ROYAUME DU MAROC
INSTITUT AGRONOMIQUE
ET VÉTÉRINAIRE HASSAN II

المملكة المغربية
معهد الحسن الثاني
للزراعة والبيطرة

Projet de Fin d'Etudes présenté pour l'obtention du diplôme d'Ingénieur en Topographie

Localisation, Back-Projection and Fusion of LWIR Hyperspectral Data from FTIR Imaging Sensors

Présenté et soutenu publiquement par :
FATIHI Ayoub

Jury :

Pr. AIT EL KADI Kenza	**(Présidente)**	**IAV HASSAN II**
Pr. SEBARI Imane	**(Rapporteuse)**	**IAV HASSAN II**
Pr. YAAGOUBI Reda	**(Rapporteur)**	**IAV HASSAN II**
Dr. LORENZ Sandra	**(Rapporteuse)**	**HZDR - HIF**
Dr. THIELE T. Samuel	**(Rapporteur)**	**HZDR - HIF**

Mars 2023

Adresse : Madinat Al Irfane, B.P. 6202. Rabat – Maroc
Tél : (00 212) 0537 77 17 58/59
Fax : (00 212) 0537 77 58 45
Site web : http://www.iav.ac.ma

العنوان: ص. ب 6202 الرباط المعاهد الرباط – المغرب
الهاتف: 59 / 58 17 77 0537 (00 212)
الفاكس: 45 58 77 0537 (00 212)
موقع الأنترنت: http://www.iav.ac.ma

بِسْمِ اللهِ الرَّحْمَنِ الرَّحِيمِ

﴿وَعَلَّمَكَ مَا لَمْ تَكُن تَعْلَمُ وَكَانَ فَضْلُ اللهِ عَلَيْكَ عَظِيمًا﴾

سورة النساء – آية رقم ١١٣

In the Name of Allah, the Most Beneficent, the Most Merciful
"*Allah (The God) has taught you what you did not know before, and great is Allah's grace upon you.*"
The Holy Quran, Chapter 4, Verse 113.

To my beloved parents Mina and Abdellah, who have always been my source of inspiration, encouragement, and unwavering support throughout my academic journey. Thank you for your endless love, sacrifices, and guidance. I dedicate this thesis to you with utmost gratitude and appreciation.

To my dear brothers Yassine and Younes and sister Aya, who have been my constant companions and pillars of strength. Your encouragement and motivation have always pushed me to do my best. This achievement is as much yours as it is mine, and I dedicate it to you with love and gratitude.

To my dear friend Oussama, who has been with me through thick and thin, providing me with moral support, guidance, and motivation. I dedicate this thesis to you with sincere appreciation and gratitude.

To my dear friend Mohammed, your generosity and kindness have not gone unnoticed, and I am inspired by your compassion and your commitment to helping others. You have truly made a difference in my life, and I am so grateful for your friendship and your presence.

To my dear roommate Elmehdi, I appreciate all the little things you do to make our space feel like a home. Whether it's cooking together, or having a good conversation, I always feel at ease and happy when we're together. Thank you for making this living arrangement such a positive and memorable experience.

I would also like to express my appreciation to Abdessamad, my childhood friend, for his unwavering support and encouragement. Your friendship has been a constant source of inspiration and motivation.

I extend my deepest gratitude to "Ndox" group for not only providing me with the opportunity to conduct labs and field work but also for the invaluable learning experience I gained from working with such a talented and dedicated team.

Finally, I would like to express my appreciation to the study group I was part of for their assistance and help. Your dedication and guidance have helped me to succeed in my academic pursuits. Thank you for being a part of my journey.

Acknowledgements

I am grateful to those who have supported and inspired me, without whom this work would not have been possible. Their unwavering guidance and encouragement has been a constant source of motivation.

I am particularly thankful to Professor Sebari Imane and Professor Yaagoubi Reda, who have provided invaluable guidance and mentorship throughout my academic journey. I am also grateful to Dr. Lorenz Sandra and Dr. Thiele Samuel at HIF HZDR for their support, guidance and contribution to this project.

I would also like to extend my gratitude to Yuleika Madriz, Erik Herrmann, and Junaidh Shaik Fareedh for their valuable input and assistance with the fieldwork.

I extend my warmest appreciation to each member of the jury for their time and efforts in reviewing this work.

Lastly, I would like to acknowledge the invaluable teachings of my professors and the scientific team at IAV HASSAN II, who have provided me with a wealth of knowledge and expertise.

Abstract

The use of high-resolution images and data acquired from terrestrial or aerial platforms enables detailed examination of features on the earth's surface, such as mineral deposits, geomorphological formations, and geological structures. This information is used to map and study the subsurface, understand geodynamic processes, and assess environmental impact. Hyperspectral remote sensing uses the electromagnetic spectrum reflected or emitted by objects on the Earth's surface for various applications, including mineral identification and analysis of the distribution and properties of materials. However achieving an accurate representation of the observed target requires the fusion and co-registration of data obtained from various sensors due to the different acquisition methods of each portion of the electromagnetic spectrum.

This study presents a semi-automatic co-registration workflow for the integration of geological close-range remote sensing data. The method is designed to overcome the challenges of hypercloud formation, where multiple sources of hyperspectral data are collected and integrated in 3D space. The methodology involves a camera calibration process to eliminate initial distortions, followed by stitching of small Long-Wave Infrared hypercubes obtained from a Hyper-cam device to form a hyper-mosaic. The next step involves matching and transformation to establish an affine transformation between the hyper-mosaic and an RGB image, its quality was evaluated by comparing the differences between the coordinates of selected points and their corresponding transforms. Next a Structure-from-Motion workflow is carried out to generate a point cloud and the determination of the RGB frame's camera position and orientation. This then allows back-projection can be performed onto the point cloud to generate a hypercloud. The back-projection process was evaluated through the calculation of the three-dimensional distance between selected points and their respective projection. An average discrepancy of 3.56 cm was achieved. The hypercloud can be filled with data from other regions of the electromagnetic spectrum and/or used to produce mineral maps and other geologically relevant products.

Keywords: co-registration, LWIR, hypercloud, Telops hyper-cam, SfM, homography, affine transformation

Résumé

L'utilisation d'images et de données à haute résolution acquises à partir de plates-formes terrestres ou aériennes permet d'examiner en détail les caractéristiques de la surface terrestre, telles que les gisements minéraux, les formations géomorphologiques et les structures géologiques. Ces informations sont utilisées pour cartographier et étudier le sous-sol, comprendre les processus géodynamiques et évaluer l'impact sur l'environnement. La télédétection hyperspectrale utilise le spectre électromagnétique réfléchi ou émis par des objets à la surface de la Terre pour diverses applications, notamment l'identification des minéraux et l'analyse de la distribution et des propriétés des matériaux. Toutefois, pour obtenir une représentation précise de la cible observée, il faut fusionner et aligner les données obtenues à partir de divers capteurs en raison des différentes méthodes d'acquisition de chaque partie du spectre électromagnétique.

Cette étude présente un flux de travail semi-automatique de co-registration pour l'intégration de données de télédétection géologique à courte portée. La méthode est conçue pour surmonter les défis de la formation d'hypernuages, où de multiples sources de données hyperspectrales sont collectées et intégrées dans l'espace 3D. La méthodologie comprend un processus de calibration de la caméra pour éliminer les distorsions initiales, suivi de l'assemblage de petits hypercubes infrarouges à ondes longues obtenus à partir d'un dispositif Hyper-cam pour former une hyper-mosaïque. L'étape suivante consiste à établir une transformation affine entre l'hyper-mosaïque et une image RVB. Sa qualité a été évaluée en comparant les différences entre les coordonnées des points sélectionnés et leurs transformations correspondantes. Ensuite, un flux de travail "Structure-from-Motion" est exécuté pour générer un nuage de points et la détermination de la position et de l'orientation de la caméra de l'image RVB. Cela permet ensuite d'effectuer une rétroprojection sur le nuage de points afin de générer un hypernuage. Le processus de rétroprojection a été évalué par le calcul de la distance tridimensionnelle entre les points sélectionnés et leur projection respective. Un écart moyen de 3,56 cm a été obtenu. L'hypernuage peut être complété par des données provenant d'autres régions du spectre électromagnétique et/ou utilisé pour produire des cartes minérales et d'autres produits géologiques pertinents.

Mots-clés : co-registration, LWIR, hypernuage (hypercloud), Telops hyper-cam, SfM, homographie, transformation affine.

Short Contents

Contents

List of Figures

List of Tables

Listings

Acronyms

2D Two-Dimensional

3D Three-Dimensional

ADC Analog Digital Conversion

CV Computer Vision

EM ElectroMagnetic

fov Field of view

FPA Focal Plane array

FTIR Fourier Transform Infrared

GNSS Global Navigation Satellite Systems

HR High-Resolution

HSI HyperSpectral Imaging

IMU Inertial measurement unit

LiDAR Light Detection And Ranging

LR Low-Resolution

LWIR Long-Wave Infrared

MPD Maximum Path Difference

MVS Multi-View Stereo

NESR Noise Equivalent Spectral Radiance

ORB Oriented FAST and Rotated BRIEF

OPD Optical Path Difference

RGB Red Green Blue

RGB-D Red Green Blue – Depth

SAR Synthetic Aperture Radar

SIFT Scale-Invariant Feature Transform

SNR Signal to Noise Ratio

SfM-MVS Structure from Motion — Multi-View Stereo

ZPD Zero Path Difference

Chapter 1

INTRODUCTION

Imagine you're standing on top of a mountain, looking down at the vast landscape below. From your vantage point, you can see the intricate patterns of farms, forests, and towns. You can see the meandering paths of rivers, the glistening expanse of lakes, and the rugged contours of mountains. But what if you could see more? What if you could see beyond the visible, into the infrared? Then you could infer the health of crops, the presence of minerals, or the effects of pollution. Meet remote sensing.

Remote sensing is defined as "the measurement or acquisition of information of some property of an object or phenomena, by a recording device that is not in physical or intimate contact with the object or phenomenon under study, e.g., the utilization at a distance" [1]. This data is collected using a variety of platforms, including satellites, aircraft, and ground-based sensors [2]. Remote sensing allows scientists to study and monitor the Earth's resources, environment, and climate in a cost-effective and efficient manner [2].

Close-range remote sensing, also known as proximal remote sensing, involves the use of sensors that are in close proximity to the objects being studied [3]. This type of remote sensing is often used for detailed observations and measurements of small areas, such as crops, forests, and geological features.

Radiometry is a branch of remote sensing that focuses on measuring the amount of radiation emitted by an object in a particular part of the electromagnetic spectrum [3]. This radiation can be in the form of visible light, infrared, or other wavelengths. Hyperspectral remote sensing is a form of radiometry that captures a wide range of wavelengths, resulting in high spectral resolution data that can be used to identify and classify different materials and substances based on their unique spectral signatures [3].

1.1 Motivation

Geological close-range remote sensing focuses on the acquisition and analysis of high-resolution images and data from terrestrial or aerial platforms, aimed at studying the geological features and processes of a given area. It allows for a more detailed examination of important outcrops, including mineral deposits, geomorphological formations, and other geological structures [4]. This enables geologists and earth scientists to map and study the subsurface features, understand the geodynamic processes, and assess the environmental impact on a particular area with greater accuracy. In essence, geological close range remote sensing combines the science of geology with the power of remote sensing technology, opening up a new frontier in the exploration and understanding of the Earth's complex geological systems [4].

The use of hyperspectral remote sensing allows for detailed analysis of the electromagnetic spectrum reflected by objects on the Earth's surface. This information can be used for a variety of applications, including mineral identification and analysis of the distribution and composition of

materials on the Earth's surface [5]. However, often times multiple sources of hyperspectral data are collected from sensors using different technologies from different viewing angles, leading to the need for integration, fusion, or co-registration in order to effectively analyze the information. Performing the integration in the 3D instead of the 2D image space in advantageous in terms of reducing occlusion and distortion issues, facilitating the correction of illumination and atmospheric effects in the hyperspectral data, and accurately interpreting the geometry of geological structures [6]. This process results in a hypercloud which is a geometrically correct, spatially 3-dimensional representation of the hyperspectral datacubes.

The end-result of this process is a detailed analysis of the electromagnetic spectrum at various locations on the Earth's surface, which can provide valuable insights into the distribution and properties of materials.

1.2 Problem Framing

3D mapping of geological sites is an essential tool for characterizing the spatial distribution of different rock types, identifying geological structures, estimating mineral resources, and monitoring changes over time [4]. This can be further enhanced by capturing the surface's response to various portions of the electromagnetic spectrum.

This information will be captured in a hypercloud, i.e. a three-dimensional representation of geometrically accurate hyperspectral data [7]. The hypercloud serves as a powerful tool in mineral exploration for comprehending geological processes such as mineralization and improving the accuracy of surface and subsurface mapping.

The acquisition of data from different portions of the electromagnetic spectrum involves the use of multiple sensors, due the variability in the employed sensing methods [3]. As a result, data is collected from disparate sources, necessitating the process of co-registration and fusion to create a unified and coherent product. The co-registration and fusion of data from various sensors enable the creation of a single integrated dataset. This allows for the exploitation of complementary information from multiple sources and enhances the accuracy and resolution of the resulting products. Which can provide a more comprehensive understanding of the target being studied.

In this work, the main objective is the co-registration of the Long-wave Infrared data from the Telops Hyper-Cam sensor to a 3D point cloud to get a hypercloud. The hyperspectral Long-wave Infrared data acquired by Fourier-Transform Infrared imaging sensor poses a particular challenge for co-registration, mainly because of the low field of view of the sensor resulting in very small captured images, and the complex and dangerous 3D sub-vertical target geometry, which translates into the difficulty of maintaining a spatial overlap between consecutive images. The obtained images do not provide any 3D information due to the static position of the sensor and the lack of GNSS receivers. One other thing is the scarcity of visual similarities between various ranges of the electromagnetic spectrum especially the Long-Wave InfraRed and visible ranges, makes working with Long-wave Infrared data acquired by Fourier-Transform Infrared imaging sensor very challenging.

Breakdown of the larger objective into smaller objectives includes:

- Data processing and correction
- Exploration of various co-registration techniques
- Exploration of deep networks for some co-registration tasks
- 3D point cloud generation using Structure-from-Motion (SfM)
- Implementation of a workflow for co-registering LWIR hyperspectral data to the hypercloud.

1.3 Thesis Outline

The thesis is divided into 6 chapters, starting with this introductory chapter. Then the literature review chapter that covers the topics of the electromagnetic spectrum, data acquisition and pro-

cessing, and mineral exploration. The methodology chapter details the proposed approach, the equipment, and the software used in the research. The implementation and results chapter highlights the fieldwork and the results and how they were obtained, followed by the discussion chapter, and the last chapter present conclusions drawn from the research and recommendations to further advance this work.

1.4 Host Institute

The Helmholtz Institute Freiberg for Resource Technology (HiF) is a research institute dedicated to finding sustainable solutions for the efficient sourcing and use of metalliferous and mineral raw materials. It was founded in 2011 as part of the German Resource Strategy, with the goal of providing solutions for the sustainable use of these resources. HIF is funded by the German Federal Government (90%) and the Free State of Saxony (10%), and is a member of the Helmholtz-Zentrum Dresden-Rossendorf (HZDR), as well as a strategic partner of TU Bergakademie Freiberg. It is also a member of the EIT RawMaterials community, and has a staff of more than 140 people from over 25 different countries.

The researchers at HiF are focused on the opportunities and limits of the Circular Economy, and are working to develop both resource- and energy-efficient technologies related to the metalliferous raw material cycle. This includes innovative processes for exploration, processing, metallurgy, and recycling. To achieve this, they are using interdisciplinary approaches that integrate modeling and valuation of recovery processes, system-integrated metal production, and resource analytics. From a materials standpoint, HiF is particularly interested in high-tech metals like indium, gallium, germanium, and rare earth elements, as well as complex materials from both primary and secondary sources.

The HZDR, of which HiF is a part, conducts research in the areas of energy, health, and matter. The aims of HiF include developing new technologies for the long-term supply of mineral and metalliferous raw materials from domestic and global sources, contributing to global environmental protection through material and energy efficiency, establishing long-term economic relations with resource-based countries, and training a new generation of highly qualified scientists and engineers for industry and academia.

Chapter 2

LITERATURE REVIEW

Contents

> "Geologists seem to have rosy prospects in remote sensing for the next decade. This period is likely to be one of consolidation rather than innovation, giving the majority of geologists the time to get to grips with what has been happening over the last three decades in geological remote sensing research, to apply the new data to exciting new geological problems instead of repeatedly pawing over tiny test areas, and to catch up with their colleagues in other fields."
>
> *Steve Drury [8]*

In 2004, Steve Drury [8] made this observation that has now undergone miles, as we will explore in the following discussion.

The investigation of geological outcrops is a domain of significant significance within the geosciences, primarily due to its capacity to amass geospatial information, quantify geometric parameters, and create maps of mineralogy and lithology [9]. Remote sensing is a very important tool in achieving that, and in geology an exploration program involves four stages [9]: (1) prospecting, (2) regional exploration, (3) detailed exploration, and (4) mine exploration.

Prior to delving further into mineral exploration, a review of the literature will be conducted to provide an overview of the current state of knowledge in the field and to guide our subsequent investigations.

2.1 Remote sensing

Remote sensing is a powerful tool for studying the Earth's surface from a distance, without making physical contact with the object of interest.

Close-range remote sensing involves the use of sensors at relatively short distances, such as from an unmanned aerial vehicle or ground-based platform. This type of remote sensing is useful for collecting high-resolution data and for studying small-scale features that may not be visible from satellite sensors. Examples of close-range remote sensing applications include surveying land, monitoring crops, and inspecting infrastructure.

Hyperspectral remote sensing involves the measurement of the reflection of electromagnetic radiation across a wide range of wavelengths, providing detailed information about the chemical and physical properties of materials on the Earth's surface. Hyperspectral sensors can identify the unique "signature" of different materials, making them useful for a variety of applications, including mineral exploration, environmental monitoring, and military surveillance.

In this thesis, the focus will be directed towards exploring the mineral content of subvertical structures such as quarry walls and open pits, utilizing a multi-sensor approach to capture the targets across different regions within the electromagnetic spectrum.

2.1.1 The electromagnetic spectrum

Light is usually interpreted as the *visible light*; that's because it is what can be perceived by the eye, but that changed in the 1800s when it was discovered that light was a more general phenomenon; and it is more common to use **electromagnetic radiation** when referring to light in its various forms [10].

The electromagnetic spectrum is the **range** of electromagnetic radiations.

The figure 2.1 shows important properties and relations between different radiations of the electromagnetic spectrum. The order of these radiations in increasing wavelength is: Gamma-rays γ, X-rays, Ultra-Violet, Visible, Infrared, Micro-waves, Radio-waves.

The infrared portion of the electromagnetic spectrum is usually divided into three sub-regions; the *near*-, *mid*- and *far*-infrared, named for their relation to the visible spectrum.

The historical debate on the **particle** versus **wave** interpretation of light is well known. «Put simply, light behaves as a wave when it propagates and like a particle when it is detected» [11].
In the **wave** theory, electromagnetic radiation has :

- *frequency* ν: the number of cycles of light that pass a given point in one second
- *amplitude* a: the magnitude of the wave's displacement (½ the height between the peaks and troughs); it is related to the intensity of the wave (brightness for light, loudness for sound)
- *wavelength* λ: the distance between corresponding points in 2 adjacent light cycles
- *speed* V: $V = \lambda \times \nu$
- *energy* E: $E = h \times \nu$ where Planck's constant $h = 6.626 \times 10^{-34} J.s$

The other theory, offers insights into how light interacts with matter. In this -**particle**- theory, the electromagnetic radiation is composed of discrete units called **photons** or **quanta** [3].

The energy of a quantum is :

$$Q = h\nu \tag{2.1}$$

where:

- Q: energy of a quantum, joules (J)
- h: Planck's constant $h = 6.626 \times 10^{-34} J.s$
- ν: frequency

All matter at temperatures above absolute zero ($0K$, or $-273°C$) continuously emits electromag-

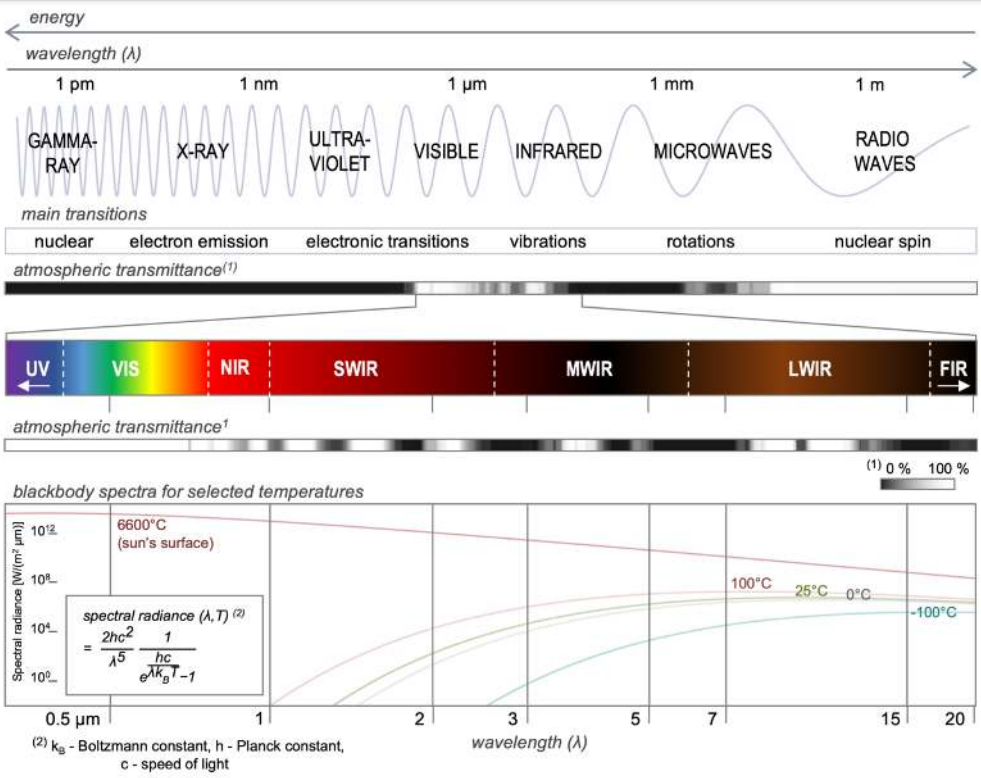


Figure 2.1: The electromagnetic spectrum [5]

netic radiation [3].

The energy radiated by objects is, among other things, a function of the surface temperature of the object as expressed by the *Stefan-Boltzmann law* [3]:

$$M = \sigma T^4 \tag{2.2}$$

where:

- M: total radiant exitance from the surface of a material; $watts(W)m^{-2}$
- σ: Stefan–Boltzmann constant, $5.6697 \times 10^{-8} W m^{-2} K^{-4}$
- T: absolute temperature (K) of the emitting material

Actually this law is expressed for an energy source that behaves as a **black body**: a hypothetical, ideal radiator that totally absorbs and re-emits all energy incident upon it. And real objects only approach the black body [3].

The spectral distribution of the emitted energy also varies just as the energy emitted by an object varies with temperature [3]. The lower part of the figure 2.1 shows energy distribution curves for a black body at different temperatures.

* EM energy propagates in waves formed by electric and magnetic fields

Quantization theory of the energy of electrons responsible of the main transitions in specific regions of the EM spectrum. see figure [5]. Every individual atomic species (atoms, ions, molecules) can have a possible quantum state that are well defined at a characteristic energy level [5].

Electronic, vibrational, rotational and translational processes and electron spins are associated with sets of energy levels that an atomic species can have [5]. Every set have one low energy or ground state, and multiple high energy or excited states reached when absorbing the equivalent of the state's energetic difference; and then momentarily transition back to a lower energy state by an emission of of the same energy for that transition [5].

The figure 2.2 illustrates this notion using an energy diagram.

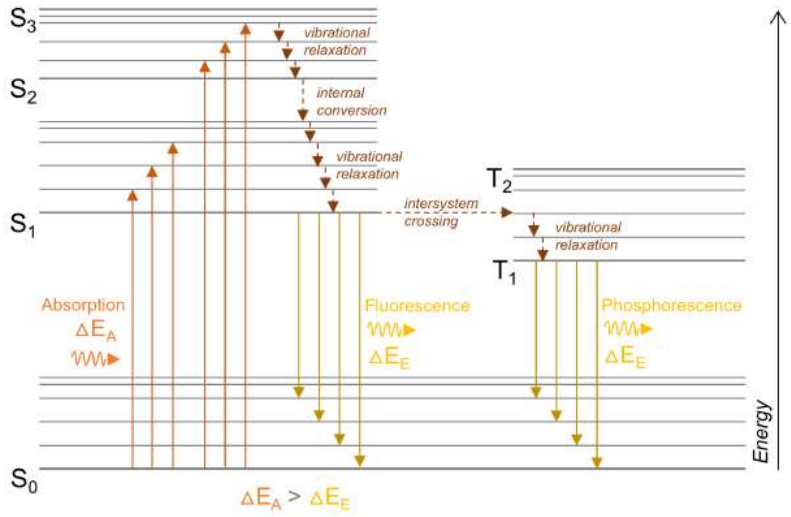


Figure 2.2: Energy diagram of transitions between electronic states of a molecule during absorption, fluorescence and phosphorescence S - single state, T - triplet state [5]

.

2.1.2 Interaction with earth surface features

When energy from light, radio waves, or other forms of electromagnetic radiation hits a feature on the earth's surface, it can do one of three things: reflect off the feature, be absorbed by it, or pass through it [3]. This is shown in the figure 2.3 for a small piece of a body of water. Applying the principle of conservation of energy, we can state the interrelationship among these three energy interactions as [3]:

$$E_I(\lambda) = E_R(\lambda) + E_A(\lambda) + E_T(\lambda) \tag{2.3}$$

where:
E_I : incident energy
E_R : reflected energy
E_A : absorbed energy
E_T : transmitted energy
with all energy components being a function of wavelength λ.

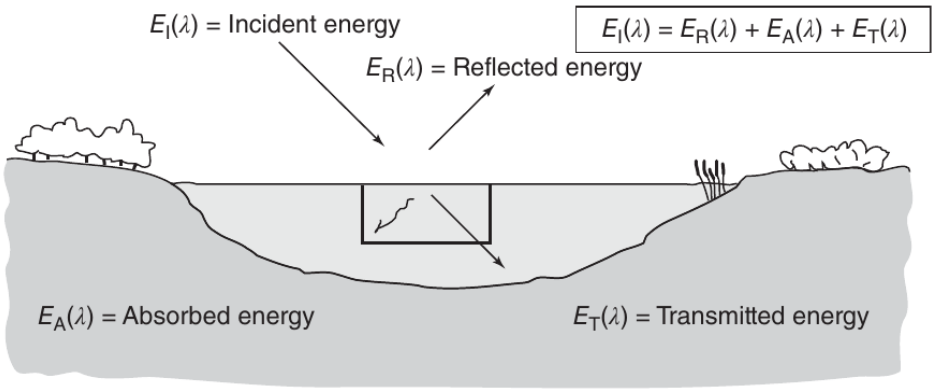


Figure 2.3: Interactions between electromagnetic energy and earth surface features [3]

This last equation is an energy balance equation that demonstrates the relationship between reflection, absorption, and transmission. Two points to consider about this relationship are: first,

the amount of energy reflected, absorbed, and transmitted will vary for different Earth features, depending on their material type and condition, these variations allow us to distinguish different features in an image [3]. Second, the wavelength dependency means that, even within a given feature type, the proportion of reflected, absorbed, and transmitted energy will vary at different wavelengths, and therefore, two features may be indistinguishable in one wavelength range but be very different in another wavelength band [3].

2.1.3 Hyperspectral imaging

Hyperspectral imaging is acquiring images in many, very narrow, contiguous spectral bands throughout the visible, near-IR, mid-IR, and thermal-IR portions of the spectrum [3].

The figure 2.4 portrays a hypercube: a pixel constructed from a continuous spectrum.

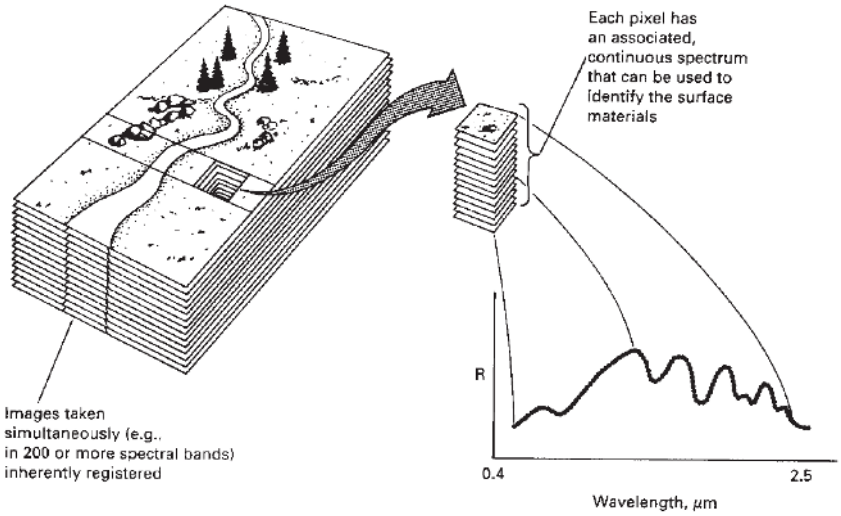


Figure 2.4: Hyperspectral image (hypercube), each pixel representing a continuous spectrum [3].

Passive hyperspectral reflectance measurements in visible and infrared portions of the electromagnetic spectrum, are rapid, non-destructive and safe [5].
It can provide information about the distribution of rock-forming and alteration minerals, specific compounds and ions [5].
More differentiation of small compositional changes of substances in comparison to true color or multi-spectral [5].
Hyperspectral remote sensing allows for a variety of scientific and industrial fields to obtain spatially continuous compositional information of samples, outcrops, or regions that are inaccessible, large, dangerous or environmentally valuable [5].

2.2 Data Acquisition

2.2.1 Principles

Remotely sensed data can be classified in a variety of ways depending on: the wavelength of electromagnetic radiation that is being used to collect the data, the target that is being observed, the scale at which the data is being collected, the type of the sensor used, and the spectral resolution (number of bands) [3].

In terms of wavelength, remote sensing data can be classified as visible, infrared, or microwave, depending on the portion of the electromagnetic spectrum that is being used to collect the data [3]. This will be addressed in the next subsection.

In terms of the target that is being observed, remote sensing data can be classified as active or passive, depending on whether the sensor is emitting energy or simply detecting energy that is being emitted by the target [3]. Active sensors, such as radar, emit their own energy and then measure the reflection of that energy off the target [3]. Passive sensors, such as cameras, simply detect energy that is being emitted by the target, such as visible light or infrared radiation [3].

The scale classification ranges from satellites observing the Earth at different altitudes (actively and passively) space-borne class, passing by the airborne-class including manned aircraft and unmanned ones e.g. drones to Terrestrial/Small-angle Scans and finally to the lab or near-field [5].

Sensor type classification the properties and capabilities of the sensors. One type of sensor is the point spectrometer, which is used to measure the intensity of light at different wavelengths [12]. Another type is the pushbroom spectrometer, which scans a scene to collect data over a wide area [12]. Spectral 2D imagers are another type of sensor, and these can be further divided into several subtypes [12]. The multi-camera 2D imager uses multiple cameras to capture images [12]. The sequential 2D imager captures images sequentially, and the snapshot 2D imager captures multiple images at the same time. Within the snapshot 2D imager, there are several sub-types such as the multi-point spectrometer, mosaic filter-on-chip camera, spatio-spectral filter-on-chip camera, characterized (modified) RGB camera, and spatio-spectral camera [12]. These different types of sensors each have their own advantages and disadvantages, and are used in a variety of applications. The figure 2.7 presents the data captured by different types of sensors.

Finally, remote sensing data can be classified based on the spectral resolution [3]. Many sensors collect data in multiple bands or wavelengths, and the number of bands can vary depending on the sensor and the application [3]. For example, a sensor with a large number of bands may be able to detect a wide range of wavelengths and provide more detailed information about the features being studied, while a sensor with fewer bands may have a narrower range of wavelengths but may be able to collect data more quickly or at a lower cost [3].

The visible and infrared ranges are certainly among the most widely studied and utilized in many applications, such as imaging and remote sensing [3]. The human eye is able to perceive light in the visible range, which makes it easy for us to interpret and understand images captured in this range [3]. Additionally, the atmosphere is relatively transparent in certain infrared bands, making it possible to gather useful data from satellites and other remote sensing platforms [3]. We will dive deeper into this 2 special ranges next.

Visible Visible wavelengths are those that are visible to the human eye and are typically used to observe features on the Earth's surface such as vegetation, bodies of water, and man-made structures.

They are typically defined as those between 400 and 700 nanometers (nm). Within this range, different colors correspond to different wavelengths: red light has a wavelength of around 700 nm, green light has a wavelength of around 550 nm, and blue light has a wavelength of around 450 nm. In remote sensing, images are often captured using sensors that are sensitive to these specific wavelengths, resulting in red, green, and blue channels of information. These channels can then be combined to create a full-color image of the Earth's surface.

Infra-Red Infrared wavelengths are longer than visible wavelengths and are used to observe features on the Earth's surface that are not visible in the visible spectrum, such as changes in temperature and humidity.

Infrared spectroscopy (IR spectroscopy or vibrational spectroscopy) is the measurement of the interaction of infrared radiation with matter by absorption, emission, or reflection.
Most materials absorb electromagnetic radiation in the IR spectral region at wavelengths (from 0.8 to 14 μm) that are characteristic of the material's molecular structure [13]. Some of the IR spectrometers are: grating-based/dispersive IR spectrometers, FTIR spectrometers, and filter-based or non-dispersive IR (NDIR) instruments [13]. We will dig deeper into FTIRs.

FTIR A **F**ourier **T**ransform **I**nfra**R**ed spectrometer also called *interforometer* is an instrument which acquires broadband spectra from NIR to LWIR [14]. The interferometer produces a unique type of signal which has all of the infrared frequencies "encoded" into it [15]. Unlike other multi/hyper-spectral instruments (grating monochromator or spectograph), FTIR spectrometers collect all wavelengths simultaneously, this is known as the **Multiplex** or **Felgett Advantage** [14].

The figure 2.5 represents the layout of the imaging FTIR spectrometer.

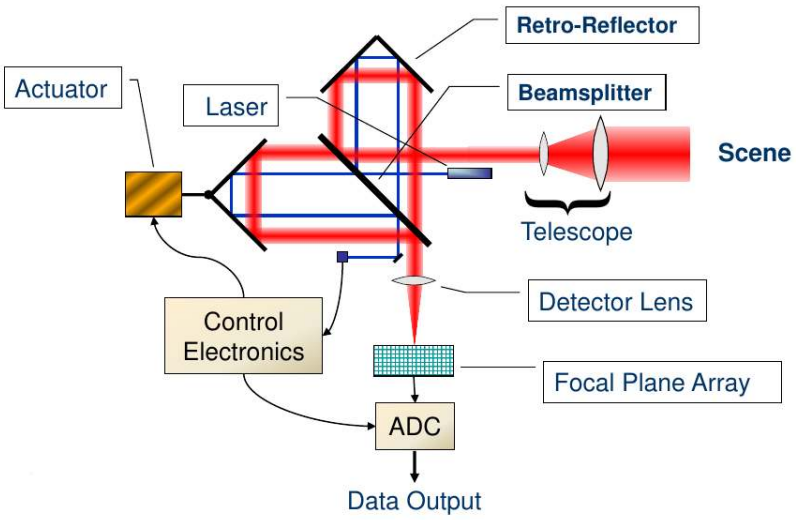


Figure 2.5: Block diagram of imaging Fourier Transform Spectroradiometer [16]

The interferometer employ a **beamsplitter** that have as input the infrared beam and divides it into 2 optical beams: one beam reflects off of a flat fixed mirror, the other beam reflects off of a flat slightly dynamic mirror (few mm) [15]. The 2 beams reflect off of their respective mirrors and are **recombined** when they meet back at the beamsplitter.
The signal exiting the interferometer called **interferogram** is the result of these 2 beams interfering with each other (because the path that one beam travels is a fixed length, the other constantly changing due to its mirror movements).
The interferogram has the unique property that every data point (a function of the moving mirror position) of the signal has information about every infrared frequency of the source.
As the interferogram is measured, all the frequencies are being measured simultaneously.
For analysis a frequency spectrum (plot of the intensity at each individual frequency) is needed. So the interferogram need to be "decoded" for interpretation. This decoding can be done via **Fourier transformation**.

Some of the major advantages of FT-IR over the dispersive technique include [15]:

- Speed: Because all of the frequencies are measured simultaneously, most measurements by FT-IR are made in a matter of seconds rather than several minutes. This is sometimes referred to as the Felgett Advantage.
- Sensitivity: Sensitivity is dramatically improved with FT-IR for many reasons. The detectors employed are much more sensitive, the optical throughput is much higher (referred to as the Jacquinot Advantage) which results in much lower noise levels, and the fast scans enable the co-addition of several scans in order to reduce the random measurement noise to any desired level (referred to as signal averaging).
- Mechanical Simplicity: The moving mirror in the interferometer is the only continuously moving part in the instrument. Thus, there is very little possibility of mechanical breakdown.
- Internally Calibrated: These instruments employ a $HeNe$ laser as an internal wavelength calibration standard (referred to as the Connes Advantage). These instruments are self-calibrating and never need to be calibrated by the user.

Optical Path Difference (**OPD**) is the difference in distance traveled by the two beams in an interferometer [14]. It is calculated by multiplying the distance traveled by the moving mirror (multiplied by 2, 4, or other multiplier, which is a function of the number of reflecting elements used), and the index of refraction of the medium in the interferometer.

FT-IR has a natural reference point when the moving and fixed mirrors are at the same distance from the beam splitter, it is called the zero path difference or ZPD [14].

LiDaR LiDaR stands for Light Detection and Ranging is a technology that provides highly precise registration of spatially distributed data [17]. The LIDAR system consists of a laser device, an inertial navigational measurement unit, a high-precision airborne global positioning system, and a computer interface [17]. In contrast to most optical remote sensing methods, the acquisition of LIDAR data is not significantly influenced by solar illumination and can be performed during nighttime [17]. Among the various return attributes, intensity is the most frequently requested and utilized [17].

The sensors mentioned above can be installed on various platforms, including satellites, airplanes, drones, terrestrial vehicles, or ground-based setups, to capture data from a target [18]. Airborne hyperspectral imaging sensors are employed for large-scale remote sensing applications, while satellite-based sensors are utilized for even larger scales, such as monitoring vegetation health, detecting land cover changes, and mapping geology [18]. On the other hand, drones equipped with hyperspectral sensors are deployed for smaller-scale applications such as precision agriculture, environmental monitoring, and infrastructure inspection [18]. In addition, terrestrial hyperspectral sensors are utilized in laboratory settings for analyzing samples or conducting field spectroscopy of vegetation and soil [18].

Before delving into how data is processed, it is worth talking about how it is saved and stored.

2.2.2 Data Products

Digital image

Before electronic sensors were invented, analog cameras used chemicals on light-sensitive film to detect changes in energy in a scene [3]. By developing the film, we would get a record of the energy signals [3]. The film acted as both a detector and a recorder [3].

These pre-digital cameras had many advantages: they were simple, affordable and provided a lot of detail and accuracy in the image [3]. Nowadays, electronic sensors make an electrical signal that matches the changes in energy in the original scene [3]. A common example of an electronic sensor is a digital camera [3]. Different types of electronic sensors have different designs and detectors, such as charge-coupled devices (CCD) or antennas for detecting microwaves signals [3].

No matter which type of detector is used, the information gathered is usually saved onto a computer storage device, such as a hard drive, memory card, solid-state storage device, or optical disk [3]. Even though electronic sensors can be more complex and costly than film-based systems, they offer benefits such as a wider range of sensitivity to different colors of light, better possibilities for calibrating and measuring with the sensor, and the possibility to save and send the information electronically [3].

The basic character of digital image data is shown in figure 2.6 [3]. Although the image in (a) looks like a normal photograph, it is actually made up of a grid of small, individual parts called pixels. The brightness of each pixel corresponds to the average brightness measured electronically from the area on the ground that the pixel covers. The figure shows 500 rows and 400 columns of pixels. In (a), it is difficult to see the individual pixels, but they are more clear in the larger views in (b) and (c). These are close-ups of smaller areas in the center of the image in (a). (b) shows a close-up of 100 rows and 80 columns, and (c) shows a close-up of 10 rows and 8 columns. Part (d) shows the numerical value assigned to each pixel, called the digital number (DN) which corresponds to the average brightness measured by the pixel in (c). These values are obtained

by converting the original electrical signal from the sensor into positive whole numbers using a process called analog-to-digital conversion [3].

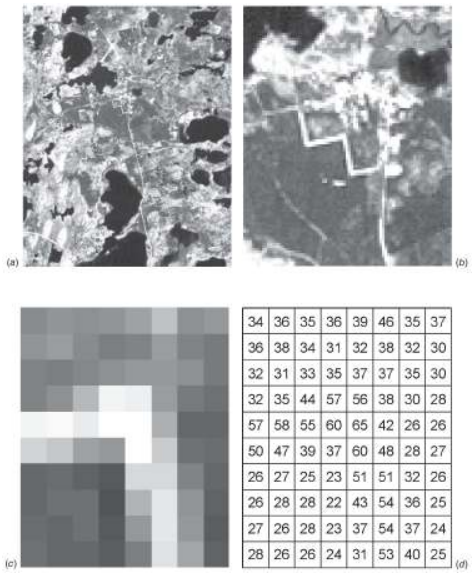


Figure 2.6: Basic character of digital image data [3]

In the case of a digital multispectral image, each pixel includes multiple DNs, one for each spectral band [3].

Hypercube

A data cube or hypercube is a three-dimensional structure made up of multiple digital images of the same scene, taken at different wavelengths that are arranged in a stacked format [3]. The dimensions of the hypercube are defined as one spectral dimension (λ) and two spatial dimensions (X and Y). [19].

The figure 2.4 portrays a hypercube.

The figure 2.7 summarizes the different data cube slices that are obtained from multiple sensor types [12].

Point cloud

Point clouds are a fundamental data representation in computer vision and remote sensing, and they can be obtained using several techniques [20]. These techniques include image-derived methods, Light Detection And Ranging (LiDAR) systems, Red Green Blue -Depth (RGB-D) cameras, and Synthetic Aperture Radar (SAR) systems. Each of these methods has unique survey principles and platforms that result in diverse data features and application ranges [20].

Hypercloud

A **hypercloud** is a 3D representation of hyperspectral data that is geometrically accurate [7, 21]. It is created by combining high-resolution point clouds with high-resolution spectral imagery or datacubes.

The figure 2.8 is a false colour visualisation of a hypercloud [6].

A hypercloud make it easier for hyperspectral images to be integrated into a single product for visualisation, interpretation and analysis [6]. The 3-D nature of this product mitigates occlusion and distortion issues that limit approaches relying on orthographic projection, and facilitates crucial corrections of illumination and atmospheric effects in the hyperspectral data [6].

Upon acquisition of the data, processing is required for it to be usable and free from artifacts.

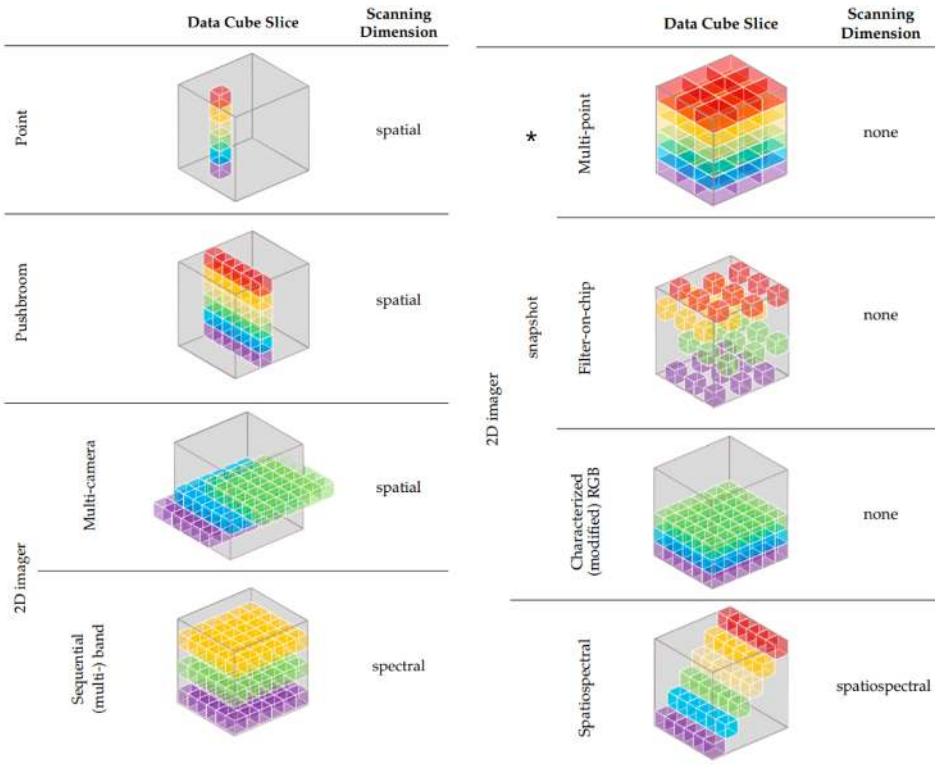


Figure 2.7: Data cube slices that are obtained from multiple sensor types [12]

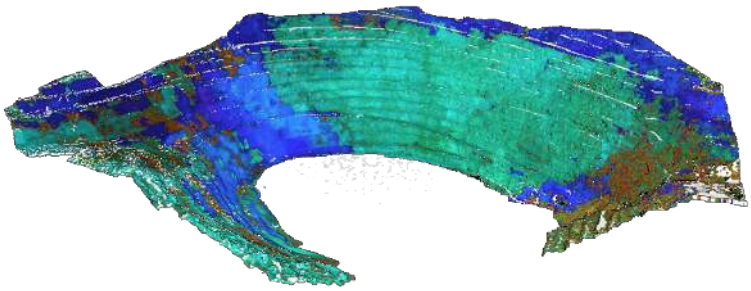


Figure 2.8: Hypercloud of an open pit in Corta Atalaya [6]

2.3 Data Processing

Image processing methods can be categorized into three functional categories [22], which are as follows, along with a list of typical processing routines [22]:

1. **Image restoration** compensates for image errors,noise, and geometric distortions introduced during the scanning, recording, and playback operations. The objective is to make the restored image resemble the scene on the terrain. Typical processing routines include:
   (a) Restoring line dropouts
   (b) Restoring periodic line striping
   (c) Restoring line offsets
   (d) Filtering random noise
   (e) Correcting for atmospheric scattering
   (f) Correcting geometric distortions
2. **Image enhancement** involves changing the appearance of an image to enhance the informa-

tion it conveys. The goal is to make the image clearer and more informative for the viewer. Common image enhancement techniques include:
   (a) Contrast enhancement
   (b) Density slicing
   (c) Edge enhancement
   (d) Making digital mosaics
   (e) Intensity, hue, and saturation transformations
   (f) Merging data sets
   (g) Synthetic stereo images
3. **Information extraction** is the process of using a computer to combine and analyze different aspects of a data set. The goal is to show the hidden spectral and other features of the scene that are not clearly visible in restored and enhanced images. Common processing routines include:
   (a) Principal-component images
   (b) Ratio images
   (c) Multispectral classification
   (d) Change-detection images The images in this report have been processed with various combinations of these routines.

In the following, we will examine the most relevant routines for our current study.

2.3.1 Radiometric corrections

Radiometric effects refer to any factors that affect the spectroscopic information within a data set [5]. These effects can be global, local to a specific location, or local to a specific wavelength [5]. They can be caused by internal factors, such as technical problems with the sensor, or external factors, such as the environment. Radiometric correction involves the alignment of the data to a reference system and can include correction for at-sensor radiance, top of atmosphere (TOA) reflectance, or surface reflectance [5].

Internal corrections refer to adjustments made to the data collected by the sensor itself [5]. This includes corrections for dark current, which is caused by thermal noise, and bad pixels, which are pixels that are not working properly. [5] Vignetting and smile corrections are also applied to account for the uneven response of the sensor across the image [5]. Keystone correction is used to correct any distortion caused by the angle of view of the sensor [5].

External radiometric effects can include changes in illumination, atmospheric conditions, and the surface being observed [5]. These effects can be corrected through a variety of approaches, including the use of calibration values, atmospheric correction algorithms, and surface reflectance models [5].

The reflected signal from a given surface depends on a number of parameters and its behavior can be described by the **Bidirectional Reflectance Distribution Function**. [23] as shown in equation 2.4:

$$f_r\left(\lambda,\theta_i,\phi_i,\theta_r,\phi_r\right)=\frac{dL_r\left(\lambda,\theta_r,\phi_r\right)}{dE_i\left(\lambda,\theta_i,\phi_i\right)}=\frac{dL_r\left(\lambda,\theta_r,\phi_r\right)}{L_i\left(\lambda,\theta_i,\phi_i\right)\cos\theta_i d\omega_i} \tag{2.4}$$

The Bidirectional Reflectance Distribution Function (BRDF) is a measure of the way that light is reflected off a surface, it describes the relationship between the amount of light that is incident on a surface and the amount of light that is reflected off of it [23]. The BRDF is a function of various parameters, including the wavelength of the light, the direction of the incident and reflected light, and the material properties of the surface [23].

The BRDF can be divided into two main components: specular reflection and diffuse reflection [23]. Specular reflection is concentrated in a small area and is characterized by a single bright spot on the surface. It is caused by light that is reflected off the surface at a single angle. Diffuse reflection, on the other hand, is spread over a wider area and is characterized by a more uniform

brightness across the surface. It is caused by light that is reflected off the surface at many different angles.

The BRDF of a surface is also influenced by the angle at which light hits the surface, as well as the surface roughness and material properties. Surfaces that reflect light in a more isotropic manner, with a constant BRDF regardless of the direction of the incident light, are referred to as Lambertian surfaces [23]. Non-Lambertian surfaces, on the other hand, have a BRDF that depends on the direction of the incident light.

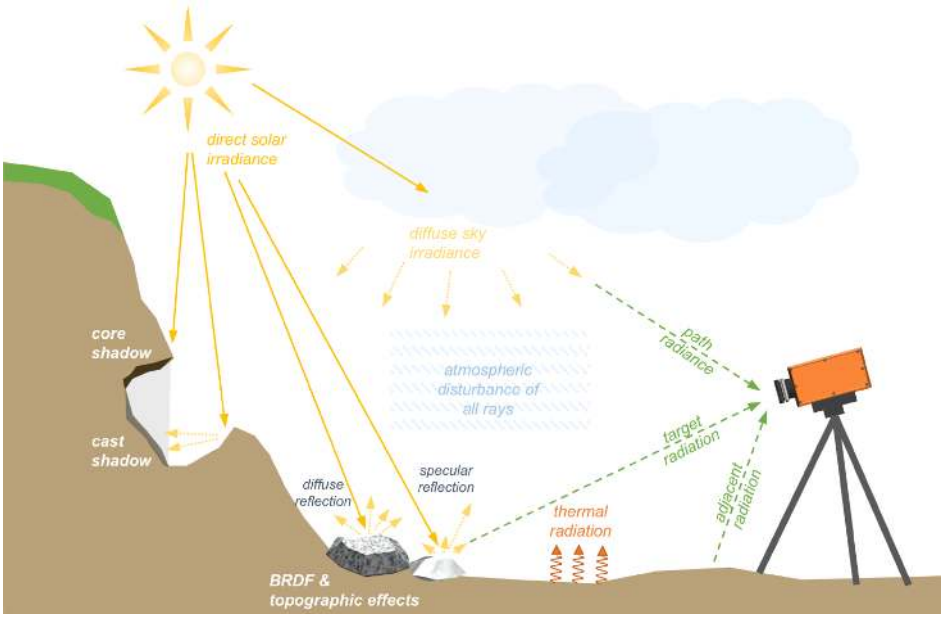


Figure 2.9: External radiometric disturbances [5]

2.3.2 Geometric corrections

Geometric disturbances are any factors that impact the spatial accuracy of an image or dataset. Spatial accuracy is achieved when the spatial projection of the image or dataset matches its actual location within a reference surface or space [5]. Geometric distortions can be corrected through a process called orthorectification, which involves compensating for geometric distortions and aligning the dataset with a reference system [5]. There are several types of geometric disturbances that can affect remotely sensed data, including technical imperfections of the sensor, viewing angle of the sensor, movement of the sensor or platform, and topographic features [5]. These distortions can be corrected through a variety of approaches, including the use of distortion coefficients, logging sensor movement and position, warping the dataset to an orthophoto with a similar or higher spatial resolution, and projecting the image onto a high-resolution digital elevation model [5].

We will dissect in more details the camera calibration, the most important geometric correction.

Camera calibration Camera calibration is an essential step in setting up a measurement system, it is necessary to calibrate the camera before any use to ensure accurate measurements [24].It helps to ensure that the features of the image conform to the laws of projective geometry, as it allows to determine the precise direction of the projection ray for each pixel, which is crucial for accurate measurements and analysis.

When calibrating a camera, there are various criteria that can be used to classify the calibration process. One way to classify camera calibration is based on whether it is linear or non-linear [25]. Linear calibration methods assume that the relationship between the camera parameters and

the image measurements is linear, while non-linear methods take into account more complex relationships [25]. Another way to classify camera calibration is based on whether it is intrinsic or extrinsic [25]. Intrinsic calibration deals with the internal parameters of the camera such as focal length and image center, while extrinsic calibration deals with the position and orientation of the camera in the world coordinate system [25]. Additionally, camera calibration can be classified as implicit or explicit [25]. Implicit calibration methods use the image measurements directly, while explicit calibration methods use additional information such as 3D points or geometrical properties [25]. Finally, camera calibration can be classified based on whether it uses 3D points or geometrical properties [25]. The use of 3D points allows the reconstruction of the 3D scene, while the use of geometrical properties only allows the determination of the camera's internal parameters [25].

There are different types of algorithms that can be used for camera calibration, each with their own advantages and limitations. Traditional visual calibration algorithms include the DLT (Direct Linear Transformation) algorithm, the Tsai (2-steps method), and the Zhang+ [26]. These methods are based on the principles of cross-ratio invariance, and are often used in conjunction with nonlinear optimization techniques [27]. Additionally, the Biplane algorithm is also widely used [27]. Camera self-calibration algorithms include those based on the absolute conic and the absolute quadric [26]. These methods are useful for cameras that are not able to perform a traditional calibration. Active-vision-based calibration algorithms include those based on three orthogonal translational motions and those based on camera pure rotation [26]. These methods are useful for cameras that have the ability to move in order to optimize the calibration process.

The table 2.1 compares the different calibration methods [26].

Table 2.1: Comparison of different classes of calibration methods [26]

Calibration algorithms	Advantages	Disadvantages
Traditional visual calibration	High precision	Complex process Requires calibration blocks
Camera self-calibration	No need for calibration blocks Good flexibility	Low precision
Active-vision-based calibration	Simple calculations	High system cost

2.3.3 Data Fusion & Co-registration

Data fusion is the process of combining multiple data sources to create a more complete and accurate dataset [28]. It involves the integration of data from multiple sensors or platforms to create a single, cohesive dataset [28]. This can be done for a variety of purposes, such as to improve the spatial, spectral, or temporal resolution of the data, or to provide complementary information about different aspects of the Earth's surface or atmosphere [28]. Co-registration refers to the process of aligning multiple data sources so that they can be accurately compared or combined [28]. This can be done through various techniques such as image registration, sensor registration, and feature-based registration [28]. In summary, data fusion is the process of integrating multiple data sources to create a more informative dataset, while co-registration is the process of aligning multiple data sources so that they can be accurately compared or combined.

There are several approaches to data fusion in remote sensing, including **pixel-level** fusion, **feature-level** fusion, and **decision-level** fusion [29]. Pixel-level fusion involves the combination of data at the individual pixel level [29]. While feature-level fusion involves the integration of features extracted from the data [29]. Decision-level fusion involves the integration of the results of different data processing techniques or classifications [29].

Data fusion can be used to address a number of challenges in remote sensing, such as the trade-off between spatial and spectral resolution, the limited availability of certain types of data, and the need to integrate data from a variety of sources.

Image transformations

The matching transformation can be rigid, affine, projective, or curved; these categories, indicating the degree of elasticity of the transformation, have been selected such that they show a clear distinction in geometrical properties [30].

A transformation is considered rigid if it preserves the distance between any two points when they are mapped from the first image to the second [30]. Conversely, an affine transformation maps any straight line in the first image onto a straight line in the second, while preserving parallelism, and it can be decomposed into a linear transformation and a translation [30]. In contrast, projective (or perspective) transformations (also called homographies) do not preserve parallelism between straight lines, as straight lines in the first image can be mapped onto non-parallel lines in the second image [30]. Curved transformations, on the other hand, may map straight lines onto curves. One well-known class includes polynomial transformations [30]. Figure 2.10 illustrates the aforementioned image transformations..

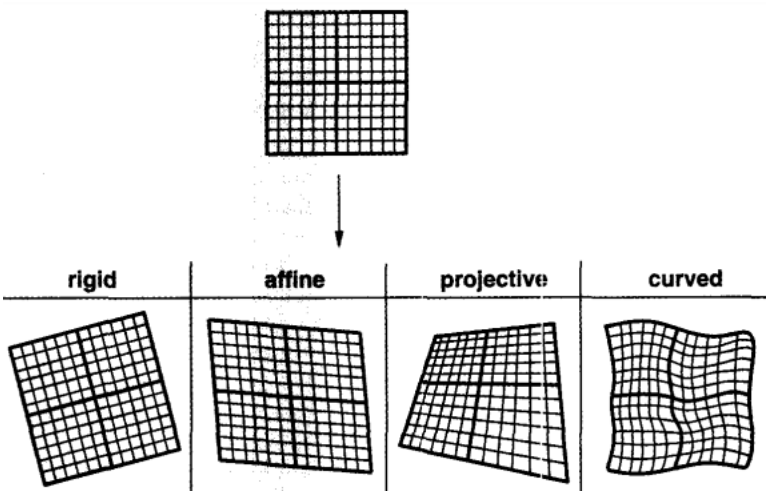


Figure 2.10: Image transformations [30]

The most important that we will use subsequently are the affine and perspective. Figure 2.11 shows an example and the corresponding matrix formula in homogeneous coordinates[1].

Both transformations can be used to align or register images, but homography is more suitable for images that have a significant perspective change, while affine transformation is better suited for images with little or no perspective change. Additionally, while homography requires at least 4 non-collinear points to estimate, affine Transformation can be estimated using only 3 points.

It is imperative to have correspondences between points in both images in order to perform these transformations. To automate this process, feature detectors have been developed.

Feature detectors

The field of feature detection in computer vision involves identifying and extracting distinctive, repeatable patterns in images. There are several popular algorithms for feature detection, including Scale-Invariant Feature Transform (SIFT) [31], Speeded Up Robust Features (SURF) [32], Features from Accelerated Segment Test (FAST) [33], Scale-Invariant Feature Transform-STAR (STAR) a derived version of CenSurE [34], and Oriented FAST and Rotated BRIEF (ORB) [35].

SIFT uses a Difference of Gaussians approach to recognize blobs as local features and has the advantage of being invariant to both scale and rotation [31]. SURF is composed of two steps,

[1] Homogeneous coordinates are a mathematical representation of points and vectors in a space of one dimension higher than the original space. This extra dimension allows for more complex transformations, such as projective transformations, to be performed in a more convenient way. The results of these transformations can then be transformed

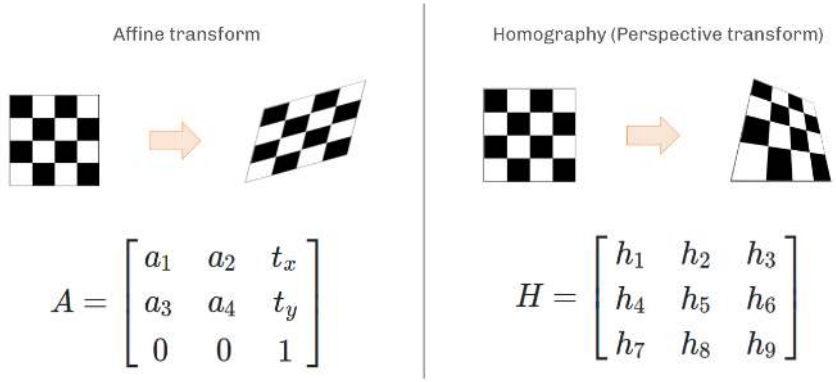


Figure 2.11: Affine and Perspective Transforms and corresponding matrix representation

feature extraction and feature description, and utilizes Hessian matrix-based interest points for efficient computation [32]. FAST considers only 16 pixels around a processing pixel, resulting in faster computation and real-time feature detection [33]. STAR is a scale and rotation invariant detector developed by the OpenCV library that uses Laplacian of Gaussians and two 45-degree difference overlapped squares, it is a derived version of CenSurE [34]. ORB is a combination of FAST and Binary Robust Independent Elementary Features (BRIEF) that results in both efficient feature detection and accurate feature description [35].

Structure from Motion

Structure-from-Motion (SfM) is a method for determining 3-D structure from a series of overlapping images [36]. Unlike conventional photogrammetry, SfM solves for camera positions, orientations, and scene geometry automatically, without requiring prior knowledge of the 3-D positions of targets. The approach involves a highly redundant and iterative bundle adjustment procedure based on a database of features extracted from the images [36]. This method is best suited for sets of images with high overlap that capture full three-dimensional structure of the scene from multiple positions, or images taken by a moving sensor.

The initial step in SfM is the identification of keypoints in individual images using systems such as the Scale Invariant Feature Transform (SIFT) [37]. These keypoints are used for image correspondence and the number of keypoints in an image depends on factors such as image resolution, texture, and complexity [37].

Once keypoints are identified, the process of generating a dense point cloud from images involves several algorithms and mathematical methods, including the sparse bundle adjustment, approximate nearest neighbour algorithm, Random Sample Consensus (RANSAC), similarity transformation, and triangulation and the result of this process is a sparse point cloud that can be further processed to obtain a more dense point cloud using algorithms such as Clustering View for Multi-view Stereo (CMVS) and Patch-based Multi-view Stereo (PMVS2) [37].

The origin of this technique lies in the field of computer vision and was developed in the 1990s (eg. [38] and [39]).

Figure 2.12 represents a typical SfM workflow.

2.4 Mineral exploration

As mentioned earlier remote sensing plays a crucial role in the search for mineral deposits and the reduction of costs associated with their exploration and development [9]. The exploration process

back into Cartesian coordinates, which can be interpreted in the original space.

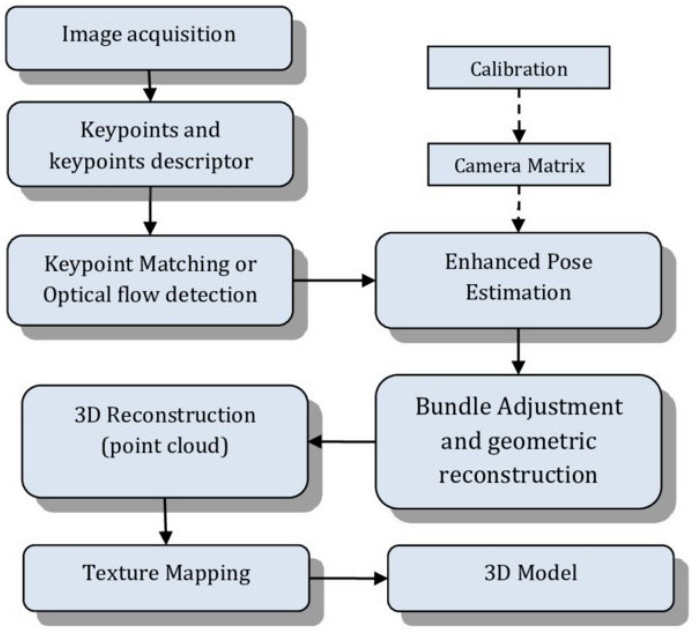


Figure 2.12: Typical SfM workflow [40]

generally consists of four stages, namely prospecting, regional exploration, detailed exploration, and mine exploration, as described by **(author?)** [9].

In the prospecting stage, the goal is to identify potential mineral deposits, or "targets." The most useful information for this comes from small-scale satellite images, larger-scale multispectral images, and airborne surveys [9]. In the regional exploration stage, surface mapping is done at a larger scale, and a few selected sites are studied using geophysical and geochemical techniques [9]. In the detailed exploration stage, investigations are done at an even larger scale, and high resolution remote sensing data can be useful [9]. The final stage, mine exploration, involves finding the minerals at depth and leading to mining and development. High spatial and spectral resolution remote sensing data is obtained [9].

However, it's important to keep in mind that remote sensing has a limited ability to penetrate the Earth's surface [9]. Most mineral deposits are not visible on the surface, and remote sensing can only see a few micrometers to a few centimeters into the Earth's surface [9]. So, the information gathered from remote sensing is often indirect, relying on geological setting, structure, and other factors, rather than directly finding the mineral deposit [9].

Geological mapping is a crucial step in mineral exploration, as it helps to identify geological features related to target mineralization [41]. Traditional methods of geological mapping involve expert knowledge and fieldwork, which can be time-consuming, subjective, and affected by environmental factors [42]. Remote sensing data and advanced data analytics, such as machine learning, can provide a more efficient and reliable alternative [41]. Multispectral and hyperspectral remote sensing instruments have been used to map geological features [4], such as lithological units [43], alteration zones [44], structures [45] and and delineating rocks and minerals [46]. Image processing methods are also used to enhance, extract, and detect features in satellite images [41]. The combination of remote sensing data and digital image processing can help exploration geologists to overcome common challenges associated with traditional methods, such as identifying barren regions and minimizing costs [41].

In recent years, the increase in demand for various minerals coupled with a decline in the number of newly discovered mineral deposits has led exploration geologists to search for more effective and innovative techniques to process various data types at every stage of mineral exploration [41]. Hyperspectral imagery is increasingly being used in geological mapping and mineral exploration [21]. Extensive information can be drawn from the remotely obtained spectral signatures, and it

is used to extract rocks composition and the occurrence of minerals [21]. This made the task of mapping geological outcrops, cliffs, and pit walls faster and safer [21]. And more effective than taking limited number of samples to the lab [21].

One special spectral region in the electromagnetic spectrum is the Thermal Infra-Red (TIR), that's because most materials absorb electromagnetic radiation in the IR at wavelengths (from 0.8 to 14 μm) that are characteristic of the material's molecular structure [13]. Multiple research studies have showcased the utility of Thermal Infra-Red (TIR) data in various thematic domains and applications, such as landscape characterization, determining soil and mineral properties [47], estimating energy fluxes [48], evapo-transpiration and soil moisture, monitoring drought, urban heat islands [49], detecting forest fires, coal fires and volcanoes [50]. That holds true also for mineral mapping and exploration.

Conventional field methods for studying large outcrops can be time-consuming, and correlating data from different parts of the outcrop can be challenging. The exploration of vertical cliff sections using conventional methods poses additional difficulties, such as collecting samples on high, vertical, crumbly, and unsafe wall faces. To overcome these challenges, various airborne or spaceborne approaches have been developed for non-contact geospatial data collection and analysis, such as LiDAR and hyperspectral systems [17]. LiDAR allows for the reconstruction of the shape of vertical outcrops as digital 3D models, but the extraction of mineralogy and lithology is still limited to the single spectral band of the LiDAR's laser [17]. In contrast, hyperspectral systems with high spectral resolution can enable quantitative analysis of surface composition [3]. Although imaging spectroscopy from airborne/spaceborne platforms is well-established, it has mainly been developed in the VNIR and SWIR regions, with limitations in selectivity for mineral identification [51]. The use of LWIR improves selectivity in certain situations due to the spectral features associated with fundamental vibrations being stronger and sharper than their overtones [9]. The inherent self-emission associated with LWIR, also known as TIR, enables geological surveys in various weather and illumination conditions as solid targets such as minerals emit and reflect TIR radiation [3].

A study by Boubanga-Tombet et al. illustrated that utilizing field-based techniques that integrate thermal infrared hyperspectral technology can facilitate the effective mapping of mineralogy and lithology in vertical cliff sections with a case study in in the Jura Cement carbonate quarry in Switzerland [52]. The hyperspectral data acquired using the Telops Hyper-Cam were analysed using temperature emissivity separation algorithms in order to distinguish between the self-emission and reflection contributions of various minerals, and GPS data was utilized for the geometric processing [52].

The Fourier Transform Infrared (FTIR) - Long-wave Infrared (LWIR) sensor is a highly advanced and widely employed technology for the acquisition of hyperspectral data in the Long-wave Infrared (LWIR) range. Despite its efficacy, the sensor is relatively heavy and necessitates mounting on an aircraft or a tripod if used from land, excluding its use with conventional drones that would enhance the convenience and ease of hyperspectral imaging (HSI) as for other light-weight sensors of other regions of the electromagnetic spectrum.

When utilizing Fourier Transform Infrared (FTIR) sensors, a significant challenge that arises is the limited field of view that the technology allows. As a consequence, the resulting data captured by the sensor is minuscule in size. In order to obtain an accurate representation of the target being observed, it becomes crucial to fuse and co-register the data captured by the FTIR sensor, especially with data from other sensors. However, this process can be quite challenging.

Chapter 3

METHODOLOGY

Contents

As we will see in the next section, the acquisition of hyperspectral Long-wave Infrared data using a Fourier-Transform Infrared (FTIR) imaging sensor poses a significant challenge for co-registration due to multiple factors. The sensor's low field of view results in the capture of small images, and the complex 3D sub-vertical target geometry of the objects being imaged makes it difficult to maintain spatial overlap between consecutive images.

Obtaining accurate sensor coordinates and orientation is not possible due to the absence of GNSS receivers and an inertial measurement unit (IMU). As a result, determining the position and orientation of the images to project them onto a geo-referenced point cloud becomes a challenging computer vision task.

The static position of the sensor also mean that the captured images do not provide any 3D information, which further complicates co-registration.

Consequently, this prompted the exploration of addressing the issue using computer vision techniques. The presence of an extra camera, whose images can be matched to those obtained from the LWIR sensor, provides a possible solution. By using a dataset of matched points, the boresight and lever arm can be deduced, or the images can be co-aligned directly through the detection and matching of features.

Moreover, the scarcity of visual similarities between different ranges of the electromagnetic spectrum, especially between Long-Wave Infrared and visible ranges, makes working with LWIR data acquired by an FTIR imaging sensor very challenging. These factors altogether require the development of innovative techniques to enable accurate co-registration of LWIR images acquired using an FTIR imaging sensor.

The decision to carry out the integration process in the 3D space rather than in the 2D image space is beneficial in various ways. Firstly, it helps to minimize problems related to occlusion and distortion that may arise due to variations in terrain topography. Secondly, it simplifies the correction of illumination and atmospheric effects that may affect the hyperspectral data. Finally, it enables accurate interpretation of the geometry of geological structures, providing more detailed and accurate results.

At first we aimed to determine the lever-arm and boresight between the LWIR sensor and the integrated camera by using hyperspectral data from the LWIR sensor and low-resolution RGB images from the integrated camera and a point cloud. This initial approach, which used the correspondences dataset, was not successful. A second approach was developed, which involved matching the LWIR hypercube with its corresponding low-resolution RGB image and matching the low-resolution RGB image with a newly acquired high-resolution RGB image. Finally, a stitching approach was employed to combine multiple LWIR images taken from a single position to construct a larger block of LWIR images that can be matched with a single high-resolution RGB image. Figure 3.1 gives an overview.

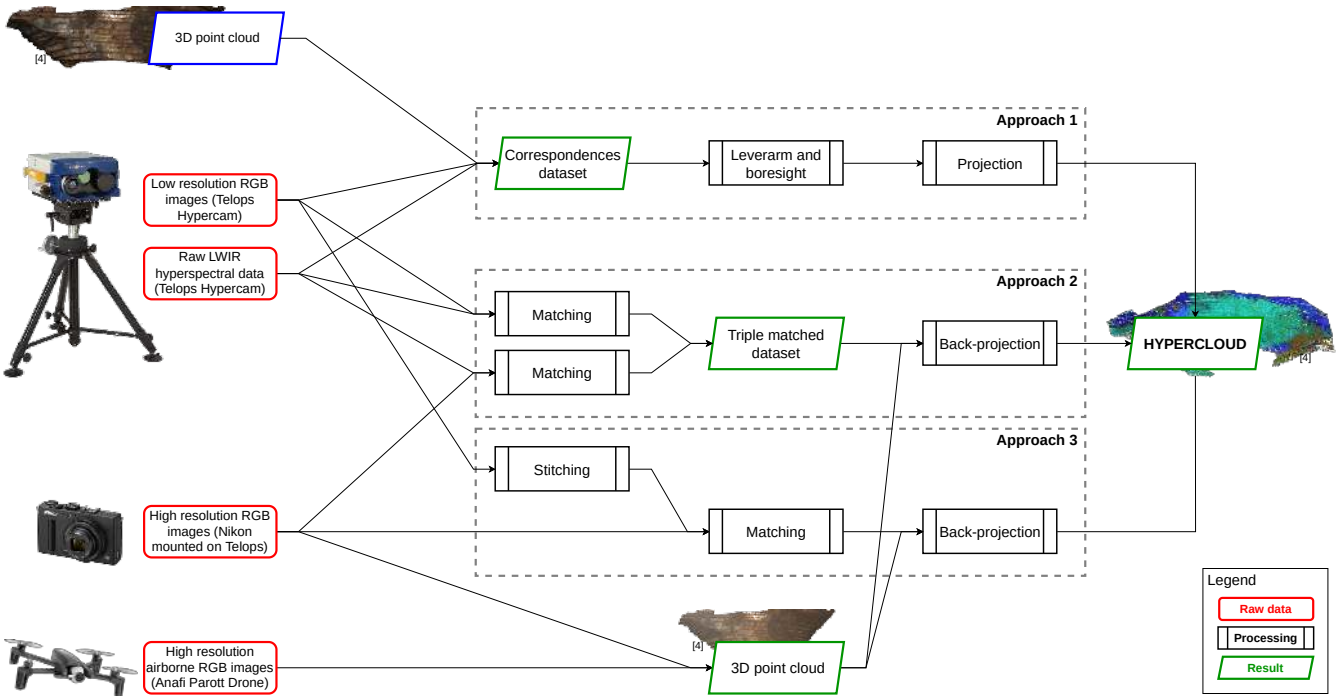


Figure 3.1: An overview of the proposed methodology with the different approaches

Prior to delving into the methodology, an overview of the equipment is provided.

3.1 Equipment

The study was performed with the use of the following equipment:Telops Hyper-Cam, Nikon Coolpix A and Anafi Parrot as sensors, and a 3D printer, station total, GNSS receivers, heat packs and calibration panels.

The detailed description of each equipment is presented below:

3.1.1 Telops Hyper-Cam

The Telops Hyper-Cam -as seen in figure 3.2 - is an advanced remote sensing instrument that uses infrared hyperspectral imaging technology. It has a combination of high spatial, spectral and temporal resolution, which gives it exceptional performance. It is a versatile tool that can be used for remote detection, identification, and quantification of various objects.

The Hyper-Cam offers several key benefits, such as:

- High spatial resolution and imaging quality: It has a 320×256-pixel FPA detector that provides the highest spatial resolution on the market, and also ensures excellent 2D image quality.
- High temporal resolution: The camera can record hyperspectral cubes as a function of time, which allows the characterization of time-dependent events, such as gas cloud dispersion and combustion. The measurement time can be adjusted according to the acquisition parameters, allowing for the fastest recording of dynamic events.
- High spectral resolution: The camera offers the best spectral resolution available, and the spectral features of the targets can be well resolved, providing good selectivity.
- High sensitivity and accuracy: The camera has a high-sensitivity sensor that is combined with automated high-efficiency calibration sources, which ensures excellent radiometric measurements. The device incorporates two internal calibration blackbodies that enable a comprehensive end-to-end radiometric calibration of the measurements.

The Hyper-Cam also comes with various accessories and options, such as: telescopes: the camera is compatible with different telescopes with different fields of view (FOV) options, such as $0.25\times$, $0.5\times$, $3.5\times$, a Global Positioning System (GPS)that has the option to include GPS for location tracking, a motorized polarizer which allows for easy polarimetric measurements, long-range fiber optic data transfer for secure and high-speed data transfer, and filter holder for easy integration of custom filters.

Figure 3.2: Telops Hyper-Cam [16]

Technical specifications

The table 3.1 presents the HyperCam specifications:

Output files

After performing the data processing, the results are organized as shown in listing 3.1.

Table 3.1: Telops Hyper-Cam specifications

Property	Value
Spectral resolution	$0,25cm^{-1}$ to 150
Spatial resolution	320×256 pixels
FPA pixel size	$30\mu m$
Field of view	$6.4 \times 5.1°$
Typical NESR	$< 20(nW/cm^2.srcm^{-1})$
Thermal Scientific Measurement	$1C°$ or 1%
Radiometric accuracy	$< 1K$
Bands	92

```
XXXX_folder
 |-- xxxx.ir.bmp          # preview of the data (broadband infrared image)
 |-- xxxx.jpg             # the rgb image from the side camera
 |-- xxxx.radiance.hdr    # header file
 |-- xxxx.radiance.sc     # the radiance obtained from raw data
 |-- xxxx.raw             # the raw data obtained by the sensor
 |-- xxxx.report.bmp      # the preview with extra info
 |-- Scenario.xml         # Reveal DI software scenario (with extra info)
```

Listing 3.1: File structure of processed output data from Telops Hyper-Cam

Working principle

As already seen in subsection 2.2.1, the Hyper-Cam from Telops is a **FTIR spectrometer**.

3.1.2 Nikon Coolpix A

The **Nikon Coolpix A** as shown in figure 3.3, is a compact digital camera made by Nikon. It features a 16.2-megapixel DX-format CMOS sensor, a 28mm wide-angle lens with a maximum aperture of f/2.8, and a 3-inch LCD monitor. It also has manual exposure controls and a built-in flash. It is capable of shooting 1080p Full HD video at 60fps and has a maximum ISO of 6400. It also has a variety of shooting modes such as Programmed Auto, Shutter Priority, Aperture Priority and Manual Mode. It has a compact and lightweight design making it easy to carry around. Additionally, it also offers wireless connectivity options such as WiFi and Bluetooth.

The Nikon camera produced images with the following specifications: the output was in the **JPEG** file type with a file size of 4.1 MB and dimensions of 4928×3264 pixels.

Figure 3.3: Nikon COOLPIX A

3.1.3 Anafi Parrot

The **Parrot ANAFI** drone, a portable, high-performance drone made by Parrot. It has a 4K HDR camera that can capture 21MP photos and shoot 2.8K videos. It has a 180-degree tilt gimbal and a 3-axis image stabilization. It has a flight time of up to 25 minutes and can fly at a top speed of 33 mph. It also features a 32-minute battery life and can fly up to 2.5 miles away. It has a SmartDronies feature that allows it to perform pre-programmed flight movements and a Follow Me mode that allows it to follow and film the user. It also has a FreeFlight 6 app that provides features such as waypoint navigation, flight planning, and live streaming.

The Anafi Parrot drone have the capability to produce images in the **JPEG** file format with a size of 3.9 MB. The dimensions of the images were 4608 × 3456.

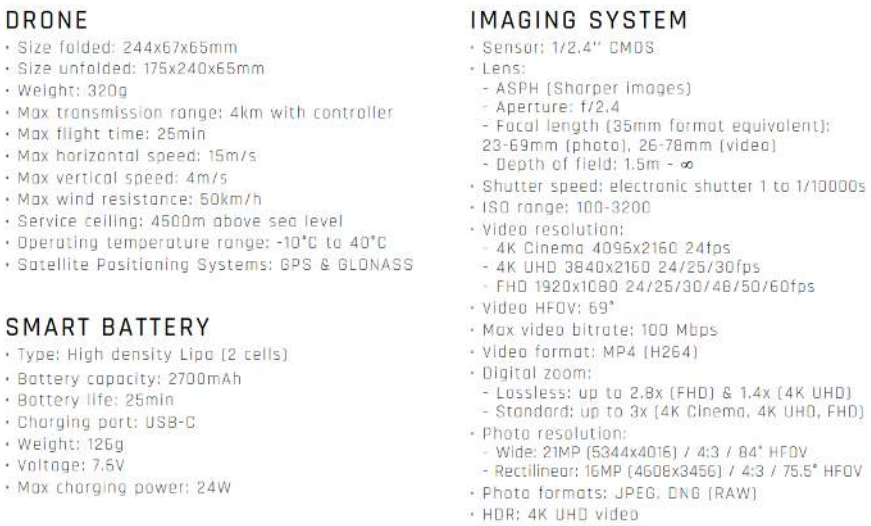


Figure 3.4: Technical specifications of Anafi Parrot

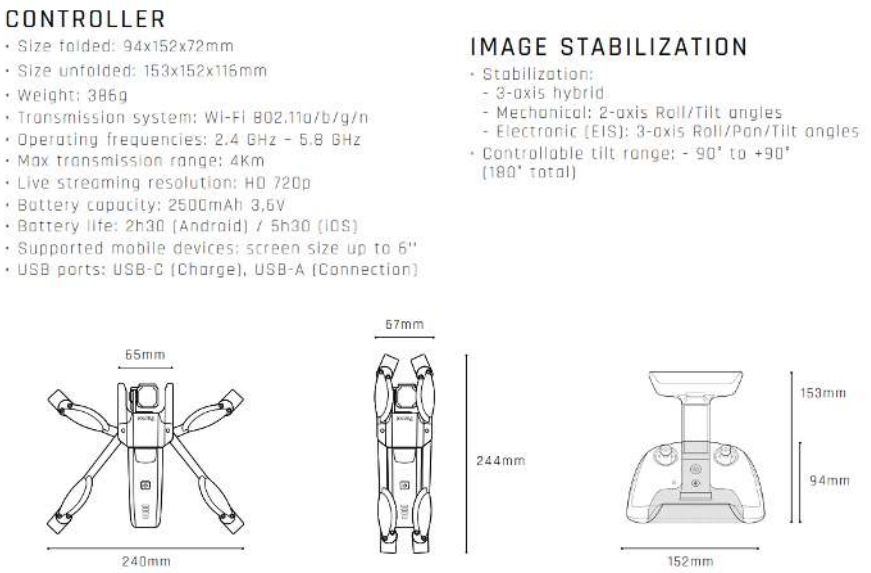


Figure 3.5: Technical specifications of Anafi Parrot - cont'n

3.1.4 3D Printer

The Original **Prusa** Mini is a compact, affordable, and easy-to-use 3D printer made by Prusa3D. It has a build volume of 18 × 18 × 18 cm, a filament run-out sensor, and a filament cooling fan. It is suitable for both beginners and advanced users. It offers a range of features such as a filament sensor, a power panic feature, a multi-material upgrade, and a filament dryer. The printer also compatible with various filaments.

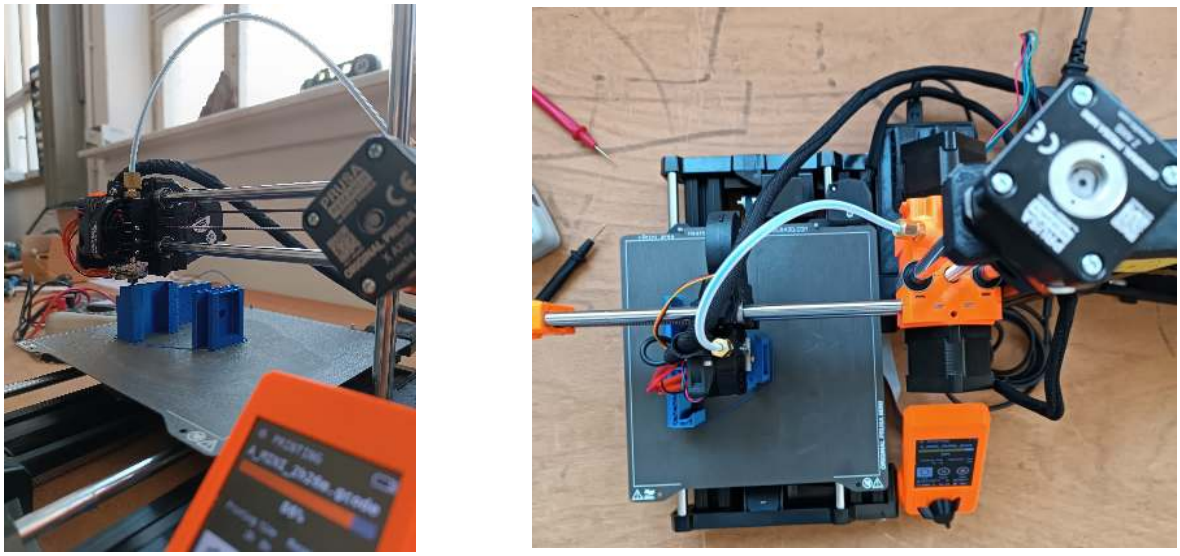


Figure 3.6: PRUSA 3D Printer

3.1.5 Other

In addition to the aforementioned equipment, this study employed the use of a station total and GNSS receivers for georeferencing the products, air-activated heat packs (known as toe warmers) that are "visible" in the LWIR portion of the EM spectrum, calibration panels for radiometric correction purposes, and a field notebook to document important observations and details.

3.2 Methodology

The methodology will be presented by detailing each approach individually.

3.2.1 Preliminary approach 1: Boresight and Lever-arm from manual matching

The initial objective of this study was to determine the lever-arm and boresight between the long-wave infrared (LWIR) sensor and the integrated camera utilizing hyperspectral data from the LWIR sensor and low-resolution RGB images from the integrated camera.

A dataset comprising 131 points on 10 images was created. These points were utilized to establish corresponding locations on the LWIR hyperspectral data the low-resolution RGB images and on a 3D point cloud generated using the Structure from Motion-Multi-View Stereo (SfM-MVS) workflow on another drone-borne dataset of the same location.

The PnP problem was investigated to find the lever-arm and boresight, however, as demonstrated in the appendix C.1, the results were found to be nor precise nor logical. This can be mainly attributed to the complicated picking of the points on the images and particularly on point cloud, which resulted in huge variations of the searched parameters from image to image. Additionally, the low resolution of the images from the integrated camera on the Telops sensor made it challenging to integrate in the SfM workflow with other high-resolution imagery e.g. obtained from drones.

3.2.2 Preliminary approach 2: Triple matching

To circumvent the limitations imposed by the low resolution of the integrated camera on the Telops Hyper-cam, we added a high-resolution camera, specifically a Nikon Coolpix A, to the system, and semi-automate the snapshots taking and saving. The camera mount system can be viewed in the appendix A.

This proposed method, referred to as **triple matching**, involves matching the LWIR hypercube with its correspondent low-resolution RGB image, followed by matching the low-resolution RGB image with its newly acquired analogous high-resolution RGB image. This ultimately results in the matching of the LWIR hypercube with the high-resolution RGB image. The HR RGB images then be easily integrated with other imagery e.g. obtained from drones, in a Structure from Motion (SfM) workflow to extract the cameras positions and then the matched LWIR data can be projected onto the resulting 3D point cloud.

Assumptions Crucial assumptions are made to ensure the efficacy of this methodology:

- ◇ The sensors are joined together to form one unified system. This results in:
  - ⇒ The system is co-aligned (the focal planes of the sensors are parallel).
- ◇ The target should be approximately orthogonal to the sensor

The figure 3.7 illustrates this second workflow. Initially, the matching between a single pair of low-resolution RGB image and LWIR hypercube was tested, with the relation described by an affine transformation. This yielded visually satisfactory results when the same affine transform was applied to the other pairs. However, this was not the case for the matching between the high-resolution RGB and the LWIR hypercube (even if not initially wanted in our methodology).

Additionally, when attempting to match the high-resolution and low-resolution RGB images using the Scale-Invariant Feature Transform (SIFT) algorithm, it was unable to match the detected keypoints in the two images, presumably due to the wide gap between resolutions of the two RGB types of images.

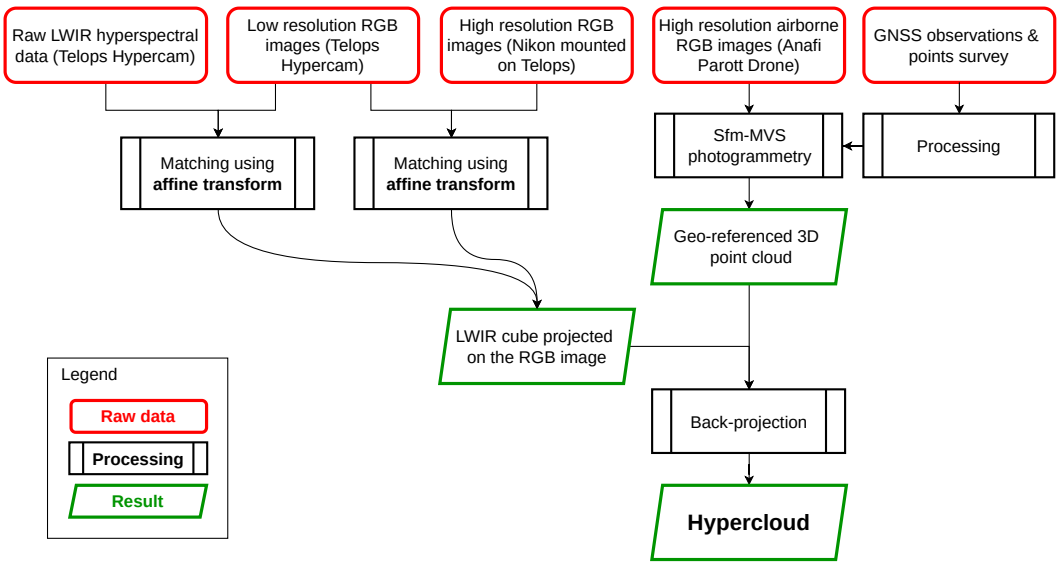


Figure 3.7: Workflow #2

3.2.3 Final approach 3: Stitch and transform

The second approach (triple matching) above, led to the development of our suggested method for LWIR images fusion. The need to construct a larger block of LWIR images arose due to the smaller

field of view (FOV) of the LWIR hypercubes in comparison to the high-resolution RGB camera.

This approach involves stitching the LWIR hypercubes into 2D panoramic mosaics in batches, allowing for each batch to be matched to a single high-resolution image. This not only eradicates the reliance on low-resolution RGB images, but it also requires only a single transformation for each pair of images, which is much faster and less error-prone. The high-resolution RGB imagery can then be integrated with supplementary imagery from the Anafi Parrot drone into a Structure from Motion (SfM) workflow, with incorporating the points that were surveyed using both total station and GNSS receivers. This then enables the extraction of position and orientation information, which can be used to back-project the newly matched LWIR hypercube (hypermosaic) onto the 3D point cloud issued from drone imagery.

The present study employed stitching, also known as mosaicking, to combine multiple images taken from a single position. The assumptions from paragraph 3.2.2 are also considered here. The stitching must preserves the projective nature of the sensor. Two main methods for stitching exist and are: panorama and scan. Panorama mode takes into consideration spherical distortions, while the scan mode simply juxtaposes the images next to each other. The stitching process was based on homography, because images were taken from relatively fixed position and therefore 3D information can not be retrieved. As a consequence it is not possible to extract the orientation indirectly through methods such as Structure from Motion (SfM).

Figure 3.8 shows the logic behind the choice of the homography transformation for the stitching. As seen in the literature review the homography is the best suited for our case where projective geometry should be considered.

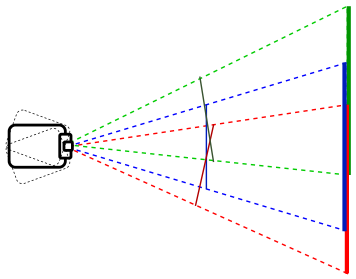


Figure 3.8: Top-view of the sensor capturing images at varying angles of a sub-vertical scene.

The homography is to be calculated using a minimum of four points, which will be automatically detected through the features detection part of the script.

The next step is to determine an approach for matching the LWIR hypercube (hypermosaic) to the high-resolution RGB image.

In order to fully automate the co-registration process, the examination of various feature detection, description, and matching techniques is conducted to identify and match corresponding points in two images.

We settled for the affine transform, because we already used the homography to get the mosaic, and the relation between this hyper-mosaic and the HR RGB image can be described by a mix of translations and scales. For getting the affine transform 4 points will be manually picked on the LWIR mosaic and the HR RGB image.

The figure 3.9 is the workflow that was adopted for the present work.

3.3 Software

3.3.1 Development environment: conda & python

Using a development environment can help to improve the efficiency and organization of the software development process. Development environments typically provide a number of tools

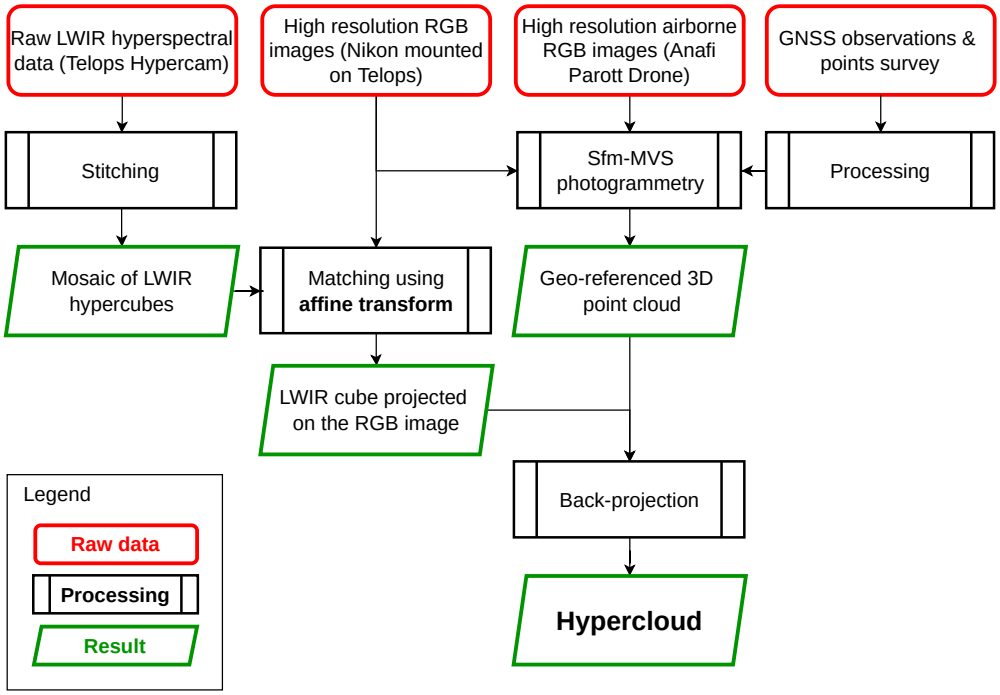


Figure 3.9: Final workflow for co-registering the LWIR data

and resources that can aid in coding, such as code highlighting, debugging, version control, and testing. Additionally, development environments can help to ensure consistency and compatibility across different platforms, making it easier to collaborate with others and deploy the final product. Overall, a development environment can help developers to write better code, faster and reduce the risk of errors.

A development environment like **conda** can also be beneficial for managing dependencies and packages in a software development project. Conda is a package manager and environment management system that can help to manage multiple versions of software packages and their dependencies. This can be particularly useful when working on projects that have many dependencies or when working with different versions of a programming language.

With conda, you can create isolated environments for different projects, each with its own set of dependencies. This can help to avoid conflicts and errors that can arise when different projects have conflicting dependencies. Additionally, conda can also be used to create reproducible environments, which can help to ensure that the code runs consistently across different systems.

Overall, conda can help to streamline the software development process by making it easier to manage dependencies and packages, and by providing a consistent environment for development and deployment.

Python is a versatile and user-friendly programming language that focuses on code readability through the use of significant indentation. It is a high-level, general-purpose language that uses dynamic typing and garbage collection. It can be used for various programming paradigms such as structured, object-oriented and functional programming. Additionally, it's known for its extensive standard library, often referred to as a "batteries included" language.

The main packages used in this work are the following:

hylite

hylite [53] is a free, open-source software package that processes and combines imagery from different types of hyperspectral sensors with high-resolution point-cloud data to create hyperclouds. It also includes tools for analyzing the data, such as methods for mapping, reducing dimensions,

and comparing spectra. Hylite can also use reference spectra from various sources and machine learning techniques to perform classified analyses. And it is based on most of the subsequent packages.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc ...

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

gphoto

gphoto is a set of software applications and libraries for use in digital photography. gPhoto supports not just retrieving of images from camera devices, but also upload and remote controlled configuration and capture, depending on whether the camera supports those features.

pyTorch

pyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, pyTorch also has a C++ interface.

SuperGlue

SuperGlue [54] is a graph neural network that simultaneously performs context aggregation, matching and filtering of local features for wide-baseline pose estimation. It is fast, interpretable, and extremely robust indoors and outdoors.

PyTorch code and pretrained weights at `https://psarlin.com/superglue/` are used for running the SuperGlue matching network on top of SuperPoint [55] keypoints and descriptors.

3.3.2 Notebooks & JupyterLab

Notebook documents contains the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally JSON files and are saved with the `.ipynb` extension. Since JSON is a plain text format, they can be version-controlled and shared with others.

All the workflow is written in notebooks, because they are convenient for exploration, and they can be easily modified to python scripts for faster execution and more organisation.

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

3.3.3 Agisoft

Agisoft PhotoScan is a powerful software solution that enables users to generate professional-grade 3D models from still images. Utilizing the latest advances in multi-view 3D reconstruction technology, the software is able to create high-quality 3D content from a wide range of image sources, regardless of the conditions in which the images were captured. With Agisoft PhotoScan, users can take pictures of an object from any angle, as long as the object is visible in at least two of the images. The software then handles the image alignment and 3D model reconstruction processes automatically, making it easy for users to create accurate and detailed 3D models with minimal effort.

3.3.4 CloudCompare

CloudCompare is a 3D point cloud (and triangular mesh) processing software. It has been originally designed to perform comparison between two dense 3D points clouds (such as the ones acquired with a laser scanner) or between a point cloud and a triangular mesh. It relies on a specific octree structure dedicated to this task. Afterwards, it has been extended to a more generic point cloud processing software, including many advanced algorithms (registration, resampling, color/normal/scalar fields handling, statistics computation, sensor management, interactive or automatic segmentation, display enhancement, etc.).

3.3.5 Autodesk Inventor

Autodesk Inventor is a 3D mechanical design software that allows for simulation, visualization, and documentation. It has the ability to integrate 2D and 3D data in a single environment, allowing users to validate the design before it is built. It also includes parametric, direct edit and free-form modeling tools, and can read multiple CAD formats.

3.3.6 OS

The operating systems utilized in this study consisted of mainly Manjaro and macOS, with the exceptions being the MetaShape and Inventor software which was executed on a Windows machine.

Conclusion This chapter presented the methodology, the hardware and software to carry this work. The following chapter comprises the execution of the methodology utilizing the hardware and software mentioned earlier, as well as the presentation of the obtained outcomes.

Chapter 4

IMPLEMENTATION & RESULTS

Contents

4.1 Site location & Setup

The chosen target for this study is a wall of a quarry situated in Naundorf, a location in close proximity to Freiberg, Sachsen, Germany. Figure 4.1 shows where the Naundorf quarry is located.

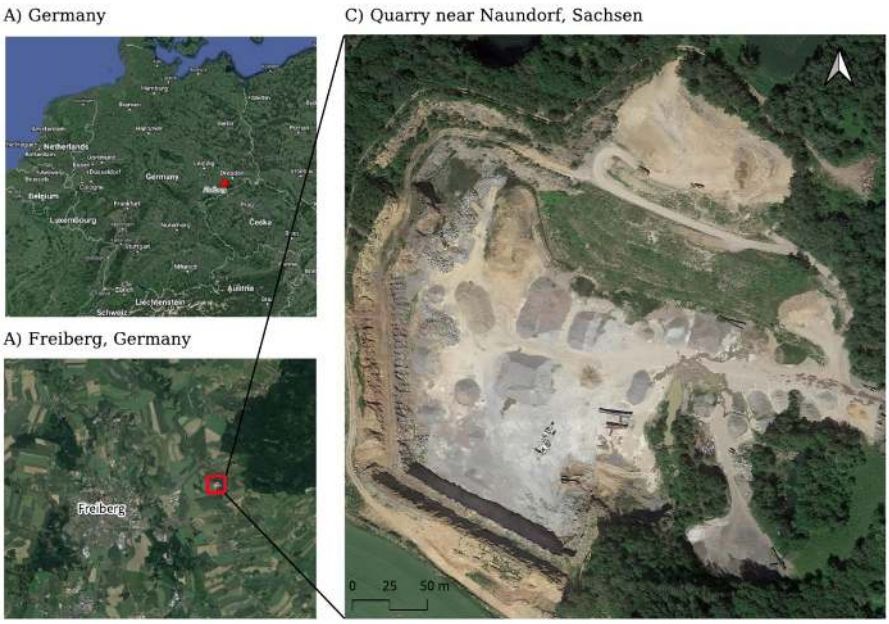


Figure 4.1: Map situation of the Naundorf quarry

Figure 4.2 represents the target wall that was remotely sensed and how the data acquisition was carried out. The images (the blue squares in the figure) taken by the new system containing the Telops Hyper-Cam and Nikon camera in a way to follow a snake shape (the white arrow in the figure) with an overlap greater than 40% between every two consecutive images. That is to smooth the process of stitching as it will be portrayed in next section.

The drone was flown near the sub-vertical wall of the quarry to capture images with a significant overlap and from a wide range of angles.

Figure 4.2: Target wall and imaging procedure: the images (blue squares) were taken as to make a snake shape (white arrow)

4.2 Camera mount

As per the methodology, the Nikon camera was mounted onto the Telops Hyper-cam so they make a solid unit. The design and prototyping of the camera mount was a multi-step process. The initial stage involved the conceptualization of the mount through sketching a detailed drawing. The design principle proposed was to utilize a sliding system in combination with gravity, in order to achieve a simple yet robust and stable design. This was then followed by the collection of measurements from both sensors, then a digital model of the mounting system is created using the Autodesk Inventor software. The 3D model was exported in the `.stl` format, which was then utilized to fabricate the camera mount prototype using the PURSA 3D printer.

The proposed design and final product of the camera mount prototype can be viewed in figure 4.3. Additional information and images pertaining to the camera mount system and its conception can be found in appendix A.

A script has been developed to take photos programmatically using the gphoto tool. The code can be found in appendix C.2.

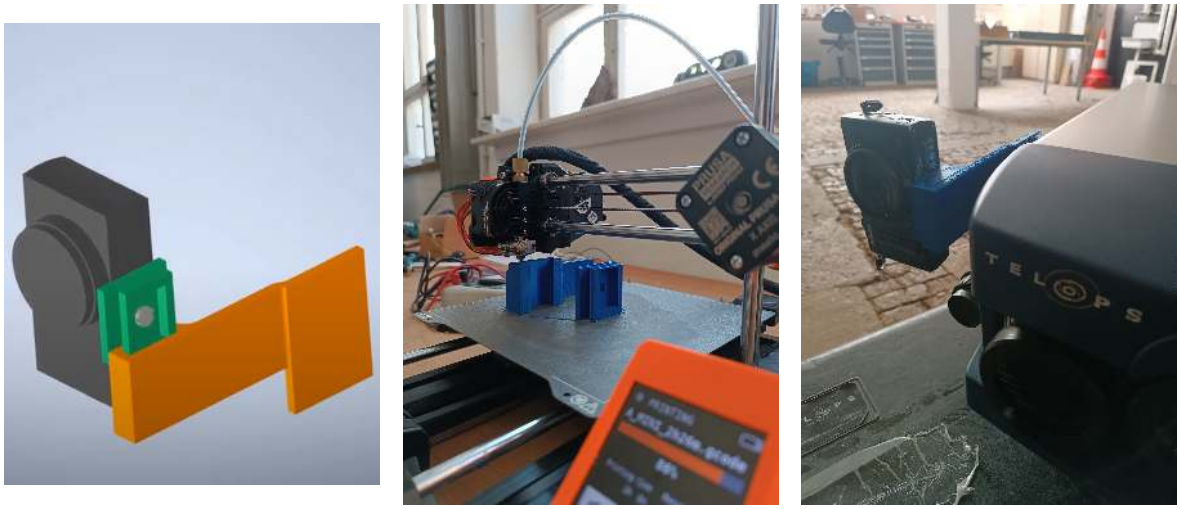


Figure 4.3: 3D modeling, printing of the camera mount for the Telops Hyper-Cam

4.3 Camera calibration

In this project, camera calibration was performed on two different sensors using two methods. The first method involved imaging a checkerboard from different angles, then the distortions are corrected after being extracted using Zhang's method. The second method is based on the Structure from Motion (SfM) workflow, when the distortions coefficients are calculated during the photo alignment step.

The Nikon camera images were processed using the SfM workflow, resulting in automatic calibration, the results are shown in listing 4.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<calibration>
  <projection>frame</projection>
  <width>4928</width>
  <height>3264</height>
  <f>3823.3577829739975</f>
  <cx>13.699199493932149</cx>
  <cy>9.4995795493603286</cy>
  <k1>-0.07547632130169811</k1>
  <k2>0.086542087371911089</k2>
  <k3>-0.024902186357503377</k3>
  <p1>-1.0002076989788903e-05</p1>
  <p2>-0.00020814400112626738</p2>
</calibration>
```

Listing 4.1: Nikon camera coefficients

On the other hand, the low resolution images from the integrated RGB camera of the Hyper-Cam could not be processed using the SfM workflow, so manual calibration using Zhang's method was implemented.

A practical camera calibration is done as follows:

- Shoot a checkerboard from all angles using the camera (the so-called "calibration dance") as can be seen in the figure 4.4.
- Extract the corners of the checkerboard in the images
- Establish the mapping relationship between the coordinates of the corner points in the image and the coordinates of the corner points in the world coordinate system
- Calculate the camera's internal and external parameters.

The code developed for the calibration procedure is provided in the appendix C.3.

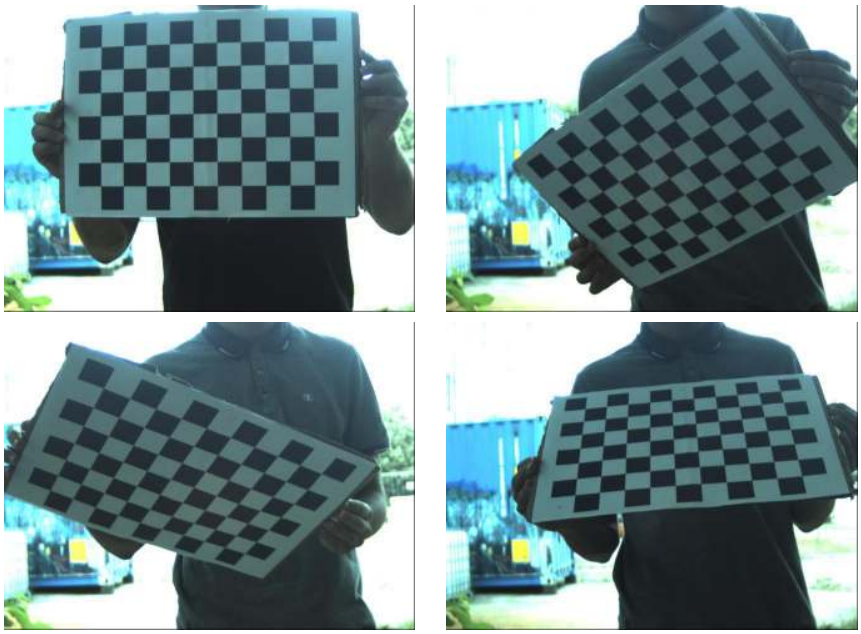


Figure 4.4: The calibration "dance": shooting a checkerboard from different angles

4.4 Stitching

For the stitching, SIFT algorithm is used to detect correspondent points on every couple of images, then a homography is calculated and applied to stitch one single image to the obtained previous block.

The script is developed using OpenCV and have advanced features such as the ability to choose from a variety of feature detectors, including ORB, SIFT (our choice), BRISK, and AKAZE, as well as different pairwise image matching methods, such as affine and homography (our choice). The result of the stitching process was a panorama image that we will call a hypermosaic. The script is included in appendix C.7.

The stitching is applied on 30 of the broadband infrared images (previews of the LWIR hypercubes), its result can be seen in figure 4.5, and then applied to every band of the hypercubes using the code in appendix C.4 to get hyper-mosaic as shown in figure 4.6.

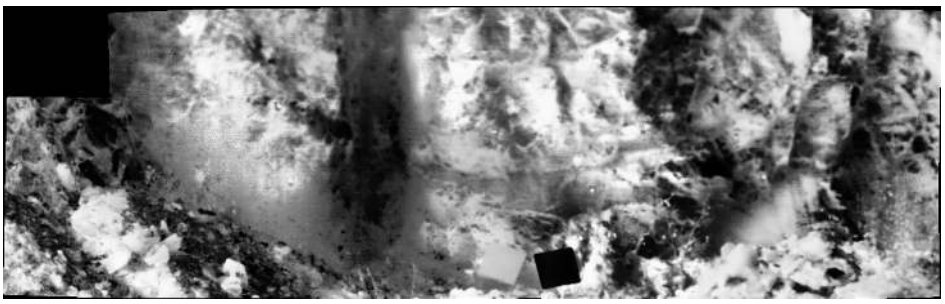


Figure 4.5: Stitched broadband infrared image (preview of the LWIR hypercube)

4.5 SfM workflow

The SfM workflow in Agisoft Metashape involves several steps, including image alignment, dense point cloud generation, and mesh and texture generation. In the image alignment step, the software analyzes the images and finds common features to align them. This is done automatically, but users

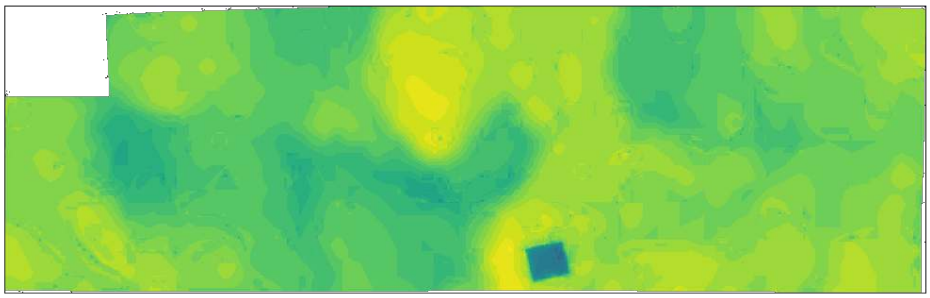


Figure 4.6: Stitched LWIR hypercubes (one band view)

can adjust the alignment settings and parameters to achieve better results. For example, users can choose between different alignment modes, such as generic, accurate or high accuracy, depending on the level of accuracy required for their project.

Once the images are aligned, the software generates a dense point cloud using a depth map calculation. This step involves estimating the distance between each pixel in the images and the camera location to create a 3D point cloud. The dense point cloud can be further processed to remove outliers and refine the geometry. The software offers several point cloud filtering and editing tools, such as noise filtering, hole filling, and smoothing, to improve the quality of the point cloud.

After the dense point cloud is generated and refined, a mesh can be created using the surface reconstruction algorithm. The mesh represents the surface of the object or scene, and can be further edited and optimized using the mesh editing tools available in the software. Users can choose between different mesh generation methods, such as Delaunay, Poisson or Height Field, depending on the characteristics of their project.

Finally, a texture can be added to the mesh to create a realistic 3D model. The texture is generated by projecting the images onto the mesh and blending them together. Users can adjust the texture settings, such as resolution, quality, and blending mode, to achieve the desired level of detail and realism.

Overall, Agisoft Metashape provides a powerful and flexible SfM workflow that can be customized to fit a wide range of applications and projects. By offering a variety of options and features, the software enables users to generate accurate and detailed 3D models from photographs with ease and efficiency.

The comprehensive list of steps can be accessed through the following User Manuals page https://www.agisoft.com/downloads/user-manuals/.

The two primary outputs of interest from the SfM process are the 3D dense point cloud and the camera positions. Appendix B is the report generated from Metashape Agisoft software.

4.6 Matching and Transformation

As part of a comprehensive effort towards fully automating the co-registration process, this study endeavors to examine the feasibility of utilizing feature detection and description techniques such as Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB) as well as deep learning-based approaches including SuperPoint and SuperGlue for the identification and correspondence of keypoints in LWIR hypercube and high-resolution RGB image. And this was tested on multiple combination of the original dataset and the invert of the LWIR preview and different grayscale versions of the RGB image using different weights.

The thorough evaluation of these methods did not produce satisfactory results, the detailed attempts can consulted on these slides at https://tinyurl.com/autoSuperPG, and appendix C.5 code for automating the prediction and plotting of the results of the network of different input parameters.

The affine transform was employed, which requires a minimum of three corresponding points in both images to calculate. To investigate the optimal number and distribution of points needed, we conducted experiments using 4 points and 8 points from the 16 available points in a spatially distributed manner e.g avoiding clustering of points in one side or middle of the target. The results obtained from these experiments were found to be approximately the same. Equation 4.1 presents the obtained affine transform.

$$M_{aff} = \begin{bmatrix} 1.34 & 3.62e^{-2} & 5.53e^{2} \\ -3.31e^{-2} & 1.36 & 1.99e^{3} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Manual selection of these points is carried on using the napari GUI and affinder extension. With a subsequent calculation of the mathematical matrix describing the transformation utilizing the coordinates of the selected points.

Once we have this affine transformation, the LWIR hypercube is now transformed to the size of the RGB image, in exact alignment as portrayed in figure 4.7.

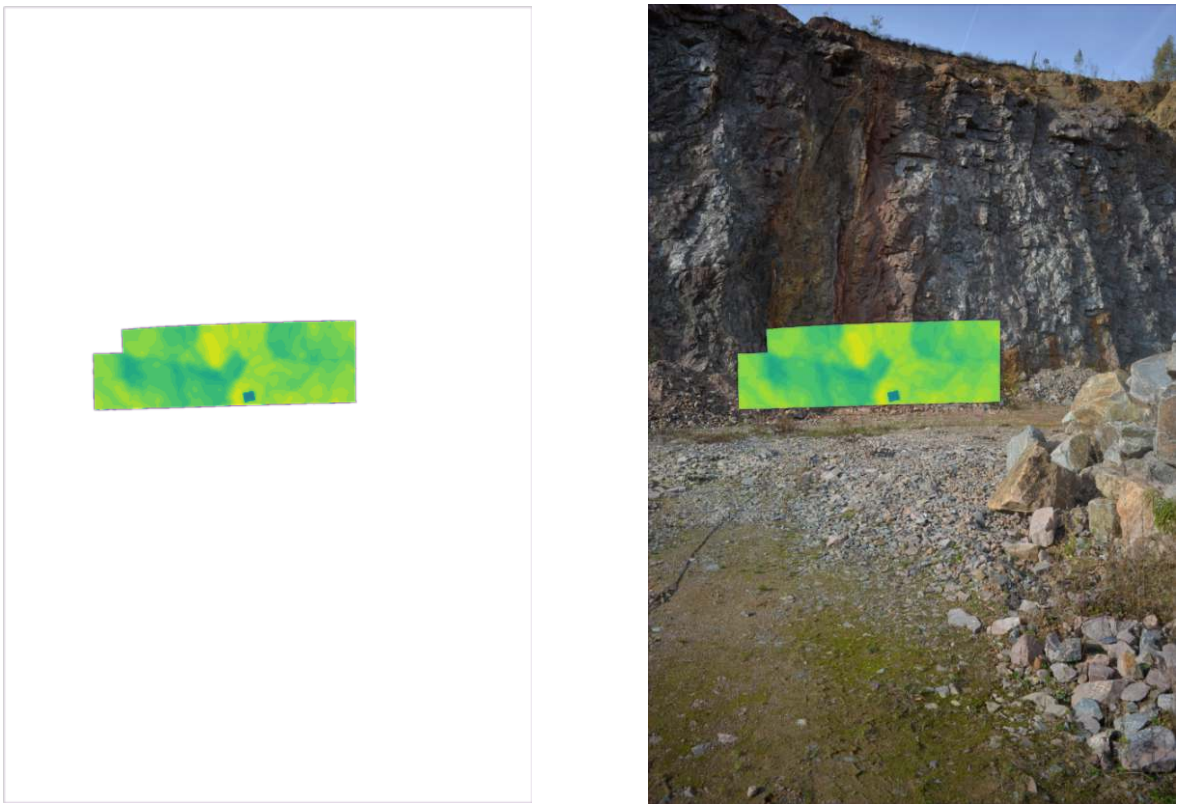


Figure 4.7: Transformed stitched LWIR hypercube and overlaid on the RGB image.

See appendix C.6 for the code proposed to extract the affine transformation.

4.7 Back-projection

The cameras positions and orientations were exported from the software where the SfM workflow took place, for convenience the best format is OPK (Omega, Phi, Kappa). A sample line from the exported file can be seen in listing 4.2.

With position and orientation of the camera issued from the SfM workflow of the RGB frame, it is possible to perform re-projection of the recently transformed LWIR hypercube via the use of the hylite. The code to achieve that is shown in listing 4.3.

The resulting partial hypercloud is depicted in figure 4.8.

```
# Cameras (...)
# PhotoID, X, Y, Z, Omega, Phi, Kappa, r11, r12, r13, r21, r22, r23, r31, r32
    , r33
DSC_5418  13.42 50.92 408.40  79.26 67.34 -80.12  0.06  -0.027  -0.99 0.37
    0.92  -0.00 0.92  -0.37 0.07
```

Listing 4.2: File structure of exported OPK of the cameras

```
# Import needed libraries
import numpy as np
import hylite
from hylite import io

# Load point cloud data from a .ply file
cloud = io.load('data/sfm/pointCloud-utm32633-2cm.ply')

# Load camera data from a .cam file
cam = io.load('camera.cam')

# Load image data from a .hdr file
hycb = io.load('result/hypermosaic.hdr')

# Replace values of 0 in the image data with NaN
hycb.set_as_nan(0)

# Project the image onto the point cloud data using the camera position and
    orientation
cloud2 = cloud.copy(data=False)
cloud2.project(hycb, cam, trim=False)

# Save the result as a .ply file
io.save('result/hypercloudLwir-posSfM-f.ply', cloud2)
```

Listing 4.3: Code developed to back-project an image onto a point cloud

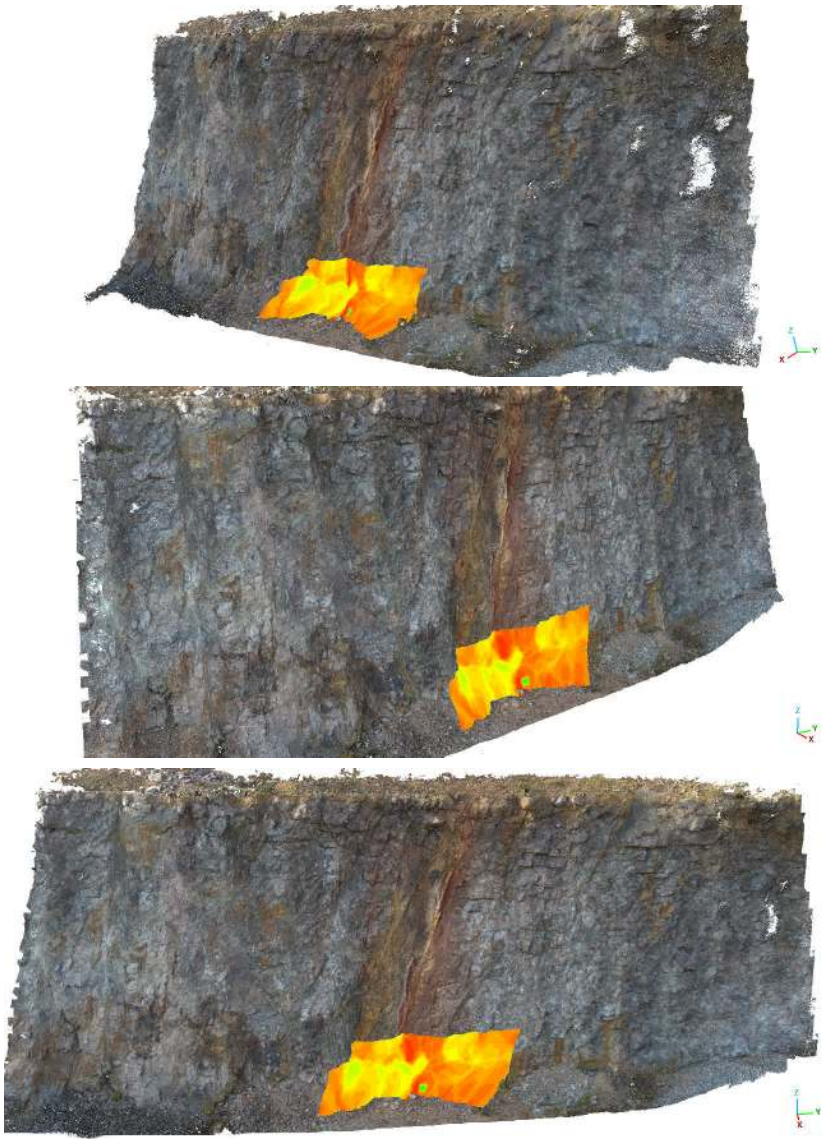


Figure 4.8: The resulting hypercloud as seen from different angles.

Chapter 5

DISCUSSION

The structure of the discussion is primarily driven by the methodology used in the study. Each component will be examined individually.

Site The selection of the site location was favorable because of the sub-vertical wall in the quarry, although optimal conditions would have been better if the sun illumination was direct to the wall, and higher temperature. The data collection for this study was conducted in the month of November, characterized by low temperatures and overcast sky conditions. The late hour of data collection is evidenced by the shadows present on the quarry wall.

The weather conditions and limited sun exposure may impact the quality of the collected data especially for the LWIR range of the EM spectrum. It was the case for this study as we could not produce useful temperature and emissivity map using the Temperature and Emissivity Sepatation (TES) algorithm.

Nonetheless, the collected data still provides a good value of the main goal of this study the "geometric" co-registration and co-alignment.

The distance between the sensors and the target was approximately 30 meters, which posed a significant challenge for achieving co-registration due to the small field of view (FOV) of the Telops Hyper-Cam. The movement of the sensor and co-registration becomes even more challenging as the target gets closer. Conversely, if the target is positioned farther away to a certain degree, it may be possible to capture a viable part of the scene that can be used for matching, but also limited by the FOV.

Mounting system Despite the nature of the mounting system, it was found to be somewhat shaky, likely due to the thin arm of the system. However, this issue can be easily addressed by adding some thickness to the arm in the 3D model and subsequently reprinting it.

Also, the stability and the ease of use of the mounting system can be ensured with the utilization of a secure cable and a picture-taking script.

Stitching The smaller field of view of Telops Hyper-Cam compared to the HR RGB camera made it necessary to create a larger block of LWIR images. Having a mosaic of hypercubes to be matched to a single HR image made it much easier and less error-prone than finding a transformation between every hypercube and its correspondent HR RGB image.

The usefulness of stitching can be shown in the following paragraph, which discusses matching and transformation. To confirm the accuracy of the stitching process, the selected points were re-projected.

The results of another experiment on another dataset indicate the importance of the order of the pictures that should follow a snake shape to make the stitching as smooth as possible and ensuring sufficient overlap. That's can be also done with rearranging the images order when feeding

them to the stitching algorithm.

Matching and Transformation The inefficacy of the classical feature detection and matching algorithms as well as the ones based on deep learning in our experiment can be attributed to the disparities in the manner in which features are perceived within the visible and Long-Wave Infrared segments of the electromagnetic spectrum. Furthermore, it is important to note that the deep learning networks were trained using a dataset consisting solely of RGB images.

The decision was made to utilize an affine transform, as the homography had already been utilized to create the mosaic. Thus the connection between the hyper-mosaic and the HR RGB image can be described as a combination of translations and scales. The affine transform was obtained by manual selection of four points on both the LWIR mosaic and HR RGB image.

To validate the approach, keypoints in both the hyper-mosaic and the HR RGB image were chosen, which included the heat packs and the corners of the calibration panels. The calculated affine matrix from 4 manually selected points was utilized to transform the remaining of 16 points, and the resulting differences were compared. This comparison is shown in table 5.1.

Table 5.1: Validation of the affine transform (using 4 points) in *pixels*

id	x	y	trans_x	trans_y	dif_x (px)	dif_y (px)
0	2006	869	2012	867	6	2
2	2277	1049	2274	1049	3	0
3	2044	1351	2046	1349	2	2
4	2183	1403	2179	1399	4	4
5	2278	1569	2274	1566	4	3
7	2247	1861	2243	1859	4	2
8	2368	1401	2372	1400	4	1
9	2392	1459	2391	1458	1	1
10	2423	1377	2423	1375	0	2
11	2443	1437	2445	1436	2	1
13	2378	1536	2375	1535	3	1
14	2449	1488	2450	1487	1	1
				min	0	0
				max	6	4
				std	1.70	1.07
				mean	**2.83**	**1.67**
				RMSE	**3.27**	**1.96**

The average re-projection errors for the selected points were found to be relatively small with a mean of $dx = 3$ pixels and $dy = 2$ pixels. It is worth noting that the Telops Hyper-Cam is equipped with an FPA detector that has a resolution of 320 x 256 pixels, which makes these errors insignificant compared to the overall image and the distance to the target being 30 meters.

At such distance this results in roughly an error of 6 mm in the scene. Which is very usable in the context of geological mapping.

SfM In the realm of data processing, the achievement of high-quality results often comes at the expense of both computational and storage costs. In other words, the greater the desire for optimal results, the more time and resources are typically required to process and store the data. This challenge presents a fundamental dilemma that necessitates careful consideration of the desired output before beginning the processing stage. It is imperative to balance the need for high-quality results with the availability of computational resources and storage capacity, to ensure that the processing is feasible and efficient.

A recommended approach is to start small with low quality results and scale out and up as needed, this will also allow for rapid exploration of different parameters faster. That said, the Photo alignment step should always be conducted at the highest setting possible (considering

computational power available).

The execution of the Structure from Motion (SfM) workflow is performed using the Metashape software. To export the resultant 3D point cloud, it is crucial to employ a projected reference system, such as the Universal Transverse Mercator (UTM) family. This is because, if a geographic coordinate system is used, the 3D point cloud will be exported as a long stripe of points, which is an erroneous result. The proper selection of the reference system is essential for ensuring the quality and validity of the generated results.

It was not attempted to cover all possible permutations of SFM workflow. The specific SFM workflow used was designed for the imagery collected for a quarry wall. Various factors, such as the nature of the imagery and the desired resolution of the outputs, can influence the choices made during the workflow.

In our workflow, the primary outputs are the point cloud that serves as the foundation for our hypercloud and the camera positions and orientations that are employed in the back-projection. While we utilized "high" for the different parameters (but not ultra-high), we can confirm that these outcomes are appropriate for our objectives based on the report's findings.

Back-projection The most important output from the SfM workflow is the cameras positions and orientations, which will be needed in the back-projection. As the quality of the back-projection depends on the quality of the transformation between the hyper-mosaic and the HR RGB image from one side, and the camera position and orientation from the other side.

The heat packs and the corners of the calibration panels will serve as a the main elements for our validation method, we picked these points on the RGB 3D point cloud, and on the newly projected hyper-mosaic in the same point cloud; we only changed which bands to view. Then we calculated the difference in the 3 axis: x, y and z and then we proceeded to calculate the 3D euclidean distance using this formula: $d(P_1, P_2) = \sqrt{(x_1^2 - x_2^2)^2 + (y_1^2 - y_2^2)^2 + (z_1^2 - z_2^2)^2}$.

Table 5.2 presents the calculated quantities and the coordinates of the points.

Table 5.2: Validation of hypercube back-projection onto the point cloud. The table contains the differences in the x, y, and z coordinates of selected points and the calculated 3D Euclidean distance

id	x	y	z	x_LWIR	y_LWIR	z_LWIR	dx (cm)	dy (cm)	dz (cm)	euc3d (cm)
0	389.271,80	5.642.927,15	409,67	389.271,81	5.642.927,15	409,65	0,88	0,28	2,13	2,32
1	389.272,50	5.642.928,07	408,65	389.272,51	5.642.928,05	408,61	1,25	2,45	3,66	4,58
2	389.272,76	5.642.928,13	407,73	389.272,76	5.642.928,14	407,73	0,32	1,60	0,01	1,63
3	389.272,01	5.642.931,04	409,48	389.272,01	5.642.931,05	409,51	0,31	1,52	2,27	2,75
4	389.272,48	5.642.931,22	408,42	389.272,48	5.642.931,19	408,40	0,34	3,18	1,83	3,69
5	389.273,12	5.642.932,34	407,70	389.273,14	5.642.932,37	407,70	1,50	2,84	0,15	3,22
6	389.272,54	5.642.934,63	409,42	389.272,56	5.642.934,70	409,45	1,82	7,10	3,02	7,93
7	389.273,27	5.642.930,77	407,01	389.273,23	5.642.930,76	407,01	4,15	0,73	0,11	4,22
8	389.273,51	5.642.931,10	406,87	389.273,52	5.642.931,11	406,84	0,69	1,55	2,84	3,31
9	389.273,38	5.642.930,47	406,63	389.273,37	5.642.930,51	406,64	1,01	3,83	1,09	4,11
10	389.273,69	5.642.930,82	406,52	389.273,67	5.642.930,82	406,52	1,88	0,74	0,70	2,14
11	389.273,60	5.642.931,20	406,86	389.273,59	5.642.931,19	406,85	0,69	1,08	0,71	1,47
12	389.273,78	5.642.931,60	406,94	389.273,76	5.642.931,61	406,95	1,31	1,52	1,41	2,45
13	389.273,91	5.642.931,11	406,45	389.273,97	5.642.931,11	406,44	6,58	0,13	1,71	6,80
14	389.274,08	5.642.931,56	406,56	389.274,10	5.642.931,58	406,56	1,89	2,00	0,11	2,75
						min	0,31	0,13	0,01	1,47
						max	6,58	7,10	3,66	7,93
						std	1,67	1,75	1,17	**1,81**
						mean	1,64	2,04	1,45	**3,56**

The average discrepancy between the original points and the projected ones is determined to be **3.56cm**, with a standard deviation of **1.81cm**. While this margin of error may be substantial for a surveyor, it is considered to be within acceptable tolerance levels in the field of geology. This is due to the fact that geological studies typically involve working on large scales, and the focus is on the

overall picture. And even if the location is off by a dozen of centimeters, it does not have deadly consequences.

Incorporating LWIR data into hyperclouds also has the potential to improve the accuracy of geological mapping. By combining LWIR data with SWIR and structural information, it may be possible to differentiate between different types of mineral deposits and accurately map their boundaries. This could significantly improve the efficiency and effectiveness of mineral exploration programs by providing more accurate and detailed information about the subsurface.

Chapter 6

CONCLUSIONS & RECOMMENDATIONS

Hypercloud are the future of hyperspectral data as combination of visualization and processing are made easy. However, a significant challenge in the construction of hyperclouds lies in the process of co-registration and data fusion.

In this research, a semi-automatic co-registration workflow was developed, which only requires minimal human interaction in the selection of four points on two images. The methodology was applied to the fusion of data acquired in the Long-Wave Infrared portion of the electromagnetic spectrum from a Telops Hyper-cam to a point cloud data, but it could also be extended to a wide range of other sensors with similar constraints as in our case.

The methodology commences with a process of Camera Calibration to eliminate initial geometric distortions. Subsequently, the stitching of small Long-Wave Infrared (LWIR) hypercubes obtained from the Hyper-cam device is performed to obtain a hyper-mosaic. The next step involves Matching and Transformation to establish an affine transformation between the hyper-mosaic and a selected RGB image. The Structure-from-Motion (SfM) workflow is then executed using images acquired from a drone and the mounted RGB camera on the Hyper-cam to generate the Point Cloud, as well as the camera position and orientation of each frame. Upon determination of the RGB frame's camera position and orientation, which was matched with the hyper-mosaic, a back-projection is performed onto the Point Cloud to generate the partial hypercloud. The generated hypercloud can then be populated with data from additional regions of the Electromagnetic spectrum, subject to analysis and subsequently utilized to produce various products, including mineral maps.

The successful semi-automatic co-registration of hyperspectral LWIR data to the Hypercloud has added value to the mining industry by making it easier to combine it to other sources, resulting in a more comprehensive understanding of geological features and processes can be obtained, which in turn can lead to more informed decision-making.

In light of the findings and results obtained throughout the course of this thesis, it is essential to offer several recommendations aimed at enhancing future research. The following suggestions are proposed to enhance the use of the developed solution:

- Further investigation into alternative feature detection and matching techniques, particularly those between the LWIR and RGB data and other deep networks for co-registration
- Investigation into the potential integration of additional sensors and data sources to increase the overall accuracy and precision of the co-registration process
- Study of the scalability and robustness of the semi-automatic co-registration workflow presented in this thesis, with a focus on its application to large-scale datasets
- Development of a graphical user interface (GUI) to facilitate the triggering of image acquisi-

tion from the Nikon camera through a dedicated button

These aspects have the potential to guide future research:

- Constructing a dataset of matched points in LWIR and high-resolution (HR) RGB images.
- Implementation of transfer learning to utilize the pre-trained weights of the first layers of deep networks and further fine-tune the model on the new dataset

APPENDICES

Appendix A

Camera Mounting System

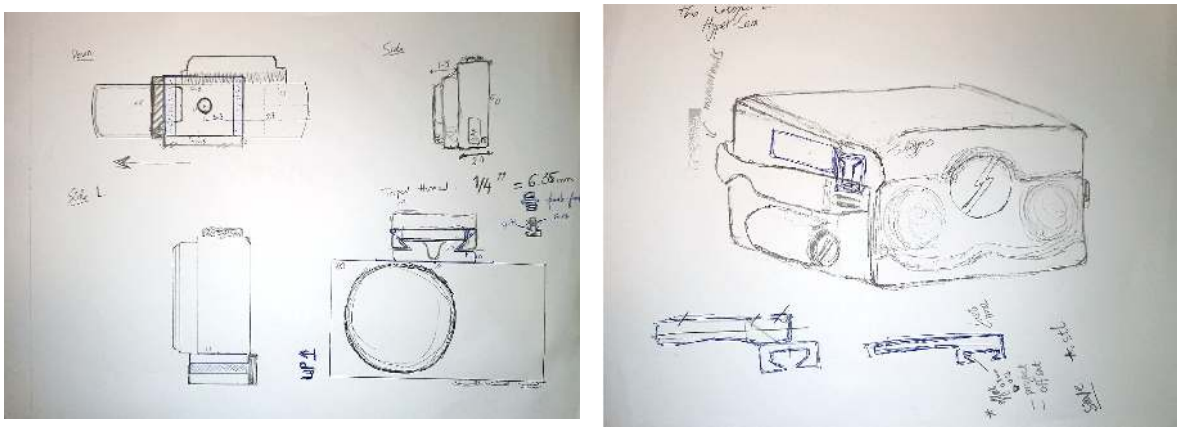
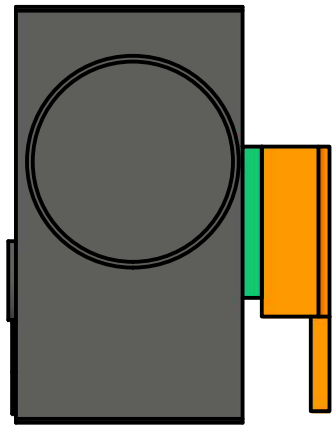
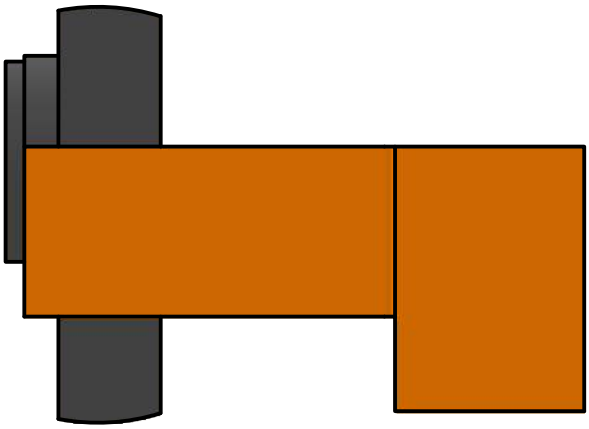


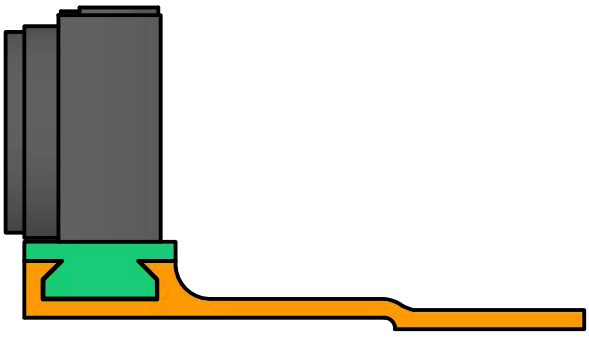
Figure A.1: First sketch of the camera mount conception

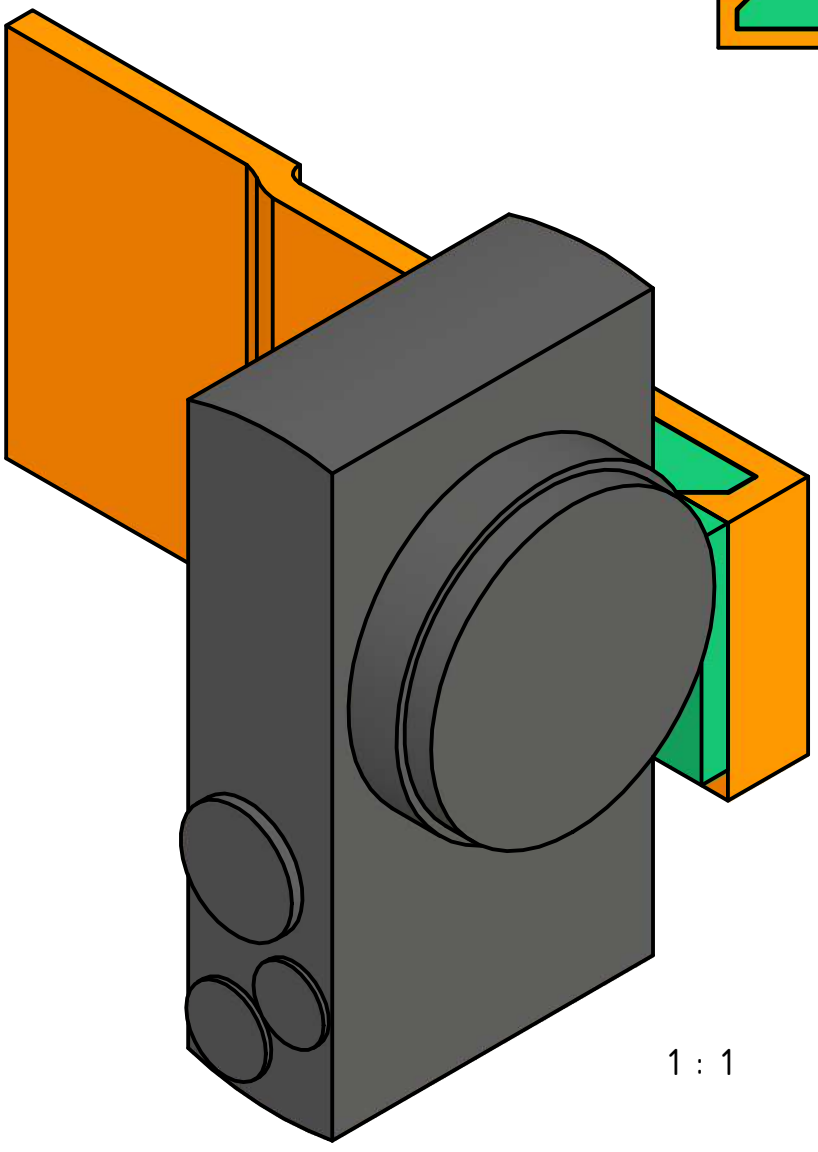
Figure A.2: Nikon camera mounted to the Telops Hyper-Cam





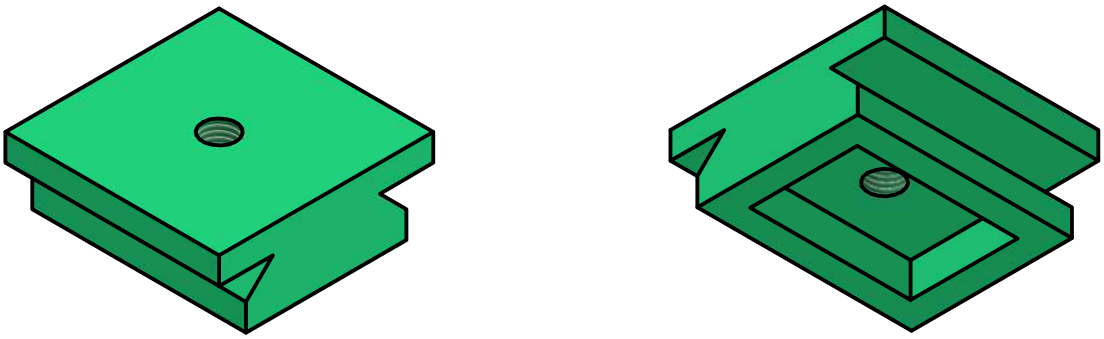
1 : 2

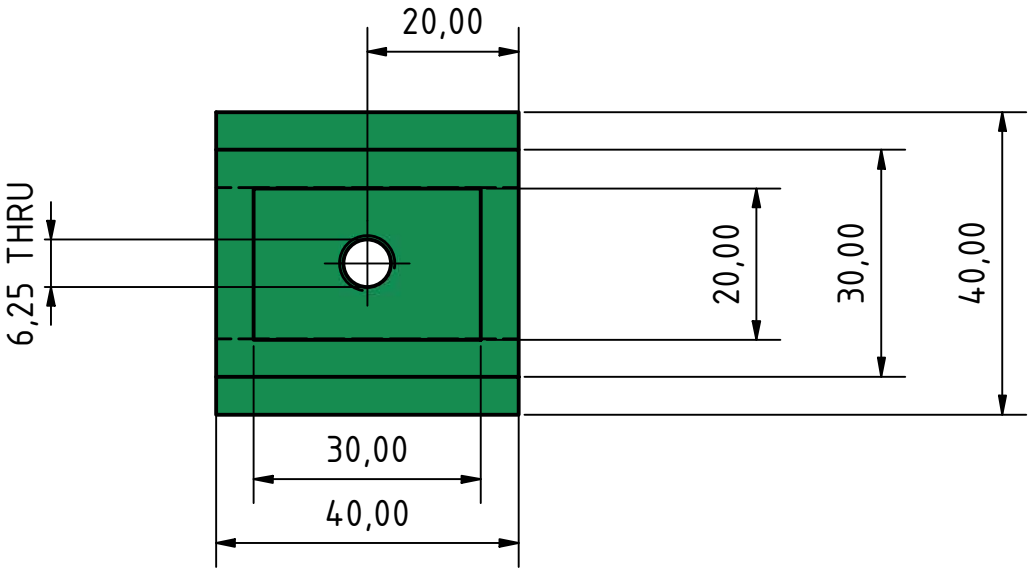


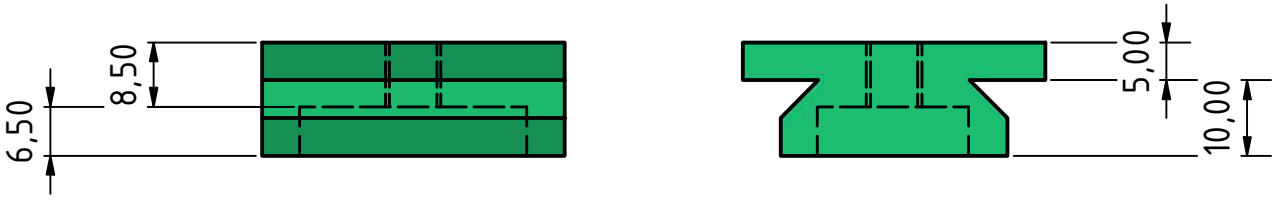


Camera mounting system

1 : 1

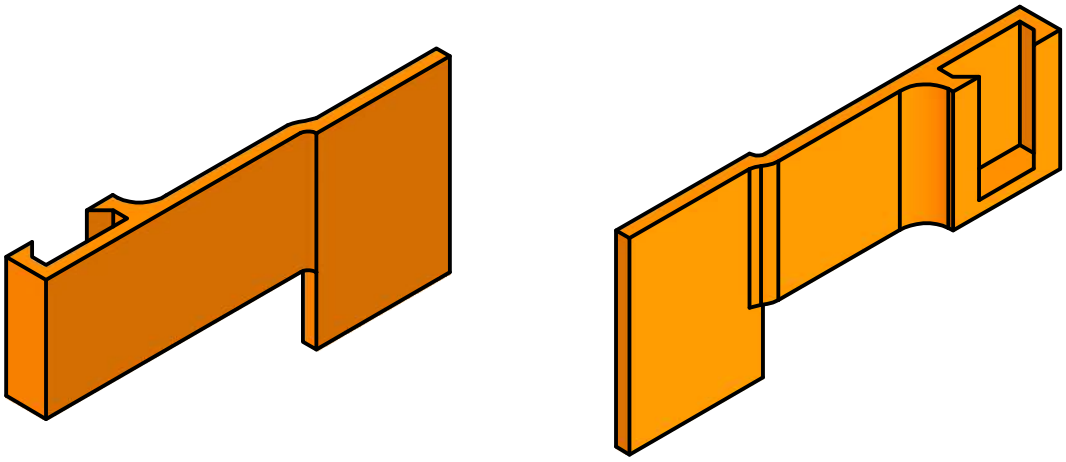


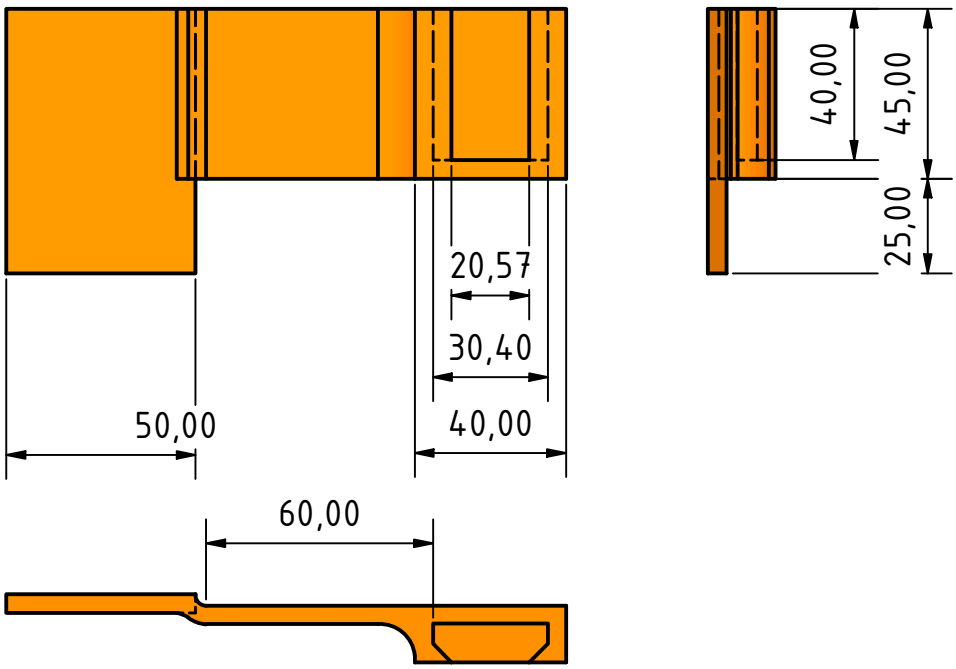




Conceived	Ayoub Fatihi
Drawn	Ayoub Fatihi
Date	08/08/2022

Camera mounting part



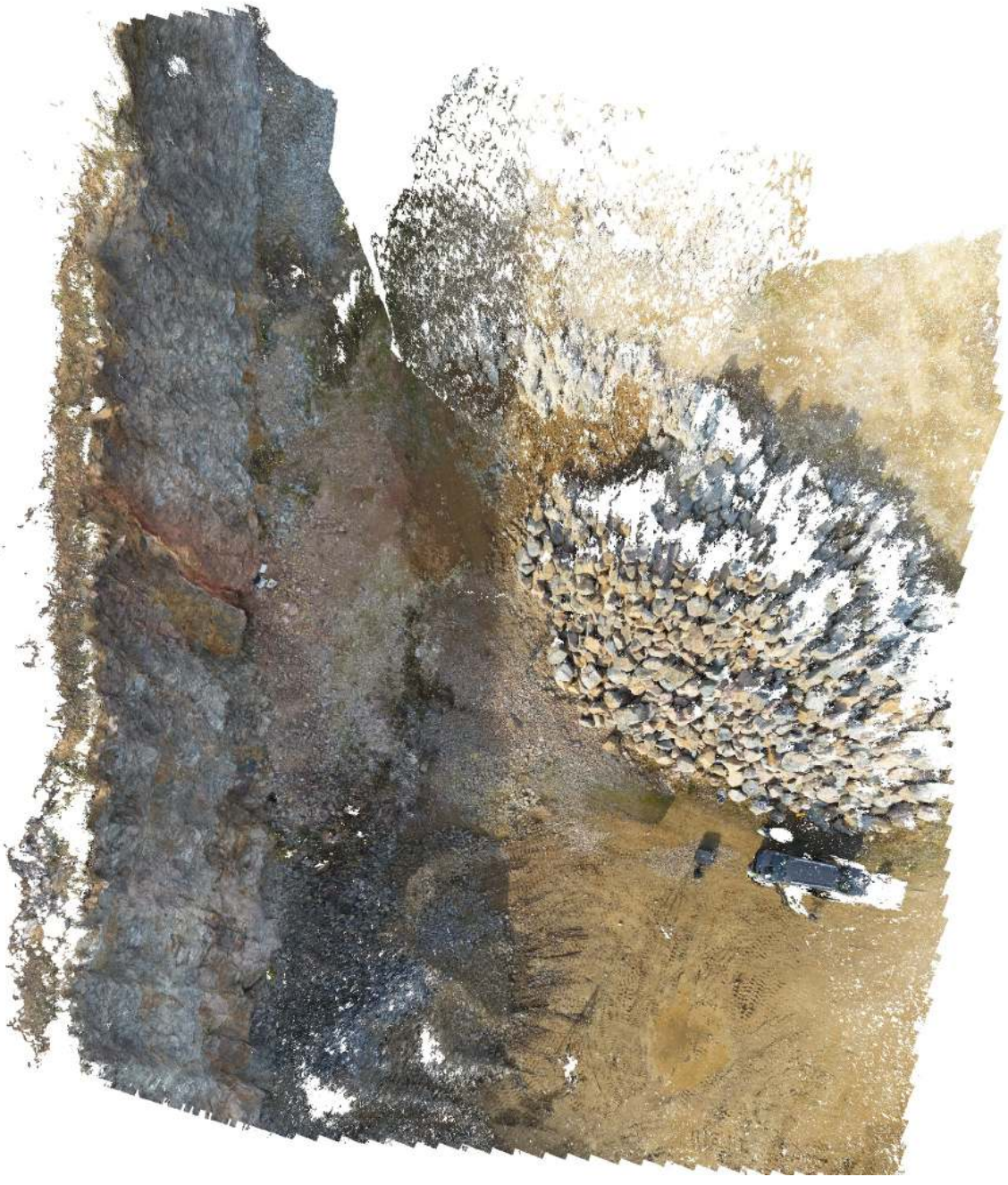


Conceived	Ayoub Fatihi
Drawn	Ayoub Fatihi
Date	08/08/2022

Telops mounting part

Appendix B

Agisoft Metashape – Processing Report



Survey Data

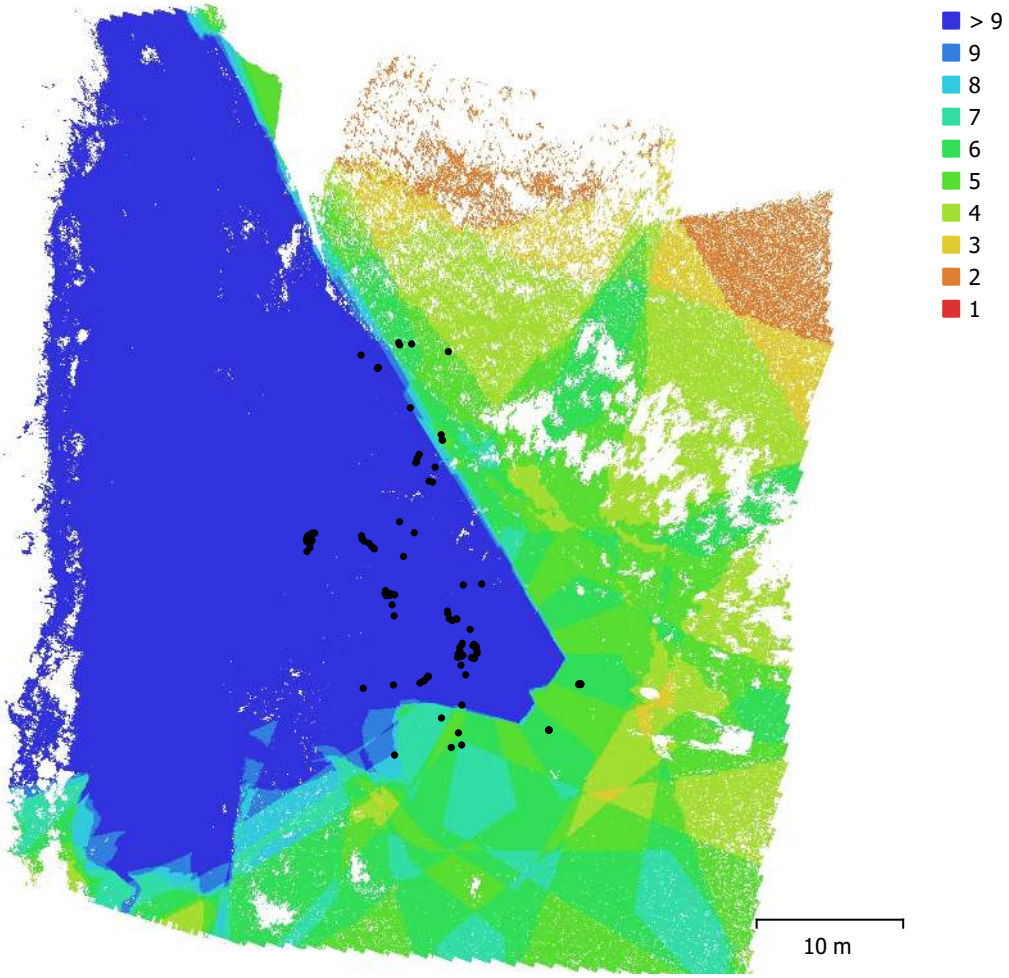


Fig. 1. Camera locations and image overlap.

Number of images:	145	Camera stations:	141
Flying altitude:	18.2 m	Tie points:	30,502
Ground resolution:	6.4 mm/pix	Projections:	321,149
Coverage area:	2.53e+03 m²	Reprojection error:	0.646 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
COOLPIX A (18.5mm)	4928 x 3264	18.5 mm	4.84 x 4.84 µm	No
Anafi (4mm)	4608 x 3456	4 mm	1.34 x 1.34 µm	No
Anafi (4mm)	4608 x 3456	4 mm	1.34 x 1.34 µm	No

Table 1. Cameras.

Camera Calibration

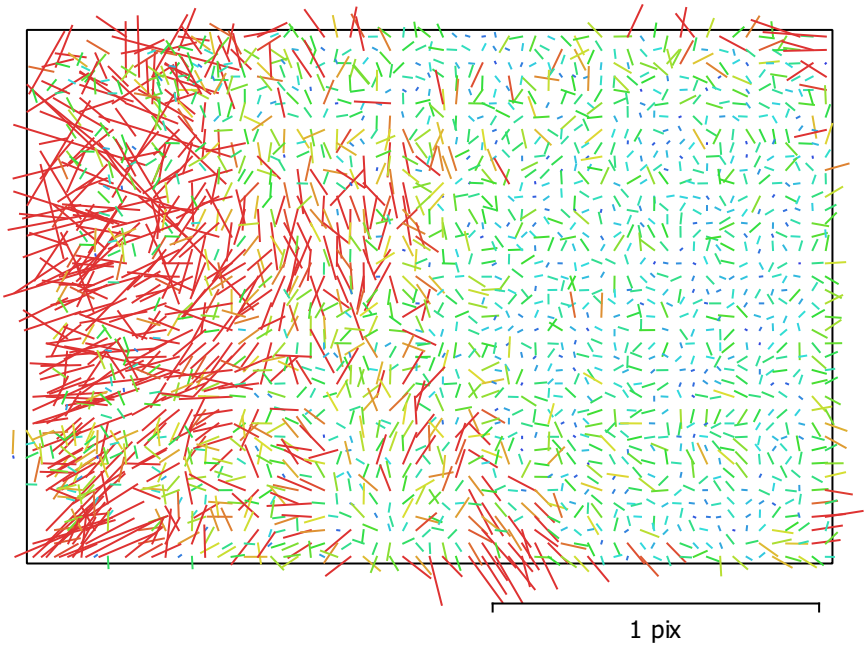


Fig. 2. Image residuals for COOLPIX A (18.5mm).

COOLPIX A (18.5mm)

34 images

Type	Resolution	Focal Length	Pixel Size
Frame	**4928 x 3264**	**18.5 mm**	**4.84 x 4.84 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	P1	P2
F	**3847.71**	0.22	1.00	-0.06	-0.08	-0.01	0.24	-0.19	0.20	-0.06
Cx	**1.41494**	0.34		1.00	0.07	-0.11	-0.08	0.11	0.83	-0.00
Cy	**10.786**	0.45			1.00	-0.18	0.10	-0.06	-0.05	0.96
K1	**-0.0725665**	0.00014				1.00	-0.90	0.84	0.00	-0.16
K2	**0.0801742**	0.0005					1.00	-0.98	-0.04	0.09
K3	**-0.0183036**	0.00055						1.00	0.07	-0.05
P1	**-0.00041456**	2.3e-05							1.00	-0.10
P2	**-0.000143972**	3.2e-05								1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Calibration

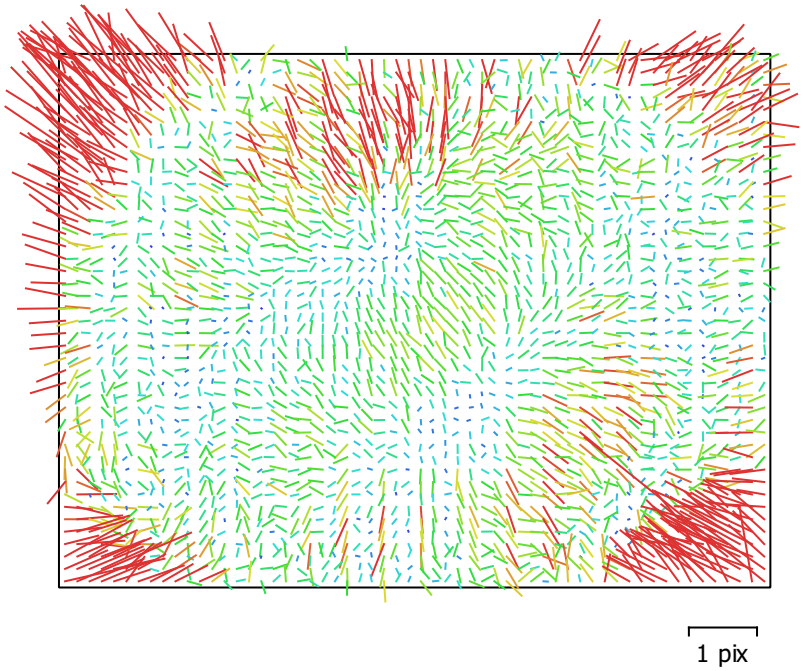


Fig. 3. Image residuals for Anafi (4mm).

Anafi (4mm)

105 images

Type	Resolution	Focal Length	Pixel Size
Frame	**4608 x 3456**	**4 mm**	**1.34 x 1.34 µm**

	Value	Error	F	Cx	Cy	B2	K1	K2	K3	P1	P2
F	**2992.69**	0.088	1.00	0.06	-0.26	0.10	-0.22	0.27	-0.24	0.01	-0.08
Cx	**8.8713**	0.18		1.00	-0.26	0.32	0.02	-0.00	0.01	0.96	-0.24
Cy	**12.8877**	0.14			1.00	-0.01	-0.04	-0.01	0.00	-0.21	0.91
B2	**-0.562464**	0.034				1.00	0.00	0.01	-0.01	0.15	-0.00
K1	**-0.00275943**	9.1e-05					1.00	-0.96	0.91	0.02	-0.06
K2	**0.0136324**	0.00025						1.00	-0.98	-0.01	0.01
K3	**-0.0103758**	0.00021							1.00	0.02	-0.00
P1	**0.00122726**	1.7e-05								1.00	-0.22
P2	**0.00265353**	1.2e-05									1.00

Table 3. Calibration coefficients and correlation matrix.

Camera Calibration

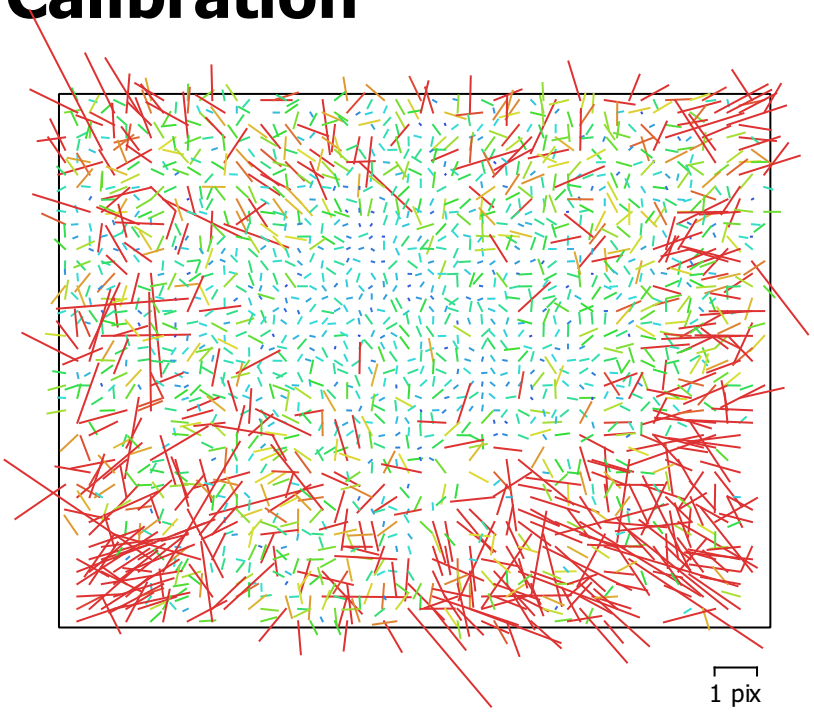


Fig. 4. Image residuals for Anafi (4mm).

Anafi (4mm)

6 images

Type	Resolution	Focal Length	Pixel Size
Frame	**4608 x 3456**	**4 mm**	**1.34 x 1.34 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	P1	P2
F	**2990.12**	0.56	1.00	0.27	-0.62	-0.28	0.28	-0.25	-0.03	0.10
Cx	**9.44161**	0.38		1.00	-0.22	-0.02	0.03	-0.04	0.85	0.01
Cy	**12.98**	0.35			1.00	-0.00	-0.03	0.04	0.02	0.42
K1	**-0.000497537**	0.0004				1.00	-0.96	0.91	-0.01	0.06
K2	**0.00789889**	0.0011					1.00	-0.98	0.02	-0.05
K3	**-0.0065996**	0.00093						1.00	-0.03	0.04
P1	**0.00130701**	4.2e-05							1.00	-0.00
P2	**0.00259583**	2.9e-05								1.00

Table 4. Calibration coefficients and correlation matrix.

Camera Locations

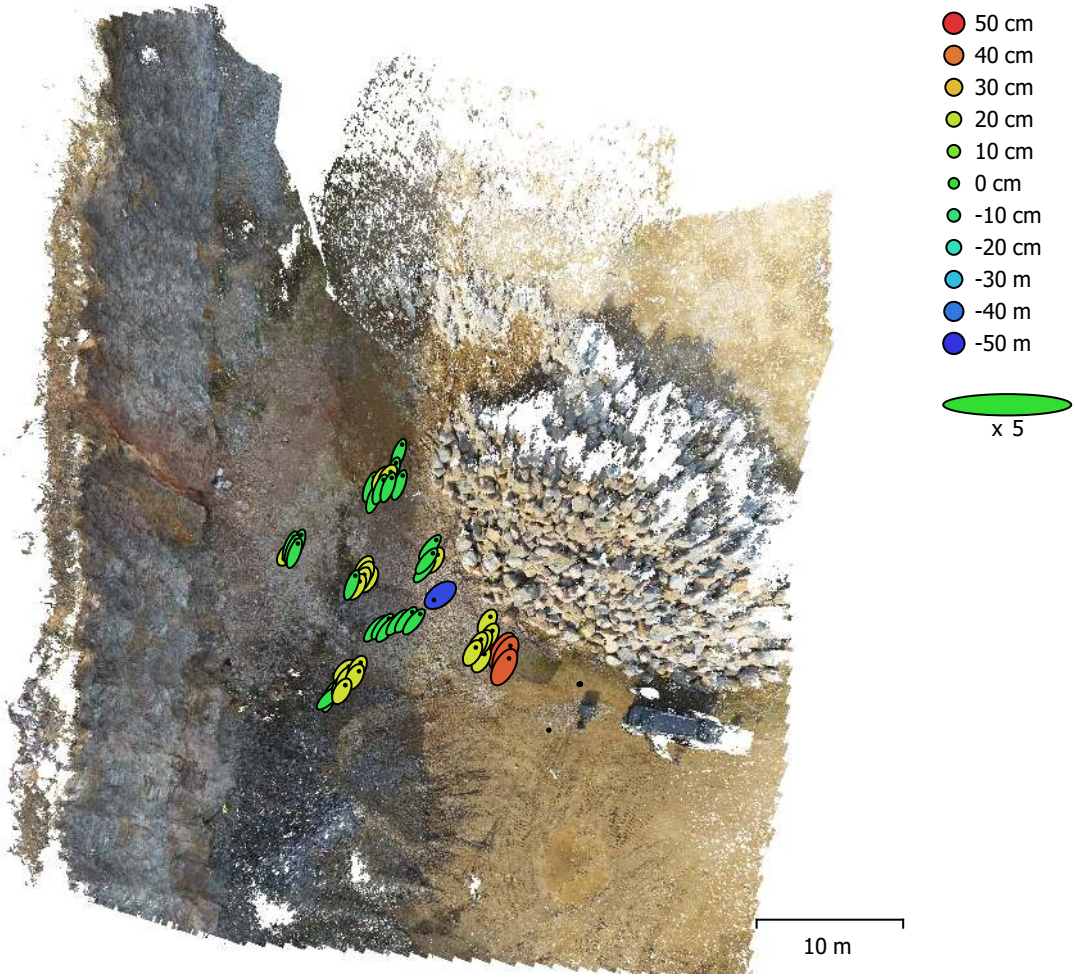


Fig. 5. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
1.76509	2.92222	43.0987	3.41393	43.2337

Table 5. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Ground Control Points

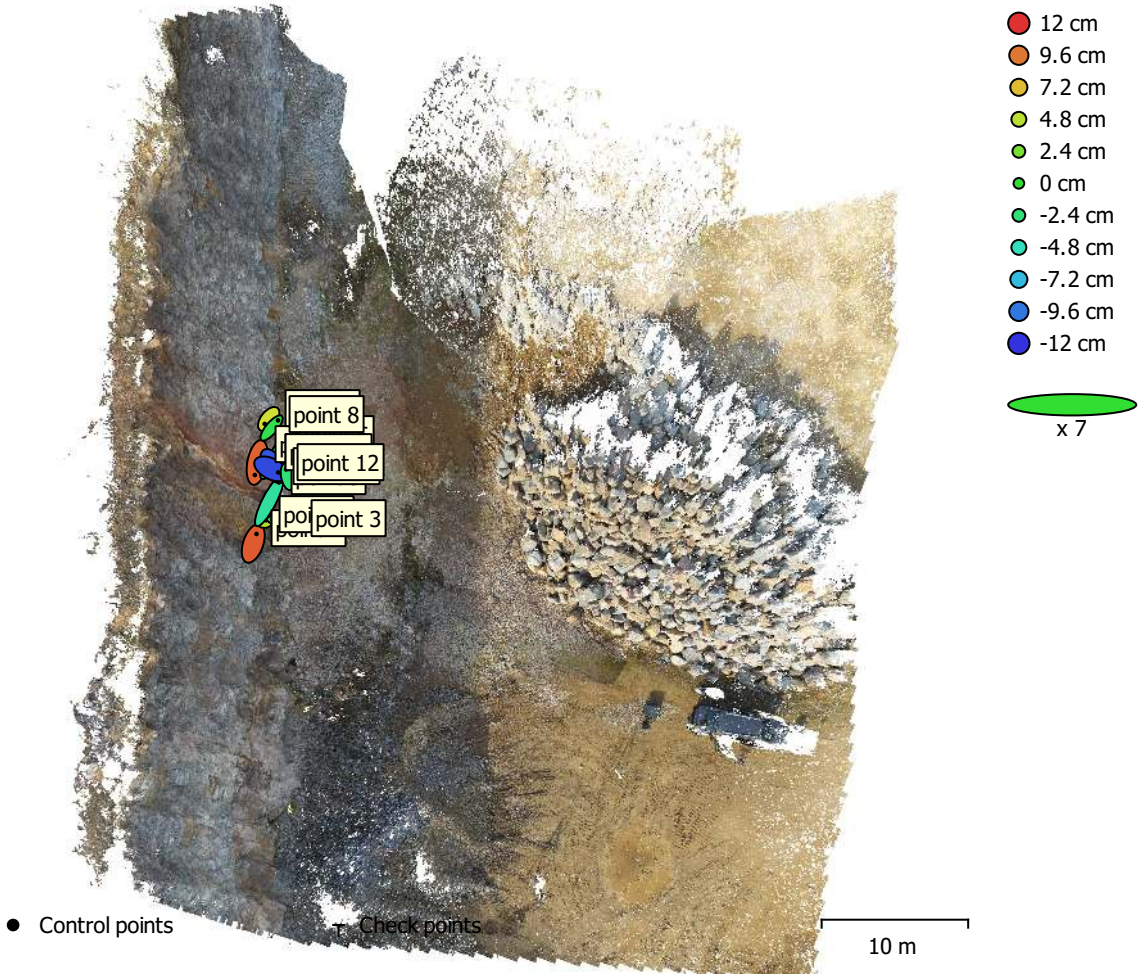


Fig. 6. GCP locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated GCP locations are marked with a dot or crossing.

Count	X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total (cm)
12	12.4503	17.2857	6.69627	21.3027	22.3304

Table 6. Control points RMSE.
X - Longitude, Y - Latitude, Z - Altitude.

Label	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Image (pix)
point 1	6.33297	18.4947	10.5669	22.2221	0.232 (101)
point 2	2.86277	0.585809	4.40858	5.28906	1.125 (18)
point 3	-27.2287	3.63068	-0.409702	27.4727	8.341 (5)
point 4	-4.72419	-24.427	10.3435	26.9441	6.941 (4)
point 5	-8.28902	-15.2631	1.13033	17.4054	2.201 (4)
point 6	-11.9251	-17.1877	-2.57175	21.077	2.965 (3)
point 7	-7.73296	-9.60561	5.46915	13.4899	1.878 (4)
point 8	12.6394	15.0278	-0.960379	19.6599	2.767 (4)
point 9	18.7793	36.4363	-3.76092	41.1632	5.548 (3)
point 10	8.25328	-14.6236	-10.7694	19.9486	8.614 (5)
point 11	13.7615	-7.05825	-11.1979	19.0943	8.003 (5)
point 12	-2.73354	13.9831	-2.36314	14.4424	6.129 (5)
Total	**12.4503**	**17.2857**	**6.69627**	**22.3304**	**3.183**

Table 7. Control points.

X - Longitude, Y - Latitude, Z - Altitude.

Digital Elevation Model

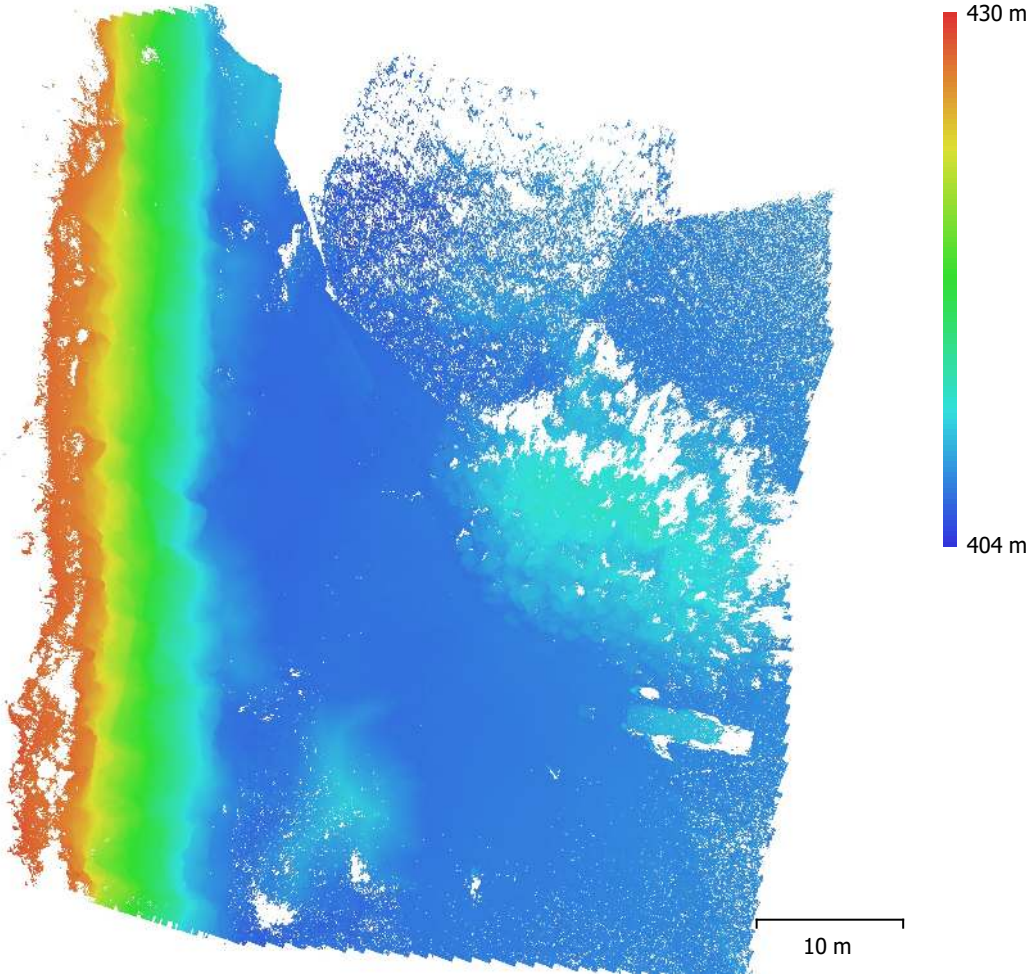


Fig. 7. Reconstructed digital elevation model.

Resolution:	unknown
Point density:	unknown

Processing Parameters

General	
Cameras	145
Aligned cameras	141
Markers	12
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Point Cloud	
Points	30,502 of 87,978
RMS reprojection error	0.158149 (0.645612 pix)
Max reprojection error	1.38695 (16.0408 pix)
Mean key point size	3.55571 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	9.61158
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Source
Key point limit	40,000
Tie point limit	4,000
Guided image matching	No
Adaptive camera model fitting	No
Matching time	59 seconds
Matching memory usage	454.54 MB
Alignment time	1 minutes 49 seconds
Alignment memory usage	61.03 MB
Optimization parameters	
Parameters	f, cx, cy, k1-k3, p1, p2
Adaptive camera model fitting	Yes
Optimization time	5 seconds
Software version	1.6.3.10732
Depth Maps	
Count	140
Depth maps generation parameters	
Quality	High
Filtering mode	Aggressive
Processing time	42 minutes 49 seconds
Software version	1.6.3.10732
Dense Point Cloud	
Points	35,347,780
Point colors	3 bands, uint8
Depth maps generation parameters	
Quality	High
Filtering mode	Aggressive
Processing time	42 minutes 49 seconds
Dense cloud generation parameters	
Processing time	29 minutes 33 seconds
Software version	1.6.3.10732
System	
Software name	Agisoft Metashape Professional

Software version	1.6.3 build 10732
OS	Windows 64 bit
RAM	127.68 GB
CPU	Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz
GPU(s)	NVIDIA GeForce GTX 1080 Ti NVIDIA GeForce GTX 1080 Ti

Appendix C

Code

C.1 Calculations of leverarms and boresights (Uis Dataset)

1 leverarm and boresight

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle
import cv2 as cv
import hylite
from hylite import io
from hylite.project.align import align_to_cloud_manual, align_to_cloud
```

[2]:
```
# open dict
with open("2-dict-path-root.pkl", 'rb') as f:
    dict_paths = pickle.load(f)

# loading camera positions of LWIR and lRGB
cams_lwir = []
cams_lrgb = []
indices = []

for i in dict_paths.keys():
    if i == 4 or i == 8:
        continue

    indices.append(i)

    camfile_lwir = f"7-cam_files/cam_file_lwir_{i}.txt"
    camfile_lrgb = f"7-cam_files/cam_file_lrgb_{i}.txt"

    cam_lwir = io.loadCameraTXT(camfile_lwir)
    cam_lrgb = io.loadCameraTXT(camfile_lrgb)

    cams_lwir.append(cam_lwir)
    cams_lrgb.append(cam_lrgb)
```

- "**boresight** (angular misalignment between the mounting axes of the IMU and onboard sensor)"
- "**lever-arm** (physical offset from the GNSS antenna to the onboard sensor)"

(Gautam et al., 2019, p. 25)

1.0.1 Lever arm for `lwir`

[4]:
```
for i in range(len(cams_lwir)):
    print(cams_lwir[i].pos - cams_lwir[0].pos)
```

```
[0. 0. 0.]
[5.917 2.2   1.235]
[-2.893 -2.873 -0.038]
[-160.192   89.116  -26.822]
[ 13.975 -26.413   2.812]
[-27.35  -17.546   6.939]
[0.384 0.747 0.112]
[-259.66   176.396  -23.44 ]
```

1.0.2 Lever arm for `rgb`

[5]:
```
for i in range(len(cams_lrgb)):
    print(cams_lrgb[i].pos - cams_lrgb[0].pos)
```

```
[0. 0. 0.]
[-0.614  0.519 -0.616]
[-0.548  0.413 -0.994]
[-0.253  0.401 -0.182]
[-0.361  0.537  0.245]
[-1.042  1.13  -0.619]
[-0.132  0.547 -0.356]
[-0.379  0.663 -0.425]
```

1.0.3 Lever arm between `lwir` and `lrgb`

[6]:
```
leverarms = []

for i in range(len(cams_lwir)):

    leverarm = cams_lwir[i].pos - cams_lrgb[i].pos

    leverarms.append(leverarm)

leverarms
```

[6]:
```
[array([-0.626,  0.261, -0.904]),
 array([5.905, 1.942, 0.947]),
 array([-2.971, -3.025,  0.052]),
 array([-160.565,   88.976,  -27.544]),
 array([ 13.71 , -26.689,   1.663]),
 array([-26.934, -18.415,   6.654]),
```

```
 array([-0.11 ,  0.461, -0.436]),
 array([-259.907,  175.994,  -23.919])]
```

?? **Should not these lever-ars be similar** ??

2 Boresight

[8]:
```
from scipy import spatial
```

[9]:
```
boresights = []

for i in range(len(cams_lwir)):

    alwir = cams_lwir[i].ori
    alrgb = cams_lrgb[i].ori

    Rlwir = spatial.transform.Rotation.from_euler('XYZ', -alwir, degrees=True).
 ↪as_matrix()
    Rlrgb = spatial.transform.Rotation.from_euler('XYZ', -alrgb, degrees=True).
 ↪as_matrix()

    Vlwir = Rlwir[2]  # or [:, 2]
    Vlrgb = Rlrgb[2]

    boresight = np.cos(np.dot(Vlwir, Vlrgb)) * np.cross(Vlwir, Vlrgb)
    boresights.append(boresight)

boresights
```

[9]:
```
[array([ 0.00325245, -0.00452947,  0.03242266]),
 array([-0.00385923, -0.00076063,  0.00442202]),
 array([-0.00117166, -0.0047706 ,  0.03802393]),
 array([-0.1680691 , -0.00522243, -0.01629946]),
 array([ 0.00848612, -0.00868461,  0.06671003]),
 array([-0.02795909, -0.00468419,  0.04492768]),
 array([ 0.00088996, -0.00209409,  0.01971254]),
 array([-0.08116191, -0.00350109, -0.00697786])]
```

?? **Should not these boresights be similar** ??

C.2 Nikon camera shooting script

1 gphoto workflow to take photos programmatically

[11]:
```
import gphoto2 as gp
import os
```

[12]:
```
cameras = gp.Camera.autodetect()
for n, (name, value) in enumerate(cameras):
    print('camera number', n+1)
    print('===============')
    print(name)
    print(value)
```

```
camera number 1
===============
Nikon Coolpix A (PTP mode)
usb:001,011
```

[13]:
```
camera = gp.Camera()
camera.init()
cam_summary = camera.get_summary()
print('Summary')
print('=======')
print(str(cam_summary))
```

```
Summary
=======
Manufacturer: Nikon Corporation
Model: COOLPIX A
  Version: V1.00
  Serial Number: 400047200000000000000000000000000
Vendor Extension ID: 0xa (1.0)
Vendor Extension Description: microsoft.com: 1.0

Capture Formats:
Display Formats: JPEG, Undefined Type, Association/Directory, DPOF, Script,
Apple Quicktime

Device Capabilities:
```

```
        File Download, File Deletion, File Upload
        No Image Capture, No Open Capture, Nikon Capture 3

Storage Devices Summary:
store_00010001:
        StorageDescription:
        VolumeLabel:             [Slot 1]
        Storage Type: Removable RAM (memory card)
        Filesystemtype: Digital Camera Layout (DCIM)
        Access Capability: Read Only with Object deletion
        Maximum Capability: 63831015424 (60874 MB)
        Free Space (Bytes): 3608936448 (3441 MB)
        Free Space (Images): 724

Device Property Summary:
Battery Level(0x5001):(read only) (type=0x2) Range [0 - 100, step 1] value: 35%
(35)
Image Size(0x5003):(readwrite) (type=0xffff) Enumeration [
        '4928x3264',
        '3696x2448',
        '2464x1632'
        ] value: '4928x3264'
Compression Setting(0x5004):(readwrite) (type=0x2) Enumeration [0,1,2,4,5,6,7]
value: JPEG Norm (1)
White Balance(0x5005):(readwrite) (type=0x4) Enumeration
[2,4,5,6,7,32784,32785,32787] value: Automatic (2)
F-Number(0x5007):(read only) (type=0x4) Enumeration [280,320,350,400,450,500,560
,630,710,800,900,1000,1100,1300,1400,1600,1800,2000,2200] value: f/4 (400)
Focal Length(0x5008):(read only) (type=0x6) Range [1800 - 1800, step 1] value:
18 mm (1800)
Focus Mode(0x500a):(read only) (type=0x4) Enumeration [1,32784,32787] value:
Manual Focus (1)
Exposure Metering Mode(0x500b):(read only) (type=0x4) Enumeration [2,3,4] value:
Multi-spot (3)
Flash Mode(0x500c):(readwrite) (type=0x4) Enumeration [2,4,32784] value: Auto
(32784)
Exposure Time(0x500d):(read only) (type=0x6) Enumeration [5,6,8,10,12,15,20,25,3
1,40,50,62,80,100,125,166,200,250,333,400,500,666,769,1000,1250,1666,2000,2500,3
333,4000,5000,6250,7692,10000,13000,16000,20000,25000,30000,40000,50000,60000,80
000,100000,130000,150000,200000,250000,300000] value: 0.0025 sec (250)
Exposure Program Mode(0x500e):(read only) (type=0x4) Enumeration
[1,2,3,4,32784,32792,32848,32849] value: Auto (32784)
Exposure Index (film speed ISO)(0x500f):(readwrite) (type=0x4) Enumeration [100,
125,160,200,250,320,400,500,640,800,1000,1250,1600,2000,2500,3200,4000,5000,6400
,8000,10000,12800,25600] value: ISO 100 (100)
Exposure Bias Compensation(0x5010):(readwrite) (type=0x3) Enumeration [-5000,-46
66,-4333,-4000,-3666,-3333,-3000,-2666,-2333,-2000,-1666,-1333,-1000,-666,-333,0
,333,666,1000,1333,1666,2000,2333,2666,3000,3333,3666,4000,4333,4666,5000]
```

```
 value: 2.0 stops (2000)
 Date & Time(0x5011):(readwrite) (type=0xffff) '20221024T165403'
 Still Capture Mode(0x5013):(readwrite) (type=0x4) Enumeration
 [1,2,32785,32788,32789,32792] value: Single Shot (1)
 Burst Number(0x5018):(readwrite) (type=0x4) Range [1 - 100, step 1] value: 1
 UseDeviceStageFlag(0xd303):(read only) (type=0x2) 1
 Property 0xd406:(readwrite) (type=0xffff) 'Windows/6.0.5330.0
 MTPClassDriver/6.0.5330.0'
 Property 0xd407:(read only) (type=0x6) 1
Camera Language(0xf018): error 2005 on query.
Release without SD card(0xf019): error 2005 on query.
Property 0xf01a: error 2005 on query.
Property 0xf01b: error 2005 on query.
Movie Quality(0xf01c): error 2005 on query.
Nikon Exposure Time(0xd100):(read only) (type=0x6) Enumeration [67536,67136,6678
6,66536,66336,66176,66036,65936,65856,65786,65736,65696,65661,65636,65616,65596,
65586,65576,65566,65561,65556,65551,65549,65546,65544,65542,65541,65540,65539,65
5385,65538,655376,655373,65537,851978,1048586,131073,1638410,196609,262145,32768
1,393217,524289,655361,851969,983041,1310721,1638401,1966081] value: 65576
Live View Prohibit Condition(0xd1a4):(read only) (type=0x6) 0
Live View Status(0xd1a2):(read only) (type=0x2) Range [0 - 1, step 1] value: No
(0)
```

[14]:
```
file = camera.capture(gp.GP_CAPTURE_IMAGE)
print(f"Camera file path: {file.folder}{file.name}")
```

```
Camera file path: /capt0000.jpg
```

[15]:
```
name = "yaay77799"
target = os.path.join('.', f"{name}.jpg")
print('Copying image to', target)
# Local file
lfile = camera.file_get(file.folder, file.name, gp.GP_FILE_TYPE_NORMAL)
lfile.save(target)
```

```
Copying image to ./yaay77799.jpg
```

[16]:
```
camera.exit()
```

C.3 Camera Calibration

1 Camera calibration of the RGB integrated TELOPS Camera

1.1 Field tasks:

1. Print chessboards
2. Get the TELOPS ready (thank's Erik)
3. Take multiple shots of the chessboard in different orientations

From OpenCV docs for better results 10 test patterns is preferred when trying to resolve distortion coefficients, we took **8**.

1.2 The calibration

Camera calibration is removing the distortions introduced by the camera.

The pinhole camera (simplest model) come with 2 major distortions : *radial* (mostly) and *tangential* (slightly).

1.3 The CODE

Original: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

[1]:
```
# impoting needed libraries
import numpy as np
import cv2 as cv
import glob
import hylite
from hylite import io
import matplotlib.pyplot as plt
```

1.3.1 FINDING CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS

For calibration we need 3D real world points **object points** and the corresponding 2D coordinates of these points in the image **image points**.

- First we need to know (X, Y, Z) values;
  - we assume the chessboard is kept stationary at XY plane ($Z = 0$); and the camera was moving
  - to find X and Y we pass the location of the points as $(0, 0), (1, 0), (2, 0)$... [the **scale** being the size of chessboard size] we could possibly work with square size.
  - so the **object points** we will be using are like this: (0,0,0), (1,0,0), (2,0,0)
- Finding the pattern in the chessboard (finding corners and adding them to **image points**)
  - `cv.findChessboardCorners()` outputs: *retval*: Boolean (return value, True if corners obtained), *corners*: the actual corners
  - Increase accuracy of corners using `cv.cornerSubPix()`

[2]:
```
# chessboaf parameters
chessboardSize = (10,7)
frameSize = (658,492)
```

[3]:
```
# termination criteria when increasing accuracy of found corners later on
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((chessboardSize[0] * chessboardSize[1], 3), np.float32)
objp[:,:2] = np.mgrid[0:chessboardSize[0],0:chessboardSize[1]].T.reshape(-1,2)
```

```
# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

images = glob.glob('1-imgCamCalib/2/*.jpg')

for image in images:

    img = cv.imread(image)
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # Find the chess board corners
    # _retval_: Boolean (return value, True if corners obtained), _corners_:␣
 ↪the actual corners
    ret, corners = cv.findChessboardCorners(gray, chessboardSize, None)

    # If found, add object points, image points (after refining them)
    if ret == True:

        objpoints.append(objp)
        # Increase accuracy of corners
        corners2 = cv.cornerSubPix(gray, corners, (11,11), (-1,-1), criteria)
        imgpoints.append(corners)

        # Draw and display the corners
        # cv.drawChessboardCorners(img, chessboardSize, corners2, ret)
        # cv.imshow('img', img)
        # cv.waitKey(700)
        # filename = f"chessCorners/{image[7:]}"
        # cv.imwrite(filename, img)

# cv.destroyAllWindows()
```

1.3.2 CALIBRATION

Now for calibration we use `cv.calibrateCamera()` which returns :

- camera matrix: camera intrinsic matrix $A = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$
- distortions coefficients: vector of distortion coefficients $(k1, k2, p1, p2[, k3[, k4....]])$
- rotation vector
- translation vector

[4]:
```
ret, cameraMatrix, dist, rvecs, tvecs = cv.calibrateCamera(objpoints,
                                                    imgpoints, frameSize, None,␣
 ↪None)
```

Optimizing camera matrix We can refine the camera matrix based on a free scaling parameter using `cv.getOptimalNewCameraMatrix()` having the parameter **alpha** that is free scaling parameter between `0` (when all the pixels in the undistorted image are valid) and `1` (when all the source image pixels are retained in the undistorted image).

Need to know more about alpha !

We will be using `0` for alpha.

[19]:
```
img = io.load('1-imgCamCalib/2/3.jpg')
h,  w = img.ydim(), img.xdim()

# input params : cameraMatrix, distCoeffs,
#                    imageSize, alpha[,newImgSize[,centerPrincipalPoint]])
newCameraMatrix, roi = cv.getOptimalNewCameraMatrix(cameraMatrix, dist, (w,h),␣
 ↪0, (w,h))

# fix the focal length
# newCameraMatrix[0,0] = newCameraMatrix[1,1] = 1.1e3
```

1.3.3 UNDISTORTION

[23]:
```
# Undistort with Remapping
mapx, mapy = cv.initUndistortRectifyMap(cameraMatrix, dist, None,␣
 ↪newCameraMatrix, (w,h), 5)
dst = cv.remap(np.transpose(img.data, (1,0,2)), mapx, mapy, cv.INTER_LINEAR)

imgr = hylite.HyImage(np.transpose(dst, (1,0,2)))
```

1.4 Reprojection Error

For estimating the accuracy of the found parameters we use the **re-projection error**. We transform the object points to image points and we calculate the absolute norm between what we got with our transformation and the corner finding algorithm.

Need to search more about how exactly this is done !

[21]:
```
# Reprojection Error
mean_error = 0

for i in range(len(objpoints)):
    imgpoints2, _ = cv.projectPoints(objpoints[i], rvecs[i], tvecs[i],␣
 ↪cameraMatrix, dist)
    error = cv.norm(imgpoints[i], imgpoints2, cv.NORM_L2)/len(imgpoints2)
    mean_error += error

print( "total error: {}".format(mean_error/len(objpoints)) )
```

```
total error: 0.07324389291486763
```

1.5 Saving the params for later use

[22]:
```
# saving

np.savez("1-camCalibParams", newCameraMatrix=newCameraMatrix,
         cameraMatrix=cameraMatrix, dist=dist, rvecs=rvecs, tvecs=tvecs,␣
 ↪mapx=mapx, mapy=mapy)
```

C.4 Arranging stitched bands

1 Organize stitched bands

[1]:
```
# Importing necessary modules
import glob
import os
import cv2 as cv
import matplotlib.pyplot as plt
from hylite import io
import numpy as np
```

[2]:
```
# Creating an empty list to store the roots
roots = []

# Adding the roots of the images to the list
for f in glob.glob('data/telops/**/*.jpg'):
    roots.append(f[:-4])

# Sorting the roots and limiting the number to 30
roots.sort()
roots = roots[:30]
```

[3]:
```
# Creating a list of roots in .ir.bmp format
rootsBMP = [f'{r}.ir.bmp' for r in roots]
```

[4]:
```
# Writing the names of the roots in a file
with open(r'roots_.txt', 'w') as f:
        f.write(' '.join(rootsBMP))
```

[5]:
```
# Loading the header of the first image for sanity check
img = io.load(f'{roots[0]}.radiance.hdr')
img.header
```

[5]:
```
{'file type': 'ENVI Standard',
 'path':
'data/telops/20090101_002054491_Scenario_1/20090101_002048062_1.radiance.hdr',
 'description': 'TEL-4415',
 'samples': '320',
```

```
 'lines': '256',
 'bands': '44',
 'acquisition time': '2008-12-31T23:20:48',
 'header offset': '657120',
 'data type': '4',
 'interleave': 'bip',
 'sensor type': 'Telops Hyper-Cam',
 'byte order': '0',
 'x start': '0',
 'y start': '0',
 'wavelength units': 'nm',
 'z plot titles': 'Wavenumber (cm-1), Radiance (W/m2 sr cm-1)',
 'band names': '1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44',
 'wavelength': array([ 879561.,  889334.,  899107.,  908880.,  918653.,
928426.,
         938199.,  947971.,  957744.,  967517.,  977290.,  987063.,
         996836., 1006610., 1016380., 1026150., 1035930., 1045700.,
        1055470., 1065250., 1075020., 1084790., 1094560., 1104340.,
        1114110., 1123880., 1133660., 1143430., 1153200., 1162980.,
        1172750., 1182520., 1192290., 1202070., 1211840., 1221610.,
        1231390., 1241160., 1250930., 1260700., 1270480., 1280250.,
        1290020., 1299800.])}
```

[56]:
```
# Creating a list of bands
bands = list(range(44))
# Dividing the bands into groups of 3
trio_bands = [bands[i:i+3] for i in range(0, len(bands), 3)]
# Removing the last band
trio_bands.pop()
trio_bands
```

[56]:
```
[[0, 1, 2],
 [3, 4, 5],
 [6, 7, 8],
 [9, 10, 11],
 [12, 13, 14],
 [15, 16, 17],
 [18, 19, 20],
 [21, 22, 23],
 [24, 25, 26],
 [27, 28, 29],
 [30, 31, 32],
 [33, 34, 35],
 [36, 37, 38],
 [39, 40, 41]]
```

[57]:
```
# Creating a directory for each group of bands
for c in trio_bands:
    p = f'data2/{c[0]:02}{c[1]:02}{c[2]:02}'
    if not os.path.exists(p): os.mkdir(p)
```

1.1 Exporting 3 bands at a time

[58]:
```
# Exporting the bands in groups of 3
for c in trio_bands:
    for f in roots:
        nm = f.split('/')[-1]
        img = io.load(f'{f}.radiance.hdr')
        img0 = img.export_bands(c)
        img1 = img0.data.transpose(1, 0, 2)[:, ::-1]
        imgnp = np.multiply(img1, 255).astype(np.uint8)
        cv.imwrite(f'data2/{c[0]:02}{c[1]:02}{c[2]:02}/{nm}.jpg', imgnp)
```

1.2 Executing the script

[ ]:
```
# Executing the script for each group of bands
for c in trio_bands:
    s = 'sift'
    b = f'{c[0]:02}{c[1]:02}{c[2]:02}'
    o = f'data3/stitched_{b}.jpg'
    !python stitching_detailed.py $(cat roots.txt) --features $s --output $o␣
 ↪--bands $b
```

2 Wrap affine

[62]:
```
# Applying the affine transform
imgjn1 = cv.imread('data/nikon/DSC_5421.JPG')
M1 = np.fromfile('temp/imgb1-affine-transform-nikon.txt').reshape((3,3))
dims = np.array(imgjn1.shape[:2][::-1])
```

[64]:
```
dims
```

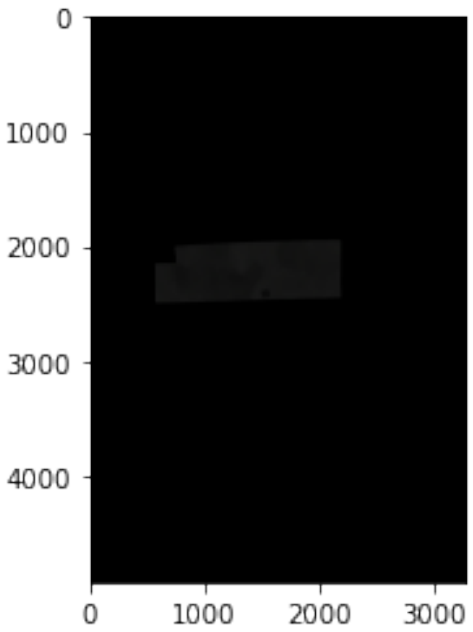
[64]:
```
array([3264, 4928])
```

[77]:
```
for c in trio_bands:
    p = f'data-mosaics3-wrap/{c[0]:02}{c[1]:02}{c[2]:02}'
    if not os.path.exists(p): os.mkdir(p)
```

[85]:
```
for f in glob.glob('data3/*.jpg'):
    img = cv.imread(f)
    dst = cv.warpAffine(img, M1[:2], dims)
    cv.imwrite(f'data-mosaics3-wrap/{f[6:-4]}.jpg', dst)
```

[74]:
```
plt.imshow(dst)
```

[74]:
```
<matplotlib.image.AxesImage at 0x7fc61c7666d0>
```



C.5 Automate and plot different combinations of params (SuperGlue)

1 Automate params checking & plotting different combinationa

1.1 Automate input different params to the script

[1]:
```
import itertools
```

[2]:
```
# 12-dec
dict_params = {
    'keypoint_threshold': [0.001, 0.01],
    'nms_radius': [2, 6],
    'sinkhorn_iterations': [10, 30],
    'match_threshold': [0.1, 0.5]
}
```

[7]:
```
# 12-dec-01
dict_params = {
    'keypoint_threshold': [0.004, 0.006],
    'nms_radius': [3, 5],
    'sinkhorn_iterations': [19, 21],
    'match_threshold': [0.19, 0.21]
}
```

[8]:
```
keys, values = zip(*dict_params.items())
permutations_dicts = [dict(zip(keys, v)) for v in itertools.product(*values)]
```

[9]:
```
len(permutations_dicts)
```

[9]:
```
16
```

[10]:
```
permutations_dicts[0]
```

[10]:
```
{'keypoint_threshold': 0.004,
 'nms_radius': 3,
 'sinkhorn_iterations': 19,
 'match_threshold': 0.19}
```

[ ]:
```
for i in range(len(permutations_dicts)):
    !./match_pairs.py --input_dir img \
```

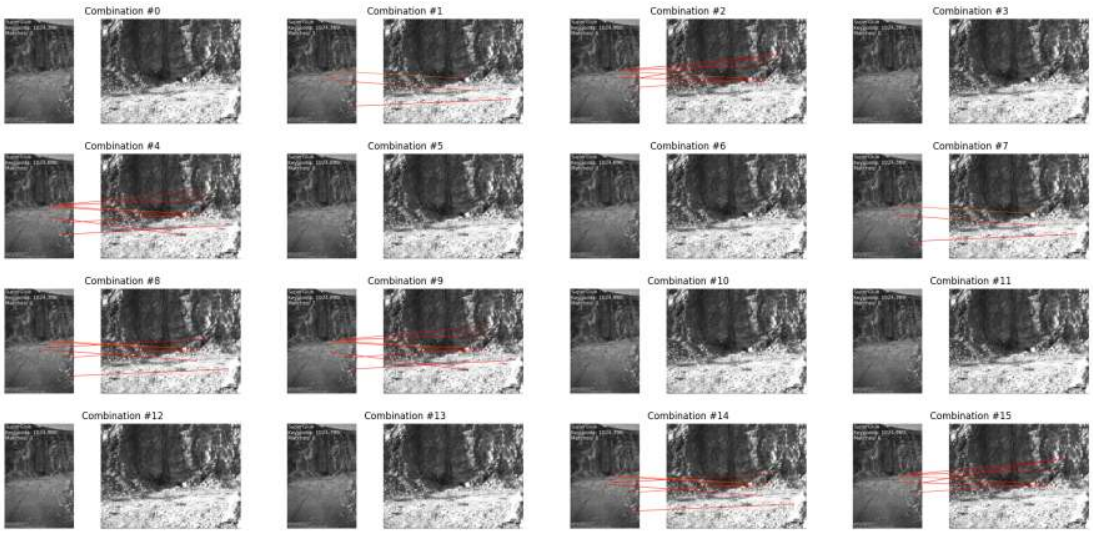
```
    --input_pairs pairs.txt \
    --resize -1 \
    --keypoint_threshold '{permutations_dicts[i]['keypoint_threshold']}'\
    --nms_radius '{permutations_dicts[i]['nms_radius']}'\
    --sinkhorn_iterations '{permutations_dicts[i]['sinkhorn_iterations']}'\
    --match_threshold '{permutations_dicts[i]['match_threshold']}'\
    --viz \
    --output result-12-dec-01/comb{i}
```

1.2 Plotting the tested combinations

[1]:
```
import matplotlib.pyplot as plt
import cv2 as cv
import glob
```

[24]:
```
imgs = []
for i in glob.glob('result2/**/*.png'):
    imgs.append(cv.imread(i))
```

[27]:
```
fig, axs = plt.subplots(4, 4, figsize=(25, 12))
axs = axs.flatten()
i = 0
for img, ax in zip(imgs, axs):
    ax.imshow(img)
    ax.set_title(f'Combination #{i}')
    ax.set_axis_off()
    i+=1
# fig.tight_layout(pad=-1)
fig.show()
```



C.6 Find the affine transform using napari

1 affine-transform napari

[1]:
```
import glob
import numpy as np
import matplotlib.pyplot as plt
import napari
from hylite import io
import cv2 as cv
import time
```

[3]:
```
def matrix_rc2xy(affine_matrix):
    swapped_cols = affine_matrix[:, [1, 0, 2]]
    swapped_rows = swapped_cols[[1, 0, 2], :]
    return swapped_rows
```

[4]:
```
roots = []

for f in glob.glob('data/telops/**/*.jpg'):
    roots.append(f[:-4])

roots.sort()

nikon = ['0-missing??']
for f in glob.glob('data/nikon/*'):
    nikon.append(f)

nikon.sort()
```

[5]:
```
imgb1 = cv.imread('stitched-bmp.jpg')
imgjn1 = cv.imread('data/nikon/DSC_5421.JPG')
imgjn1 = cv.cvtColor(imgjn1, cv.COLOR_RGB2BGR)
imgj0 = cv.imread('stitched-jpg10.jpg')
```

1.1 Stitched LWIR -2- single Nikon image

[6]:
```
viewer = napari.Viewer()
```

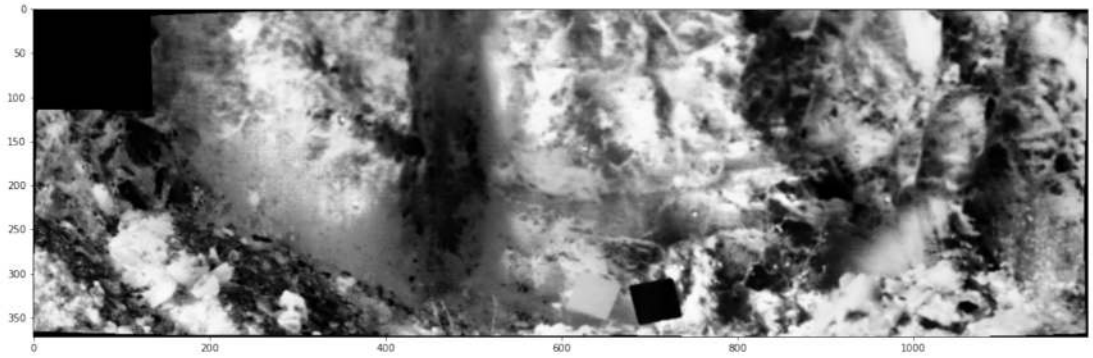
[7]:
```
l0 = viewer.add_image(imgjn1, colormap='bop blue', blending='translucent')
l1 = viewer.add_image(imgb1, colormap='bop purple', blending='translucent')
```

[8]:
```
qtwidget, widget = viewer.window.add_plugin_dock_widget(
        'affinder', 'Start affinder'
        )
```

[9]:
```
widget.reference.bind(l0)
widget.moving.bind(l1)
widget()
```

[10]:
```
plt.figure(figsize=(19, 8))
plt.imshow(imgb1)
```

[10]:
```
<matplotlib.image.AxesImage at 0x7efe3d38e8b0>
```



DO YOUR POINTS SELECTIONS (FIRST 3 for each and THEN 1-1 ...)

[11]:
```
imgb1_pts = viewer.layers[f"{f'{imgb1=}'.split('=')[0]}_pts"].data
imgb1_ref_pts = viewer.layers[f"{f'{imgjn1=}'.split('=')[0]}_pts"].data
```

[12]:
```
M1 = matrix_rc2xy(np.asarray(l1.affine))

# RUN ONLY ONCE !
# M1.tofile(f"{f'{imgb1=}'.split('=')[0]}-affine-transform-nikon.txt")
M1
```

[12]:
```
array([[ 1.34200718e+00,  3.80837209e-02,  5.50951051e+02],
       [-4.01079481e-02,  1.35677411e+00,  1.99118341e+03],
       [ 0.00000000e+00,  0.00000000e+00,  1.00000000e+00]])
```

[13]:
```
viewer.close()
```

[14]:
```
# Apply the affine transform matrix to the imgb
dims = np.array(imgjn1.shape[:2][::-1])
dst = cv.warpAffine(imgb1, M1[:2], dims)

fig, ax = plt.subplots(figsize = dims*.02)
ax.imshow(imgjn1)
plt.imshow(dst, alpha=.3)
plt.show()
```

C.7 Stitching

```
"""
Stitching sample (advanced)
===========================

Show how to use Stitcher API from python.
Credits: https://github.com/opencv/opencv/blob/master/samples/python/stitching_detailed.py
"""

# Python 2/3 compatibility
from __future__ import print_function

import argparse
from collections import OrderedDict

import cv2 as cv
import numpy as np

EXPOS_COMP_CHOICES = OrderedDict()
EXPOS_COMP_CHOICES['gain_blocks'] = cv.detail.ExposureCompensator_GAIN_BLOCKS
EXPOS_COMP_CHOICES['gain'] = cv.detail.ExposureCompensator_GAIN
EXPOS_COMP_CHOICES['channel'] = cv.detail.ExposureCompensator_CHANNELS
EXPOS_COMP_CHOICES['channel_blocks'] = cv.detail.ExposureCompensator_CHANNELS_BLOCKS
EXPOS_COMP_CHOICES['no'] = cv.detail.ExposureCompensator_NO

BA_COST_CHOICES = OrderedDict()
BA_COST_CHOICES['ray'] = cv.detail_BundleAdjusterRay
BA_COST_CHOICES['reproj'] = cv.detail_BundleAdjusterReproj
BA_COST_CHOICES['affine'] = cv.detail_BundleAdjusterAffinePartial
BA_COST_CHOICES['no'] = cv.detail_NoBundleAdjuster

FEATURES_FIND_CHOICES = OrderedDict()
try:
    cv.xfeatures2d_SURF.create()  # check if the function can be called
    FEATURES_FIND_CHOICES['surf'] = cv.xfeatures2d_SURF.create
except (AttributeError, cv.error) as e:
    print("SURF not available")
# if SURF not available, ORB is default
FEATURES_FIND_CHOICES['orb'] = cv.ORB.create
try:
    FEATURES_FIND_CHOICES['sift'] = cv.SIFT_create
except AttributeError:
    print("SIFT not available")
try:
    FEATURES_FIND_CHOICES['brisk'] = cv.BRISK_create
except AttributeError:
    print("BRISK not available")
try:
    FEATURES_FIND_CHOICES['akaze'] = cv.AKAZE_create
except AttributeError:
    print("AKAZE not available")

SEAM_FIND_CHOICES = OrderedDict()
SEAM_FIND_CHOICES['dp_color'] = cv.detail_DpSeamFinder('COLOR')
SEAM_FIND_CHOICES['dp_colorgrad'] = cv.detail_DpSeamFinder('COLOR_GRAD')
SEAM_FIND_CHOICES['voronoi'] = cv.detail.SeamFinder_createDefault(
    cv.detail.SeamFinder_VORONOI_SEAM)
SEAM_FIND_CHOICES['no'] = cv.detail.SeamFinder_createDefault(
    cv.detail.SeamFinder_NO)

ESTIMATOR_CHOICES = OrderedDict()
ESTIMATOR_CHOICES['homography'] = cv.detail_HomographyBasedEstimator
ESTIMATOR_CHOICES['affine'] = cv.detail_AffineBasedEstimator

WARP_CHOICES = (
    'spherical',
    'plane',
    'affine',
    'cylindrical',
    'fisheye',
    'stereographic',
    'compressedPlaneA2B1',
    'compressedPlaneA1.5B1',
    'compressedPlanePortraitA2B1',
    'compressedPlanePortraitA1.5B1',
    'paniniA2B1',
    'paniniA1.5B1',
    'paniniPortraitA2B1',
    'paniniPortraitA1.5B1',
    'mercator',
    'transverseMercator',
)

WAVE_CORRECT_CHOICES = OrderedDict()
WAVE_CORRECT_CHOICES['horiz'] = cv.detail.WAVE_CORRECT_HORIZ
WAVE_CORRECT_CHOICES['no'] = None
WAVE_CORRECT_CHOICES['vert'] = cv.detail.WAVE_CORRECT_VERT

BLEND_CHOICES = ('multiband', 'feather', 'no',)

parser = argparse.ArgumentParser(
    prog="stitching_detailed.py", description="Rotation model images stitcher"
)
parser.add_argument(
    'img_names', nargs='+',
    help="Files to stitch", type=str
)
```

```
parser.add_argument(
    '--try_cuda',
    action='store',
    default=False,
    help="Try to use CUDA. The default value is no. All default values are for CPU mode.",
    type=bool, dest='try_cuda'
)
parser.add_argument(
    '--work_megapix', action='store', default=0.6,
    help="Resolution for image registration step. The default is 0.6 Mpx",
    type=float, dest='work_megapix'
)
parser.add_argument(
    '--features', action='store', default=list(FEATURES_FIND_CHOICES.keys())[0],
    help="Type of features used for images matching. The default is '%s'." % list(
        FEATURES_FIND_CHOICES.keys())[0],
    choices=FEATURES_FIND_CHOICES.keys(),
    type=str, dest='features'
)
parser.add_argument(
    '--matcher', action='store', default='homography',
    help="Matcher used for pairwise image matching. The default is 'homography'.",
    choices=('homography', 'affine'),
    type=str, dest='matcher'
)
parser.add_argument(
    '--estimator', action='store', default=list(ESTIMATOR_CHOICES.keys())[0],
    help="Type of estimator used for transformation estimation. The default is '%s'." % list(
        ESTIMATOR_CHOICES.keys())[0],
    choices=ESTIMATOR_CHOICES.keys(),
    type=str, dest='estimator'
)
parser.add_argument(
    '--match_conf', action='store',
    help="Confidence for feature matching step. The default is 0.3 for ORB and 0.65 for other feature types.",
    type=float, dest='match_conf'
)
parser.add_argument(
    '--conf_thresh', action='store', default=1.0,
    help="Threshold for two images are from the same panorama confidence.The default is 1.0.",
    type=float, dest='conf_thresh'
)
parser.add_argument(
    '--ba', action='store', default=list(BA_COST_CHOICES.keys())[0],
    help="Bundle adjustment cost function. The default is '%s'." % list(
        BA_COST_CHOICES.keys())[0],
    choices=BA_COST_CHOICES.keys(),
    type=str, dest='ba'
)
parser.add_argument(
    '--ba_refine_mask', action='store', default='xxxxx',
    help="Set refinement mask for bundle adjustment. It looks like 'x_xxx', "
         "where 'x' means refine respective parameter and '_' means don't refine, "
         "and has the following format:<fx><skew><ppx><aspect><ppy>. "
         "The default mask is 'xxxxx'. "
         "If bundle adjustment doesn't support estimation of selected parameter then "
         "the respective flag is ignored.",
    type=str, dest='ba_refine_mask'
)
parser.add_argument(
    '--wave_correct', action='store', default=list(WAVE_CORRECT_CHOICES.keys())[0],
    help="Perform wave effect correction. The default is '%s'" % list(
        WAVE_CORRECT_CHOICES.keys())[0],
    choices=WAVE_CORRECT_CHOICES.keys(),
    type=str, dest='wave_correct'
)
parser.add_argument(
    '--save_graph', action='store', default=None,
    help="Save matches graph represented in DOT language to <file_name> file.",
    type=str, dest='save_graph'
)
parser.add_argument(
    '--warp', action='store', default=WARP_CHOICES[0],
    help="Warp surface type. The default is '%s'." % WARP_CHOICES[0],
    choices=WARP_CHOICES,
    type=str, dest='warp'
)
parser.add_argument(
    '--seam_megapix', action='store', default=0.1,
    help="Resolution for seam estimation step. The default is 0.1 Mpx.",
    type=float, dest='seam_megapix'
)
parser.add_argument(
    '--seam', action='store', default=list(SEAM_FIND_CHOICES.keys())[0],
    help="Seam estimation method. The default is '%s'." % list(
        SEAM_FIND_CHOICES.keys())[0],
    choices=SEAM_FIND_CHOICES.keys(),
    type=str, dest='seam'
)
parser.add_argument(
    '--compose_megapix', action='store', default=-1,
    help="Resolution for compositing step. Use -1 for original resolution. The default is -1",
    type=float, dest='compose_megapix'
)
parser.add_argument(
    '--expos_comp', action='store', default=list(EXPOS_COMP_CHOICES.keys())[0],
    help="Exposure compensation method. The default is '%s'." % list(
        EXPOS_COMP_CHOICES.keys())[0],
    choices=EXPOS_COMP_CHOICES.keys(),
```

```
    type=str, dest='expos_comp'
)
parser.add_argument(
    '--expos_comp_nr_feeds', action='store', default=1,
    help="Number of exposure compensation feed.",
    type=np.int32, dest='expos_comp_nr_feeds'
)
parser.add_argument(
    '--expos_comp_nr_filtering', action='store', default=2,
    help="Number of filtering iterations of the exposure compensation gains.",
    type=float, dest='expos_comp_nr_filtering'
)
parser.add_argument(
    '--expos_comp_block_size', action='store', default=32,
    help="BLock size in pixels used by the exposure compensator. The default is 32.",
    type=np.int32, dest='expos_comp_block_size'
)
parser.add_argument(
    '--blend', action='store', default=BLEND_CHOICES[0],
    help="Blending method. The default is '%s'." % BLEND_CHOICES[0],
    choices=BLEND_CHOICES,
    type=str, dest='blend'
)
parser.add_argument(
    '--blend_strength', action='store', default=5,
    help="Blending strength from [0,100] range. The default is 5",
    type=np.int32, dest='blend_strength'
)
parser.add_argument(
    '--output', action='store', default='result.jpg',
    help="The default is 'result.jpg'",
    type=str, dest='output'
)
parser.add_argument(
    '--timelapse', action='store', default=None,
    help="Output warped images separately as frames of a time lapse movie, "
         "with 'fixed_' prepended to input file names.",
    type=str, dest='timelapse'
)
parser.add_argument(
    '--rangewidth', action='store', default=-1,
    help="uses range_width to limit number of images to match with.",
    type=int, dest='rangewidth'
)

__doc__ += '\n' + parser.format_help()


def get_matcher(args):
    try_cuda = args.try_cuda
    matcher_type = args.matcher
    if args.match_conf is None:
        if args.features == 'orb':
            match_conf = 0.3
        else:
            match_conf = 0.65
    else:
        match_conf = args.match_conf
    range_width = args.rangewidth
    if matcher_type == "affine":
        matcher = cv.detail_AffineBestOf2NearestMatcher(
            False, try_cuda, match_conf)
    elif range_width == -1:
        matcher = cv.detail_BestOf2NearestMatcher(try_cuda, match_conf)
    else:
        matcher = cv.detail_BestOf2NearestRangeMatcher(
            range_width, try_cuda, match_conf)
    return matcher


def get_compensator(args):
    expos_comp_type = EXPOS_COMP_CHOICES[args.expos_comp]
    expos_comp_nr_feeds = args.expos_comp_nr_feeds
    expos_comp_block_size = args.expos_comp_block_size
    # expos_comp_nr_filtering = args.expos_comp_nr_filtering
    if expos_comp_type == cv.detail.ExposureCompensator_CHANNELS:
        compensator = cv.detail_ChannelsCompensator(expos_comp_nr_feeds)
        # compensator.setNrGainsFilteringIterations(expos_comp_nr_filtering)
    elif expos_comp_type == cv.detail.ExposureCompensator_CHANNELS_BLOCKS:
        compensator = cv.detail_BlocksChannelsCompensator(
            expos_comp_block_size, expos_comp_block_size,
            expos_comp_nr_feeds
        )
        # compensator.setNrGainsFilteringIterations(expos_comp_nr_filtering)
    else:
        compensator = cv.detail.ExposureCompensator_createDefault(
            expos_comp_type)
    return compensator


def main():
    args = parser.parse_args()
    img_names = args.img_names
    print(img_names)
    work_megapix = args.work_megapix
    seam_megapix = args.seam_megapix
    compose_megapix = args.compose_megapix
    conf_thresh = args.conf_thresh
    ba_refine_mask = args.ba_refine_mask
```

```
    wave_correct = WAVE_CORRECT_CHOICES[args.wave_correct]
    if args.save_graph is None:
        save_graph = False
    else:
        save_graph = True
    warp_type = args.warp
    blend_type = args.blend
    blend_strength = args.blend_strength
    result_name = args.output
    if args.timelapse is not None:
        timelapse = True
        if args.timelapse == "as_is":
            timelapse_type = cv.detail.Timelapser_AS_IS
        elif args.timelapse == "crop":
            timelapse_type = cv.detail.Timelapser_CROP
        else:
            print("Bad timelapse method")
            exit()
    else:
        timelapse = False
    finder = FEATURES_FIND_CHOICES[args.features]()
    seam_work_aspect = 1
    full_img_sizes = []
    features = []
    images = []
    is_work_scale_set = False
    is_seam_scale_set = False
    is_compose_scale_set = False
    for name in img_names:
        full_img = cv.imread(cv.samples.findFile(name))
        if full_img is None:
            print("Cannot read image ", name)
            exit()
        full_img_sizes.append((full_img.shape[1], full_img.shape[0]))
        if work_megapix < 0:
            img = full_img
            work_scale = 1
            is_work_scale_set = True
        else:
            if is_work_scale_set is False:
                work_scale = min(1.0, np.sqrt(
                    work_megapix * 1e6 / (full_img.shape[0] * full_img.shape[1])))
                is_work_scale_set = True
            img = cv.resize(src=full_img, dsize=None, fx=work_scale,
                            fy=work_scale, interpolation=cv.INTER_LINEAR_EXACT)
        if is_seam_scale_set is False:
            if seam_megapix > 0:
                seam_scale = min(1.0, np.sqrt(
                    seam_megapix * 1e6 / (full_img.shape[0] * full_img.shape[1])))
            else:
                seam_scale = 1.0
            seam_work_aspect = seam_scale / work_scale
            is_seam_scale_set = True
        img_feat = cv.detail.computeImageFeatures2(finder, img)
        features.append(img_feat)
        img = cv.resize(src=full_img, dsize=None, fx=seam_scale,
                        fy=seam_scale, interpolation=cv.INTER_LINEAR_EXACT)
        images.append(img)

    matcher = get_matcher(args)
    p = matcher.apply2(features)
    matcher.collectGarbage()

    if save_graph:
        with open(args.save_graph, 'w') as fh:
            fh.write(cv.detail.matchesGraphAsString(img_names, p, conf_thresh))

    indices = cv.detail.leaveBiggestComponent(features, p, conf_thresh)
    img_subset = []
    img_names_subset = []
    full_img_sizes_subset = []
    for i in range(len(indices)):
        img_names_subset.append(img_names[indices[i]])
        img_subset.append(images[indices[i]])
        full_img_sizes_subset.append(full_img_sizes[indices[i]])
    images = img_subset
    img_names = img_names_subset
    full_img_sizes = full_img_sizes_subset
    num_images = len(img_names)
    print('image 0 shape is :', images[0].shape)
    if num_images < 2:
        print("Need more images")
        exit()

    estimator = ESTIMATOR_CHOICES[args.estimator]()
    b, cameras = estimator.apply(features, p, None)
    if not b:
        print("Homography estimation failed.")
        exit()
    for cam in cameras:
        cam.R = cam.R.astype(np.float32)

    adjuster = BA_COST_CHOICES[args.ba]()
    adjuster.setConfThresh(conf_thresh)
    refine_mask = np.zeros((3, 3), np.uint8)
    if ba_refine_mask[0] == 'x':
        refine_mask[0, 0] = 1
    if ba_refine_mask[1] == 'x':
        refine_mask[0, 1] = 1
```

```
    if ba_refine_mask[2] == 'x':
        refine_mask[0, 2] = 1
    if ba_refine_mask[3] == 'x':
        refine_mask[1, 1] = 1
    if ba_refine_mask[4] == 'x':
        refine_mask[1, 2] = 1
    adjuster.setRefinementMask(refine_mask)
    b, cameras = adjuster.apply(features, p, cameras)
    if not b:
        print("Camera parameters adjusting failed.")
        exit()
    focals = []
    for cam in cameras:
        focals.append(cam.focal)
    focals.sort()
    if len(focals) % 2 == 1:
        warped_image_scale = focals[len(focals) // 2]
    else:
        warped_image_scale = (
            focals[len(focals) // 2] + focals[len(focals) // 2 - 1]) / 2
    if wave_correct is not None:
        rmats = []
        for cam in cameras:
            rmats.append(np.copy(cam.R))
        rmats = cv.detail.waveCorrect(rmats, wave_correct)
        for idx, cam in enumerate(cameras):
            cam.R = rmats[idx]
    corners = []
    masks_warped = []
    images_warped = []
    sizes = []
    masks = []
    for i in range(0, num_images):
        um = cv.UMat(
            255 * np.ones((images[i].shape[0], images[i].shape[1]), np.uint8))
        masks.append(um)

    # warper could be nullptr?
    warper = cv.PyRotationWarper(
        warp_type, warped_image_scale * seam_work_aspect)
    for idx in range(0, num_images):
        K = cameras[idx].K().astype(np.float32)
        swa = seam_work_aspect
        K[0, 0] *= swa
        K[0, 2] *= swa
        K[1, 1] *= swa
        K[1, 2] *= swa
        corner, image_wp = warper.warp(
            images[idx], K, cameras[idx].R, cv.INTER_LINEAR, cv.BORDER_REFLECT)
        corners.append(corner)
        sizes.append((image_wp.shape[1], image_wp.shape[0]))
        images_warped.append(image_wp)
        p, mask_wp = warper.warp(
            masks[idx], K, cameras[idx].R, cv.INTER_NEAREST, cv.BORDER_CONSTANT)
        masks_warped.append(mask_wp.get())

    images_warped_f = []
    for img in images_warped:
        imgf = img.astype(np.float32)
        images_warped_f.append(imgf)

    compensator = get_compensator(args)
    compensator.feed(corners=corners, images=images_warped, masks=masks_warped)

    seam_finder = SEAM_FIND_CHOICES[args.seam]
    masks_warped = seam_finder.find(images_warped_f, corners, masks_warped)
    compose_scale = 1
    corners = []
    sizes = []
    blender = None
    timelapser = None
    # https://github.com/opencv/opencv/blob/4.x/samples/cpp/stitching_detailed.cpp#L725 ?
    for idx, name in enumerate(img_names):
        # # this is me
        # name = name.split('/')[-1][:-7]
        # print(name)
        # # full_img = cv.imread(f'../itk/img3/{name}-b012.jpg')
        # full_img = cv.imread(f'data2/012/{nm}.jpg')

        full_img = cv.imread(name)

        if not is_compose_scale_set:
            if compose_megapix > 0:
                compose_scale = min(1.0, np.sqrt(
                    compose_megapix * 1e6 / (full_img.shape[0] * full_img.shape[1])))
            is_compose_scale_set = True
            compose_work_aspect = compose_scale / work_scale
            warped_image_scale *= compose_work_aspect
            warper = cv.PyRotationWarper(warp_type, warped_image_scale)
            for i in range(0, len(img_names)):
                cameras[i].focal *= compose_work_aspect
                cameras[i].ppx *= compose_work_aspect
                cameras[i].ppy *= compose_work_aspect
                sz = (int(round(full_img_sizes[i][0] * compose_scale)),
                      int(round(full_img_sizes[i][1] * compose_scale)))
                K = cameras[i].K().astype(np.float32)
                roi = warper.warpRoi(sz, K, cameras[i].R)
                corners.append(roi[0:2])
                sizes.append(roi[2:4])
```

```
493         if abs(compose_scale - 1) > 1e-1:
494             img = cv.resize(src=full_img, dsize=None, fx=compose_scale, fy=compose_scale,
495                             interpolation=cv.INTER_LINEAR_EXACT)
496         else:
497             img = full_img
498         _img_size = (img.shape[1], img.shape[0])
499         K = cameras[idx].K().astype(np.float32)
500         corner, image_warped = warper.warp(
501             img, K, cameras[idx].R, cv.INTER_LINEAR, cv.BORDER_REFLECT)
502         mask = 255 * np.ones((img.shape[0], img.shape[1]), np.uint8)
503         p, mask_warped = warper.warp(
504             mask, K, cameras[idx].R, cv.INTER_NEAREST, cv.BORDER_CONSTANT)
505         compensator.apply(idx, corners[idx], image_warped, mask_warped)
506         image_warped_s = image_warped.astype(np.int16)
507         dilated_mask = cv.dilate(masks_warped[idx], None)
508         seam_mask = cv.resize(
509             dilated_mask, (mask_warped.shape[1], mask_warped.shape[0]), 0, 0, cv.INTER_LINEAR_EXACT)
510         mask_warped = cv.bitwise_and(seam_mask, mask_warped)
511         if blender is None and not timelapse:
512             blender = cv.detail.Blender_createDefault(cv.detail.Blender_NO)
513             dst_sz = cv.detail.resultRoi(corners=corners, sizes=sizes)
514             blend_width = np.sqrt(dst_sz[2] * dst_sz[3]) * blend_strength / 100
515             if blend_width < 1:
516                 blender = cv.detail.Blender_createDefault(cv.detail.Blender_NO)
517             elif blend_type == "multiband":
518                 blender = cv.detail_MultiBandBlender()
519                 blender.setNumBands(
520                     (np.log(blend_width) / np.log(2.) - 1.).astype(np.int32))
521             elif blend_type == "feather":
522                 blender = cv.detail_FeatherBlender()
523                 blender.setSharpness(1. / blend_width)
524             blender.prepare(dst_sz)
525         elif timelapser is None and timelapse:
526             timelapser = cv.detail.Timelapser_createDefault(timelapse_type)
527             timelapser.initialize(corners, sizes)
528         if timelapse:
529             ma_tones = np.ones(
530                 (image_warped_s.shape[0], image_warped_s.shape[1]), np.uint8)
531             timelapser.process(image_warped_s, ma_tones, corners[idx])
532             pos_s = img_names[idx].rfind("/")
533             if pos_s == -1:
534                 fixed_file_name = "fixed_" + img_names[idx]
535             else:
536                 fixed_file_name = img_names[idx][:pos_s +
537                                                  1] + "fixed_" + img_names[idx][pos_s + 1:]
538             cv.imwrite(fixed_file_name, timelapser.getDst())
539         else:
540             blender.feed(cv.UMat(image_warped_s), mask_warped, corners[idx])
541     if not timelapse:
542         result = None
543         result_mask = None
544         result, result_mask = blender.blend(result, result_mask)
545         cv.imwrite(result_name, result)
546         zoom_x = 600.0 / result.shape[1]
547         dst = cv.normalize(src=result, dst=None, alpha=255.,
548                            norm_type=cv.NORM_MINMAX, dtype=cv.CV_8U)
549         dst = cv.resize(dst, dsize=None, fx=zoom_x, fy=zoom_x)
550         # cv.imshow(result_name, dst)
551         # cv.waitKey()
552 
553     print("Done")
554 
555 
556 if __name__ == '__main__':
557     main()
558     # cv.destroyAllWindows()
```

Listing C.1: Stitching algorithm

Bibliography

[1] J. Estes, K. Kline, and E. Collins. Remote Sensing. In Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 13144–13150. Pergamon, Oxford, January 2001.

[2] Hamid Jafarbiglu and Alireza Pourreza. A comprehensive review of remote sensing platforms, sensors, and applications in nut crops. *Computers and Electronics in Agriculture*, 197:106844, June 2022.

[3] Thomas Lillesand, Ralph W. Kiefer, and Jonathan Chipman. *Remote Sensing and Image Interpretation*. John Wiley & Sons, February 2015.

[4] Alexander F. H. Goetz and Lawrence C. Rowan. Geologic Remote Sensing. *Science*, 211(4484):781–791, February 1981.

[5] Sandra Lorenz. *The Need for Accurate Pre-processing and Data Integration for the Application of Hyperspectral Imaging in Mineral Exploration*. PhD thesis, November 2019.

[6] Samuel T. Thiele, Sandra Lorenz, Moritz Kirsch, I. Cecilia Contreras Acosta, Laura Tusa, Erik Herrmann, Robert Möckel, and Richard Gloaguen. Multi-scale, multi-sensor data integration for automated 3-D geological mapping. *Ore Geology Reviews*, 136:104252, September 2021.

[7] Moritz Kirsch, Sandra Lorenz, Robert Zimmermann, Laura Tusa, Robert Möckel, Philip Hödl, René Booysen, Mahdi Khodadadzadeh, and Richard Gloaguen. Integration of Terrestrial and Drone-Borne Hyperspectral and Photogrammetric Sensing Methods for Exploration Mapping and Mining Monitoring. *Remote Sensing*, 10(9):1366, September 2018.

[8] Dr S. A. Drury and S. A. Drury. *Image Interpretation in Geology*. Routledge, Malden, MA, 3rd edition edition, November 2004.

[9] Ravi P. Gupta. *Remote Sensing Geology*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[10] David Ball. The Electromagnetic Spectrum: A History. *Spectroscopy*, 22(3):14–20, March 2007.

[11] T. L. Dimitrova and A. Weis. The wave-particle duality of light: A demonstration experiment. *American Journal of Physics*, 76(2):137–142, February 2008.

[12] Helge Aasen, Eija Honkavaara, Arko Lucieer, and Pablo J. Zarco-Tejada. Quantitative Remote Sensing at Ultra-High Resolution with UAV Spectroscopy: A Review of Sensor Technology, Measurement Procedures, and Data Correction Workflows. *Remote Sensing*, 10(7):1091, July 2018.

[13] Corporation Newport. IR Absorption Spectroscopy. https://www.newport.com/n/ir-absorption-spectroscopy, 2019.

[14] Corporation Newport. FT-IR Spectroscopy. https://www.newport.com/n/introduction-to-ftir-spectroscopy, 2020.

[15] Corporation Thermo Nicolet. Introduction to Fourier Transform Infrared Spectrometry. https://www.chem.uci.edu/~dmitryf/manuals/Fundamentals/FTIR%20principles.pdf, 2001.

[16] Telops Telops. TELOPS General Presentation, 2017.

[17] Demetrios Gatziolis and Hans-Erik. Andersen. A guide to LIDAR data acquisition and processing for the forests of the Pacific Northwest. Technical Report PNW-GTR-768, U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station, Portland, OR, 2008.

[18] Anette Eltner, Dirk Hoffmeister, Andreas Kaiser, Pierre Karrasch, Lasse Klingbeil, Claudia Stöcker, and Alessio Rovere. *UAVs for the Environmental Sciences*. March 2022.

[19] José Manuel Amigo, Hamid Babamoradi, and Saioa Elcoroaristizabal. Hyperspectral image analysis. A tutorial. *Analytica Chimica Acta*, 896:34–51, October 2015.

[20] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59, December 2020.

[21] Sandra Lorenz, Sara Salehi, Moritz Kirsch, Robert Zimmermann, Gabriel Unger, Erik Vest Sørensen, and Richard Gloaguen. Radiometric Correction and 3D Integration of Long-Range Ground-Based Hyperspectral Imagery for Mineral Exploration of Vertical Outcrops. *Remote Sensing*, 10(2):176, January 2018.

[22] Floyd F. Sabins. *Remote Sensing: Principles and Interpretations*. W. H. Freeman, New York, third edition edition, October 1996.

[23] F E Nicodemus, J C Richmond, J J Hsia, I W Ginsberg, and T Limperis. Geometrical considerations and nomenclature for reflectance. Technical Report NBS MONO 160, National Bureau of Standards, Gaithersburg, MD, 1977.

[24] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, December 2000.

[25] Joaquim Salvi, Xavier Armangué, and Joan Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35(7):1617–1635, July 2002.

[26] Li Long and Shan Dongri. Review of Camera Calibration Algorithms. In Sanjiv K. Bhatia, Shailesh Tiwari, Krishn K. Mishra, and Munesh C. Trivedi, editors, *Advances in Computer Communication and Computational Sciences*, Advances in Intelligent Systems and Computing, pages 723–732, Singapore, 2019. Springer.

[27] Wang Qi, Fu Li, and Liu Zhenzhong. Review on camera calibration. In *2010 Chinese Control and Decision Conference*, pages 3354–3358, May 2010.

[28] Hassan Ghassemian. A review of remote sensing image fusion methods. *Information Fusion*, 32:75–89, November 2016.

[29] C. Pohl and J. L. Van Genderen. Review article Multisensor image fusion in remote sensing: Concepts, methods and applications. *International Journal of Remote Sensing*, 19(5):823–854, January 1998.

[30] P.A. van den Elsen, E.-J.D. Pol, and M.A. Viergever. Medical image matching-a review with classification. *IEEE Engineering in Medicine and Biology Magazine*, 12(1):26–39, March 1993.

[31] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[32] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 404–417, Berlin, Heidelberg, 2006. Springer.

[33] Edward Rosten and Tom Drummond. Machine Learning for High-Speed Corner Detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 430–443, Berlin, Heidelberg, 2006. Springer.

[34] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, Lecture Notes in Computer Science, pages 102–115, Berlin, Heidelberg, 2008. Springer.

[35] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Barcelona, Spain, November 2011. IEEE.

[36] M.J. Westoby, J. Brasington, N.F. Glasser, M.J. Hambrey, and J.M. Reynolds. 'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, December 2012.

[37] Richard Szeliski. *Computer Vision: Algorithms and Applications, 2nd Edition*. 2022.

[38] Minas Spetsakis and John Yiannis Aloimonos. A multi-frame approach to visual motion perception. *International Journal of Computer Vision*, 6(3):245–255, August 1991.

[39] Richard Szeliski and Sing Bing Kang. Recovering 3D Shape and Motion from Image Streams Using Nonlinear Least Squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, March 1994.

[40] Dany Laksono. *CloudSfM: 3D Reconstruction Using Structure from Motion Web Service*. PhD thesis, January 2016.

[41] Hojat Shirmard, Ehsan Farahbakhsh, R. Dietmar Muller, and Rohitash Chandra. A review of machine learning in processing remote sensing data for mineral exploration. *Remote Sensing of Environment*, 268:112750, January 2022.

[42] Qiuming Cheng. Spatial and scaling modelling for geochemical anomaly separation. *Journal of Geochemical Exploration*, 65(3):175–194, May 1999.

[43] David W. Leverington and Wooil M. Moon. Landsat-TM-Based Discrimination of Lithological Units Associated with the Purtuniq Ophiolite, Quebec, Canada. *Remote Sensing*, 4(5):1208–1231, May 2012.

[44] Lawrence C. Rowan, Robert G. Schmidt, and John C. Mars. Distribution of hydrothermally altered rocks in the Reko Diq, Pakistan mineralized area based on spectral analysis of ASTER data. *Remote Sensing of Environment*, 104(1):74–87, September 2006.

[45] Tsilavo Raharimahefa and Timothy M. Kusky. Structural and remote sensing analysis of the Betsimisaraka Suture in northeastern Madagascar. *Gondwana Research*, 15(1):14–27, February 2009.

[46] Pankajini Mahanta and Sabyasachi Maiti. Regional scale demarcation of alteration zone using ASTER imageries in South Purulia Shear Zone, East India: Implication for mineral exploration in vegetated regions. *Ore Geology Reviews*, 102:846–861, November 2018.

[47] Andreas Eisele, Ian Lau, Robert Hewson, Dan Carter, Buddy Wheaton, Cindy Ong, Thomas John Cudahy, Sabine Chabrillat, and Hermann Kaufmann. Applicability of the Thermal Infrared Spectral Region for the Prediction of Soil Properties Across Semi-Arid Agricultural Landscapes. *Remote Sensing*, 4(11):3265–3286, November 2012.

[48] Corinne Myrtha Frey and Eberhard Parlow. Flux Measurements in Cairo. Part 2: On the Determination of the Spatial Radiation and Energy Balance Using ASTER Satellite Data. *Remote Sensing*, 4(9):2635–2660, September 2012.

[49] Igor Ogashawara and Vanessa Da Silva Brum Bastos. A Quantitative Approach for Analyzing the Relationship between Urban Heat Islands and Land Cover. *Remote Sensing*, 4(11):3596–3618, November 2012.

[50] Adam Carter and Michael Ramsey. Long-Term Volcanic Activity at Shiveluch Volcano: Nine Years of ASTER Spaceborne Thermal Infrared Observations. *Remote Sensing*, 2(11):2571–2583, November 2010.

[51] Sandra Jakob, Robert Zimmermann, and Richard Gloaguen. The Need for Accurate Geometric and Radiometric Corrections of Drone-Borne Hyperspectral Data for Mineral Exploration: MEPHySTo—A Toolbox for Pre-Processing Drone-Borne Hyperspectral Data. *Remote Sensing*, 9(1):88, January 2017.

[52] Stephane Boubanga-Tombet, Alexandrine Huot, Iwan Vitins, Stefan Heuberger, Christophe Veuve, Andreas Eisele, Rob Hewson, Eric Guyot, Frédérick Marcotte, and Martin Chamberland. Thermal Infrared Hyperspectral Imaging for Mineralogy Mapping of a Mine Face. *Remote Sensing*, 10(10):1518, September 2018.

[53] hif-explo. Hylite https://github.com/hifexplo/hylite, December 2022.

[54] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks, March 2020.

[55] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description, April 2018.

ملخص:

يتيح استخدام الصور والبيانات عالية الدقة التي تم الحصول عليها من المنصات الأرضية أو الجوية إجراء فحص مفصل للميزات الموجودة على سطح الأرض ، مثل الرواسب المعدنية والتكوينات الجيومورفولوجية والهياكل الجيولوجية. تُستخدم هذه المعلومات لتخطيط ودراسة ما تحت السطح ، وفهم العمليات الجيوديناميكية ، وتقييم الأثر البيئي. يستخدم الاستشعار عن بعد فائق الطيف الطيف الكهرومغناطيسي المنعكس أو المنبعث من الأجسام الموجودة على سطح الأرض في تطبيقات مختلفة ، بما في ذلك تحديد المعادن وتحليل توزيع وخصائص المواد .ومع ذلك ، فإن تحقيق تمثيل دقيق للهدف المرصود يتطلب اندماج وتسجيل مشترك للبيانات التي تم الحصول عليها من أجهزة استشعار مختلفة بسبب طرق الاستحواذ المختلفة لكل جزء من الطيف الكهرومغناطيسي.

تقدم هذه الدراسة سير عمل التسجيل المشترك شبه التلقائي لدمج بيانات الاستشعار عن بعد الجيولوجية القريبة المدى. تم تصميم هذه الطريقة للتغلب على تحديات تكوين السحاب الفائق ، حيث يتم جمع مصادر متعددة من البيانات الفائقة الطيفية ودمجها في مساحة ثلاثية الأبعاد. تتضمن المنهجية عملية معايرة الكاميرا للتخلص من التشوهات الأولية ، تليها خياطة المكعبات الصغيرة ذات الموجات الطويلة بالأشعة تحت الحمراء التي تم الحصول عليها من جهاز Hyper-cam لتشكيل mosaic مفرطة. تتضمن الخطوة التالية المطابقة والتحويل لإنشاء تحول أفيني بين الفسيفساء الفائقة وصورة RGB ، وتم تقييم جودتها من خلال مقارنة الاختلافات بين إحداثيات النقاط المحددة والتحولات المقابلة لها. بعد ذلك ، يتم تنفيذ سير عمل Structure-from-Motion لإنشاء سحابة نقطية وتحديد موضع كاميرا إطار RGB واتجاهها. يسمح هذا بعد ذلك بإمكانية إجراء الإسقاط الخلفي على سحابة النقطة لتوليد سحابة مفرطة جزئية. تم تقييم عملية الإسقاط الخلفي من خلال حساب المسافة ثلاثية الأبعاد بين النقاط المحددة وإسقاط كل منها. تم تحقيق تفاوت متوسط قدره 3.56 سم. يمكن ملء hypercloud ببيانات من مناطق أخرى من الطيف الكهرومغناطيسي و / أو استخدامها لإنتاج خرائط معدنية ومنتجات جيولوجية أخرى ذات صلة.

الكلمات الرئيسية: التسجيل المشترك ، الأشعة تحت الحمراء طويلة الموجة ، Hypercloud ، Telops Hyper-Cam، الهيكل من الحركة (SfM)، التماثل ، التحويل الأفيني

ROYAUME DU MAROC
INSTITUT AGRONOMIQUE
ET VÉTÉRINAIRE HASSAN II

المملكة المغربية
معهد الحسن الثاني
للزراعة والبيطرة

مشروع نهاية الدراسات لنيل دبلوم مهندس في الطبوغرافية

توطين ، إسقاط خلفي ، ودمج بيانات الأشعة تحت الحمراء الفائقة الطيفية طويلة الموجة من مستشعرات فوريه للتصوير بالأشعة تحت الحمراء

قدم للعموم ونوقش من طرف:

أيوب الفاتحي

أمام اللجنة المكونة من:

الأستاذة كنزة آيت القاضي	(معهد الحسن الثاني للزراعة و البيطرة)	رئيسة
الأستاذة إيمان الصيباري	(معهد الحسن الثاني للزراعة و البيطرة)	مقررة
الأستاذ رضى اليعقوبي	(معهد الحسن الثاني للزراعة و البيطرة)	مقرر
الدكتورة ساندرا لورنز	(HZDR – HIF)	مقررة
الدكتور ساموبل ثيبل	(HZDR – HIF)	مقرر

مارس 2023

Adresse : Madinat Al Irfane, B.P. 6202. Rabat – Maroc
Tél : (00 212) 0537 77 17 58/59
Fax : (00 212) 0537 77 58 45
Site web : http://www.iav.ac.ma

العنوان: ص. ب 6202 الرباط المعاهد الرباط – المغرب
الهاتف: 59 / 58 17 77 0537 (00 212)
الفاكس: 45 58 77 0537 (00 212)
موقع الأنترنت:http://www.iav.ac.ma