecols=range(1,2))
  x_cord1=np.arange(0, 620,0.0001)
  x_cord=np.arange(0, 620)
  newSensorModel=np.interp(x_cord1,x_cord,sensorModel)
  k=400
  angle_1=newSensorModel[k]
  angle_2=math.atan(math.tan(angle_1)-0.075/expected_alt)
  excepted_shift=(angle_1-angle_2)/(0.54*10**(-3))
  altitude=ins[row_nr]

  for i in range(0,len(imgL)):
    alt_difference=altitude-ins[row_nr+i]
    new_alt=expected_alt+alt_difference
    angle_3=math.atan(math.tan(angle_1)-0.075/new_alt)
    new_shift=(angle_1-angle_3)/(0.54*10**(-3))
    shift_difference=excepted_shift-new_shift
    imgL[i]=shiftImage(imgL[i],-shift_difference)
    imgR[i]=shiftImage(imgR[i],-shift_difference)
```

```
57
58      return imgL,imgR
59
60
61  def returnLinear(a,u):
62      return a*u
63
64  def returnSinc(x, k):
65      return np.sinc(x-k)
66
67  def returnGauss(x,k):
68      return stats.norm.pdf(x,k)
69
70
71  def calcPC(imgL,imgR):
72      x_len=0.5*len(imgL[0])
73      y_len=0.5*len(imgL.T[0])
74
75      hanning=window('hann', imgL.shape)
76      imgL=imgL*hanning
77      imgR=imgR*hanning
78      DFT_L=np.fft.fft2(imgL)
79      DFT_R=np.fft.fft2(imgR)
80
81      H=np.ones(y_len, x_len)
82      zeroes= np.zeros(imgL.shape)
83      zeroes[:H.shape[0],:H.shape[1]] = H
84      Q=DFT_L*np.conjugate(DFT_R)/(abs(DFT_L*np.conjugate(DFT_R)))*zeroes
85
86      PC=(np.fft.ifft2(Q)).real
87      return PC
88
89  def detectShiftPC(imgL,imgR, min_disp,max_disp):
90      PC=calcPC(imgL,imgR)
91      shift=np.argmax(PC[0][min_disp:max_disp])+min_disp
92      x_values=np.arange(0,len(PC[0]),1)
93      try:
94          fit_parameters=curve_fit(returnGauss, x_values[shift-3:shift+4],
95          PC[0][shift-3:shift+4], p0=[shift])
96      except RuntimeError:
97          print("Sinc fitting ..")
98          try:
99              fit_parameters=curve_fit(returnSinc, x_values[shift-3:shift+4],
100             PC[0][shift-3:shift+5], p0=[shift])
101         except RuntimeError:
102             print("No fit found")
103             return -1,  PC[0], shift
104     PC_fit=fit_parameters[0][0]
105     if PC_fit < min_disp or PC_fit > max_disp:
106         return -1,  PC[0], shift
107     return PC_fit, PC[0], shift
108
109
110 def fourierDomainShiftDetector(imgL,imgR, x_size, x_window):
111
112     hanning=window('hann', imgL.shape)
113     imgL=imgL*hanning
114     imgR=imgR*hanning
115
116     DFT_L=np.fft.fft2(imgL)
117     DFT_R=np.fft.fft2(imgR)
118
119     Q=DFT_L*np.conjugate(DFT_R)/(abs(DFT_L*np.conjugate(DFT_R)))
```

```
120
121         phases=[]
122         for j in range(int(len(imgL)/x_window)):
123             line=[]
124             for i in range(int(len(imgL[0])/x_window)):
125                 line.append(cmath.phase(Q[j][i]))
126             phases.append(line)
127         phases=np.asarray(phases)
128         phases_unwrapped=np.apply_over_axes(np.unwrap, phases, np.arange(len(phases.shape)))
129         phases_unwrapped_mean=np.average(phases_unwrapped ,axis=0)
130
131         x_values=np.arange(0,int(len(imgL[0])/x_window),1)
132         a = np.polyfit(x_values, phases_unwrapped_mean, 1)
133
134         return -a[0]*x_size/(2*np.pi)
135
136     def SGBMCv2(imgL,imgR):
137         window_size = 11
138         dispNumb=16
139         left_matcher = cv2.StereoSGBM_create(
140             minDisparity=0,
141             numDisparities=dispNumb,
142             blockSize=window_size,
143             P1=8 * 3 * 15**2,
144             P2=32 * 3 * 15**2,
145             disp12MaxDiff=5,
146             uniquenessRatio=10,
147             speckleWindowSize=100,
148             speckleRange=1,
149             preFilterCap=10,
150             mode=cv2.STEREO_SGBM_MODE_HH
151         )
152
153
154         right_matcher = cv2.ximgproc.createRightMatcher(left_matcher)
155         lmbda = 8000
156         sigma =1.5
157         wls_filter = cv2.ximgproc.createDisparityWLSFilter(matcher_left=left_matcher)
158         wls_filter.setLambda(lmbda)
159
160         wls_filter.setSigmaColor(sigma)
161         displ = left_matcher.compute(imgL, imgR)
162         dispr = right_matcher.compute(imgR, imgL)
163         displ = np.int16(displ)
164         dispr = np.int16(dispr)
165         disparity= (wls_filter.filter(displ, imgL, None, dispr))
166         filteredImg = (wls_filter.filter(displ, imgL, None, dispr))
167         filteredImg = cv2.normalize(src=filteredImg, dst=filteredImg,
168         beta=0, alpha=255, norm_type=cv2.NORM_MINMAX);
169         filteredImg = np.uint8(filteredImg)
170         np.savetxt('C:/Users/andre/Desktop/TestData/disparity_values.txt',disparity/16)
171         return disparity/16
172
173
174     def fillHole(disparity,y_dir,x_dir):
175         new_disp=[]
176         for y in range(len(disparity)):
177             line=[]
178             for x in range(len(disparity[0])):
179                 value=disparity[y][x]
180                 if value == -1:
181                     y_min=y-y_dir
182                     y_max=y+y_dir
```

```
                x_min=x-x_dir
                x_max=x+x_dir
                if y_min < 0:
                    y_min=0
                if y_max >=len(disparity):
                    y_max = len(disparity) -1
                if x_min < 0:
                    x_min=0
                if x_max >= len(disparity[0]):
                    x_max = len(disparity[0]) -1
                newValues=[]
                newValues=disparity[y_min:y_max, x_min:x_max]
                for i in range(y_dir):
                    newValues = np.delete(newValues, np.where(newValues == -1))
                new_value=[]
                new_value=np.mean(newValues)
                line.append(new_value)
                continue
            line.append(value)
        new_disp.append(line)
    new_disp=np.asarray(new_disp)
    return new_disp

def denoiseTotalVariation(disparities_raw,weight):
    x_std = disparities_raw
    data_denoised = denoise_tv_chambolle(x_std, weight, eps=0.00001)
    return data_denoised


def iirFilter(sig,k):
    b = [1.0 / k] * k
    a = 1
    final_sig = lfilter(b,a,sig)
    return final_sig

def buildDisparityMap(disparity, windowSize_x, windowSize_y):
    disparityMap=[]
    for y in range(len(disparity)):
        line=[]
        try:
            x_lenght=len(disparity[0])
        except TypeError:
            x_lenght=1
        for x in range(x_lenght):
            try:
                value=disparity[y][x]
            except IndexError:
                value=disparity[y]
            for i in range(windowSize_x):
                line.append(value)
        for j in range(windowSize_y):
            disparityMap.append(line)
    return np.asarray(disparityMap)


def lowPass(input,cutoff):
    b, a = butter(5, cutoff, btype='low')
    filtered_signal=signal.filtfilt(b,a,input)
    return abs(filtered_signal)
```

```
246  def lowFilterPass2D(disparityMap, cutoff_x, cutoff_y):
247      filtered_x=[]
248      for disparities in disparityMap:
249          filtered_x.append(lowPass(disparities,1/cutoff_x))
250      filtered_x=np.asarray(filtered_x)
251      filtered_yx=[]
252      for disparities in filtered_x.T:
253          filtered_yx.append(lowPass(disparities,1/cutoff_y))
254      return (np.asarray(filtered_yx)).T
```

B 3D Terrain Mapping and Georeferencing Python Scrips

```python
import numpy as np
import math
import laspy

def computeAltitude(disparity, y_size):
    insData = np.loadtxt('INSfile.txt', usecols=range(1,7))
    pixel_origo=(np.vstack((insData.T[0],insData.T[1],insData.T[2]))).T)
    pixel_origo=pixel_origo[:3800].tolist()
    altitude=[]
    sensorModel=np.loadtxt('sensorModel.txt',usecols=range(1,2))
    x_cord=np.arange(0, 620)
    x_cord_interpolated=np.arange(0, 620,0.0001)
    new_sensorModel=np.interp(x_cord_interpolated,x_cord,sensorModel)

    for i in range(len(disparity.T[0])):
        z=i//y_size
        line=[]
        for j in range(0,len(disparity[0])):
            dispartiyShift=disparity[i][j]*10000
            k=j*10000
            angle=new_sensorModel[k]
            if j > 16  and disparity[i][j]>0:
                height = 0.075/(np.tan(angle)-np.tan(new_sensorModel[int(k-dispartiyShift)]))
                length_x_dir=height*math.tan(np.abs(angle))
                height=length_x_dir*math.tan(abs(angle))+height
                new_z= pixel_origo[z][2]-height
                line.append(new_z)
        altitude.append(line)
    return np.asarray(altitude)


def analyzeTrend(data, window):
    return np.gradient(data,window)


def trendAdjuster(disparity, altitude, insData,weight):
    new_altitudes=[]
    insData_trend=analyzeTrend(insData,1)
    for j in range(0,len(altitude[0])):
        disparity_trend=analyzeTrend(disparity.T[0],1)
        altitude_trend=analyzeTrend(altitude.T[0],1)
        zeroes=[0]
        new_altitude=[]
        for i in range(1,len(altitude)):
            if insData_trend[i] > 0 and altitude_trend[i] < 0 and disparity_trend[i] < 0:
                zeroes.append(zeroes[i-1]-insData_trend[i-1]*weight)
            elif insData_trend[i] <= 0 and altitude_trend[i] >= 0 and disparity_trend[i] >= 0:
                zeroes.append(zeroes[i-1]-insData_trend[i-1]*weight)
            else:
                zeroes.append(zeroes[i-1])
        zeroes=np.asarray(zeroes)
        new_altitude=altitude.T[j] - zeroes
        new_altitudes.append(new_altitude)
    new_altitudes=np.asarray(new_altitudes)
    return new_altitudes.T
```

```
60  def calculatePointCloud(disparity, elevation):
61      insData = np.loadtxt('INSfile.txt', usecols=range(1,7))
62      sensorModel=np.loadtxt('sensormodel.txt',usecols=range(1,2))
63
64      pixel_origo=(np.vstack((insData.T[0],insData.T[1],insData.T[2])).T)
65
66      pixel_origo=pixel_origo.tolist()
67      roll_pitch_heading=(np.vstack((insData.T[3],insData.T[4],insData.T[5])).T)
68      roll_pitch_heading=roll_pitch_heading.tolist()
69      x_cord=np.arange(0, 620)
70      x_cord_interpolated=np.arange(0, 620,0.0001)
71      new_sensorModel=np.interp(x_cord_interpolated,x_cord,sensorModel)
72
73      points=[]
74      R=6378.1*10**3
75
76      print("Calculating pointcloud ...")
77      for i in range(len(elevation.T[0])):
78          for j in range(0,len(elevation[0])):
79              dispartiyShift=disparity[i][j]*10000
80              k=j*10000
81              angle=new_sensorModel[k]
82              #if j > 16  and disparity[i][j]>0:#max disp her
83
84              height = 0.075/(np.tan(angle)-np.tan(new_sensorModel[int(k-dispartiyShift)]))
85              length_x_dir=height*math.tan(np.abs(angle))
86              height=length_x_dir*math.tan(abs(angle))+height
87
88              lat1= math.radians(pixel_origo[i][1])
89              lon1 = math.radians(pixel_origo[i][0])
90              if j <= 310:
91                  bearing=math.radians(roll_pitch_heading[i][2]+90)
92              elif j>310:
93                  bearing=math.radians(roll_pitch_heading[i][2]-90)
94              lat2 = math.asin(math.sin(lat1)*math.cos(length_x_dir/R) + math.cos(lat1)*math.sin(length_x_dir
95              lon2 = lon1 + math.atan2(math.sin(bearing)*math.sin(length_x_dir/R)*math.cos(lat1),
96                  math.cos(length_x_dir/R)-math.sin(lat1)*math.sin(lat2))
97              new_z= elevation[i][j]
98              lon2=math.degrees(lon2)
99              lat2=math.degrees(lat2)
100             new_point=[lon2,lat2, new_z]
101             points.append(new_point)
102
103     points=np.asarray(points)
104     print("Creating .las ...")
105     createLas(points)
106
107
108 def createLas(dataPoints):
109
110     my_data=np.vstack((dataPoints.T[0],dataPoints.T[1],dataPoints.T[2])).T
111
112     header = laspy.LasHeader(point_format=7, version="1.4")
113     header.offsets = np.min(my_data, axis=0)
114     header.scales = np.array([0.0000000001, 0.0000000001, 0.00001])
115
116     with laspy.open("lasFile.las", mode="w", header=header) as writer:
117         point_record = laspy.ScaleAwarePointRecord.zeros(my_data.shape[0], header=header)
118         point_record.x = my_data[:, 0]
119         point_record.y = my_data[:, 1]
120         point_record.z = my_data[:, 2]
121         writer.write_points(point_record)
```

C Synthetic Data Generating Python Scrips

```python
import scipy
import numpy as np
from PIL import Image, ImageEnhance

def shiftImage(img, xShift):
    try:
        img=scipy.ndimage.shift(img, (0, xShift))
        return img
    except RuntimeError:
        img=scipy.ndimage.shift(img, xShift)
        return img

def addShotNoise(img,amount):
  noisy_signal=np.random.poisson(amount, (len(img), len(img[0])))
  noisy_img=img+noisy_signal
  return np.asarray(noisy_img)

def addPeriodicNoise(img, freq):
  dataPoints_periodic=[]
  for i in range(len(img[0])):
    dataPoints_periodic.append(50*np.cos(i*np.pi*freq))
  noisy_img = img + dataPoints_periodic
  return np.asarray(noisy_img)


def brightenImage(imgPaht, enhancement):
  image = Image.open(imgPaht)
  img_enhancer = ImageEnhance.Brightness(image)
  return img_enhancer.enhance(enhancement)

def convertCV2Data(img):
  img=img-img.min()
  img=img/img.max()*255
  return np.uint8(img)
```

D Results

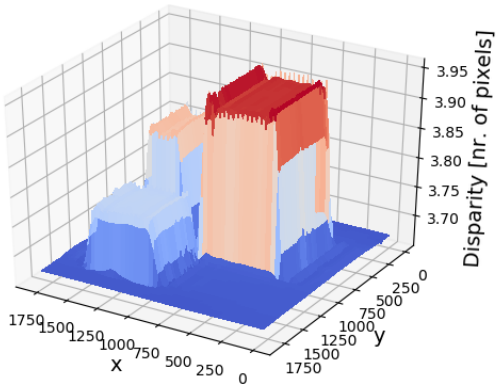


Figure 74: Disparity map resulting from PC function fitting individually with a Gaussian function. The ground truth disparity map is presented to the right in Figure 33 for comparison.

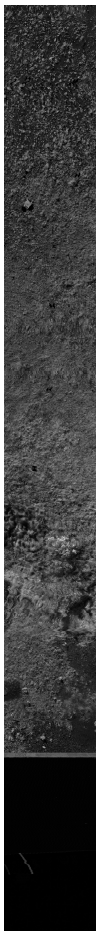


Figure 75: Spatially misregistered image captured by Camera 1 from a UAV altitude of 20 m above ground.

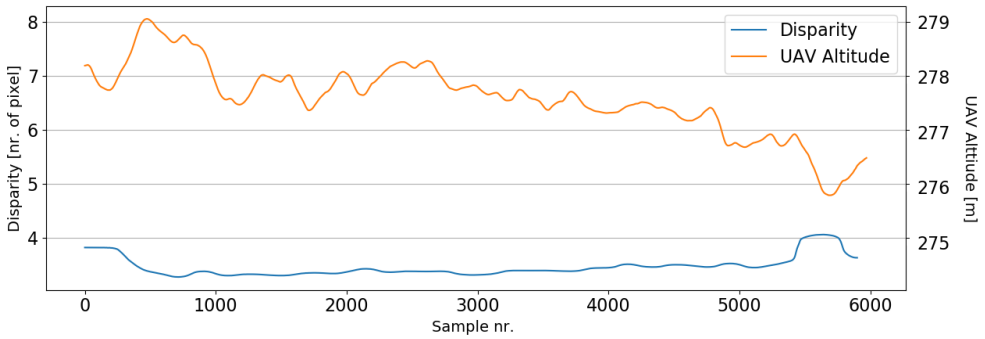


Figure 76: Along-track disparity profile and the corresponding INS altitude. Very little correlation between the disparity profile and the altitude was observed.

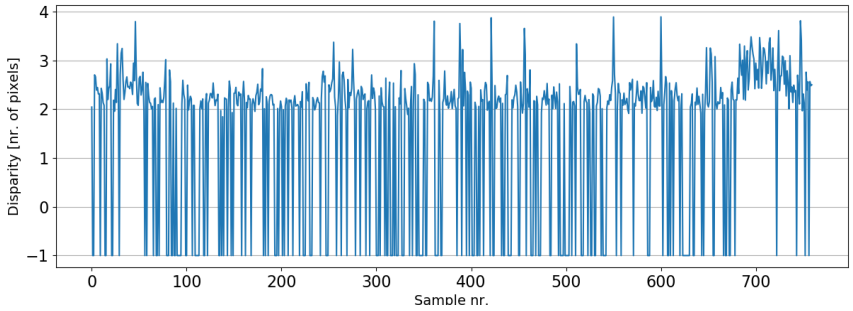


Figure 77: Raw along-track disparity profile from a UAV altitude of 60 m. Derived with a window size of 62×20 pixels^2 and 15 additional bands outside the overlapping wavelength region. The disparity profile displays a large number of disparity holes.

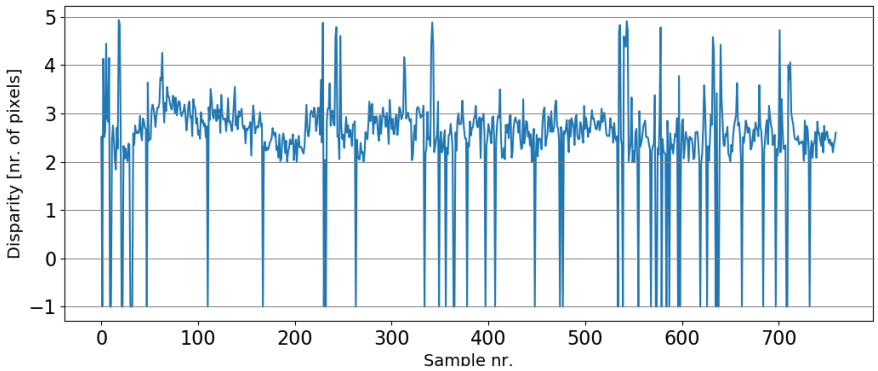


Figure 78: Disparity profile of more complicated terrain from 60 m UAV altitude. The disparity profile displays a large number of disparity holes as well as much fluctuation from one sample to the next.

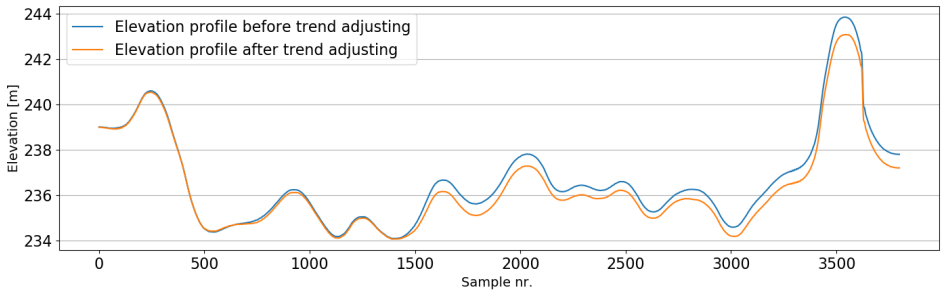
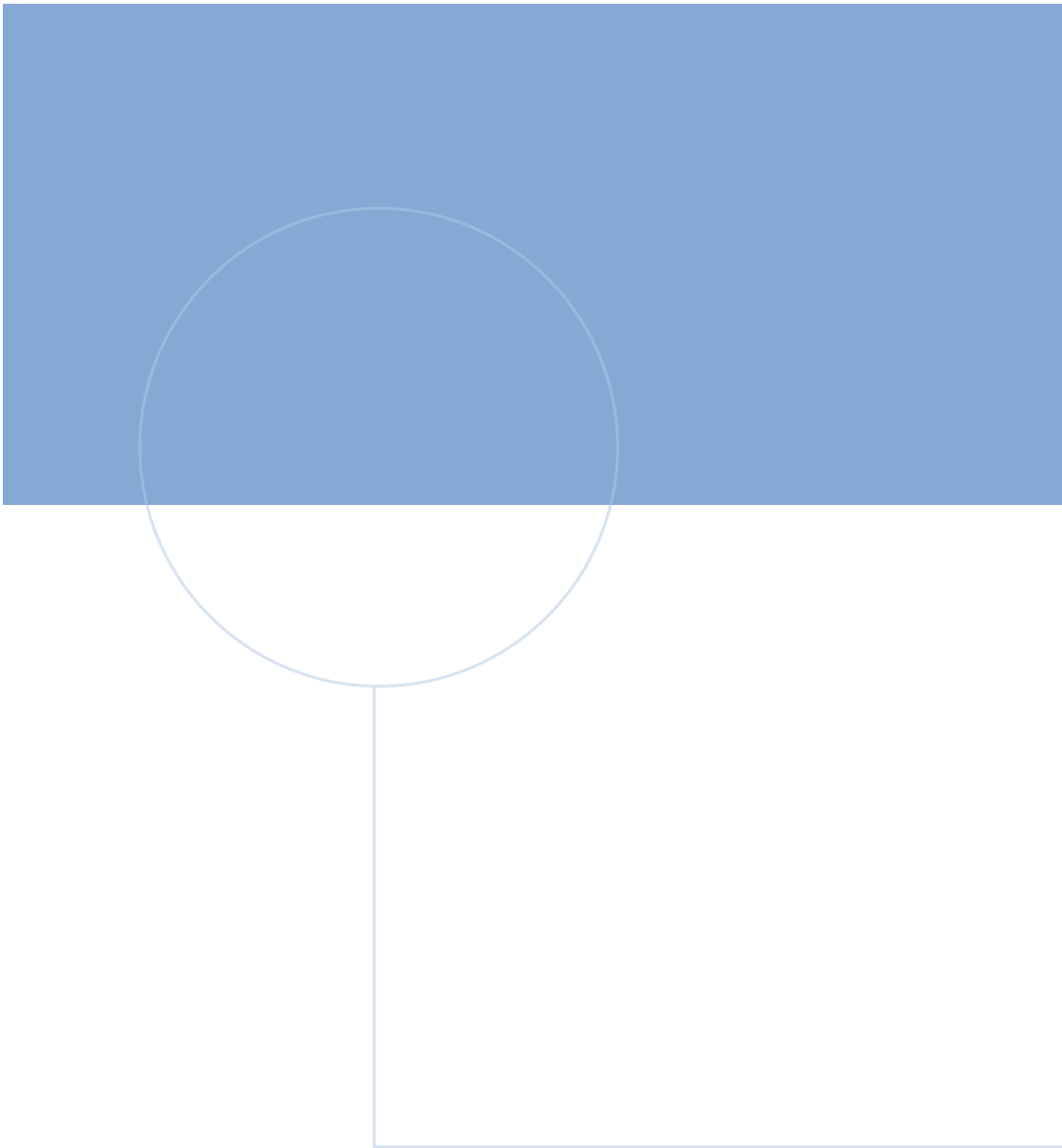


Figure 79: Elevation profiles from 60 m UAV altitude before and after trend adjusting. No significant increase in performance was observed by trend adjusting.



NTNU

Norwegian University of
Science and Technology