BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DETECTION OF CHANGES IN THE SCENE CAPTURED BY THE DRONE AT DIFFERENT TIMES

DETEKCE ZMĚN VE SCÉNĚ SNÍMANÉ DRONEM V RŮZNÝCH ČASECH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. MARTIN MINÁRIK
SUPERVISOR VEDOUCÍ PRÁCE	Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2022

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Master's Thesis Specification

23294

Student:	**Minárik Martin, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Software Engineering
Title:	**Detection of changes in the scene captured by the drone at different times**
Category:	Signal Processing

Assignment:

1. Get acquainted with methods for processing and analysis of sensory data captured by a commonly available drone (especially camera, position, etc.).
2. Design a procedure that appropriately preprocesses the data captured by the drone at different times to analyze changes in the scanned scene. Design user process and relevant GUI for resulting tool.
3. Implement the designed application using relevant available technologies and libraries.
4. Evaluate the properties of the resulting solution based on real-world experiments.
5. Present the key features of the solution in the form of a poster and a short video.

Recommended literature:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- S. M. Seitz, B. Curless, J. Diebel, D. Scharstein and R. Szeliski, *A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms,* 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 519-528, doi: 10.1109/CVPR.2006.19.
- J. L. Schönberger and J. Frahm, *Structure-from-Motion Revisited*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104-4113, doi: 10.1109/CVPR.2016.445.

Requirements for the semestral defence:

- Items 1, 2, and partially item 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Beran Vítězslav, Ing., Ph.D.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	May 18, 2022
Approval date:	March 24, 2022

Abstract

This work aims to design the user processes and implement an application used to detect changes in the scene captured by the drone at different times. The application supplies a complex solution for the three-dimensional reconstruction and comparison of two aviation data. The main benefits include being an all-in-one solution, operating with unmodified data from the drone, reconstruction and alignment without any human adjustments, and no necessity for ground control points or specialized sensors, making this tool available for a broad assortment of use cases. In the thesis, I created a pipeline that uses state-of-the-art reconstruction algorithms, allowing the user to upload, organize, and reconstruct the data into three-dimensional models and view the difference in volume between the flights at different times.

Abstrakt

Cieľom tejto práce je navrhnúť používateľské procesy a naimplementovať aplikáciu, ktorá zaznamenáva zmeny v scéne snímanej dronom v rôznych časoch. Aplikácia ponúka komplexné riešenie pre trojdimenzionálnu rekonštrukciu a porovnanie dvoch setov leteckých dát. Medzi hlavné benefity patrí, že ide o riešenie all-in-one, pri ktorom sa využívajú nemodifikované dáta, a ktoré umožňuje rekonštrukciu a zarovnanie bez nutnosti zásahu používateľa a nevyžaduje prítomnosť pozemných kontrolných bodov alebo špecializovaných senzorov, čo ponúka široké možnosti použitia. Vo svojej práci som vytvoril aplikáciu využívajúcu najlepšie rekonštrukčné nástroje súčasnosti, ktorá dáva používateľovi možnosť nahrávať dáta, usporiadavať ich do letov, rekonštruovať ich do trojdimenzionálnych modelov a zobrazovať rozdiely v objeme medzi letmi v rôznych časoch.

Keywords

drone, 3D object reconstruction, structure from motion, difference in volume, point cloud, photogrammetry

Klíčová slova

dron, rekonštrukcia 3D objektov, štruktúra z pohybu, zmena objemu, bodové mračno, fotogrametria

Reference

MINÁRIK, Martin. *Detection of changes in the scene captured by the drone at different times.* Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vítězslav Beran, Ph.D.

Rozšířený abstrakt

Cieľom tejto práce je vytvoriť aplikáciu na detekciu zmien v scéne snímanej dronom v rôznych časoch. Práca ponúka riešenie problému, v ktorom používateľ, ktorý získal dáta počas viacerých letov dronom a chce tieto dvojdimenzionálne dáta v podobe fotografií alebo videa previezť do trojdimenzionálnej podoby a následne ich porovnať, vidieť zmenu monitorovaného priestoru v čase. Riešenie takéhoto problému bolo doteraz možné iba za použitia viacerých, samostatných druhov softvéru s explicitnými zásahmi používateľa.

Prvou časťou práce bolo vytvoriť spomínaný používateľský proces, pomocou ktorého používateľ dokáže manuálne rekonštruovať trojdimenzionálny objekt. Táto časť pomohla objasniť aktuálne trendy a existujúce čiastkové riešenia. Implementácia jednotlivých krokov riešenia bola najprv iba v vo forme aplikácie v príkazovom riadku. Táto verzia však nebola používateľsky prívetivá a aplikácia neponúkala používateľské rozhranie. Nutnosť použitia rekonštrukčných knižních a užívateľského rozhrania vykryštalizovali finálnu architektúru aplikácie. Aplikácia beží ako samostatný docker kontajner, ktorý obsahuje väčšinu podstatných knižníc. Obmedzením inštalácie potrebných knižníc sa uľahčila jej distribúcia, aplikácia sa stala bezpečnejšia, spolahlivejšia a ľahšie verzovateľná.

Požívateľ, ktorý získal dáta z dronu po spustení aplikácie do nej dokáže nahrať dáta. Tieto obsahujú čas, miesto a poznámku, aby ich mohol používateľ rozoznávať a organizovať. Po nahraní dát sa môže spustiť proces rekonštrukcie. Ten beží na pozadí, používateľ je o stave a prípadnom neúspechu informovaný pomocou logov. Proces rekonštrukcie začína vytvorením použíteľných dát, a teda rozsekaním vstupu, v prípade videa, na jednotlivé snímky. S použiteľným vstupom sa začne proces algoritmu štruktúry z pohybu. Tento proces je implementovaný knižnicou OpenSfM, ktorú obsahuje OpenDroneMap. Z fotografií sa najprv zistia charakteristické body, ktoré sú potom na základe podobnosti spojené, čo vytvára riedke bodové mračno a fotky kamier.

Výsledky algoritmu štruktúru z pohybu sú potom posunuté algoritmu Multi-view Stereo. Tento algoritmus iteratívne rekonštruuje scénu a pridáva do nej tak nové detaily. Algoritmus Multi-view Stereo je implementovaný knižniou OpenMVS, ktorá je opäť súčasťou OpenDroneMap. Výsledkom tohto kroku je husté bodové mračno, ktoré sa dá použiť na výpočty.

Pred výpočtami sa však tieto bodové mračná geo-referencujú. Pri tomto procese sa na základe ich metadát zisťuje, kde boli vyfotografované a následne sa tieto informácie pridajú do bodového mračna, transformujúc ho z lokálnej polohy do geo-referencovanej.

Pár takýchto dát, vo formáte .csv, upravených pomocou knižnice numpy dokážu byť použité na výpočet zmien medzi nimi. Najprv sa pomocou funkcií numpy upravia vstupné point cloudy, tie sa následne pomocou knižnice Open3D prevzorkujú a vytvorí sa Delaunayova triangulácia zmien, ktorá sa dá použiť ako trojdimenzionálny objekt pre reprezentáciu zmien, z vizuálnych výsledkov sa tiež vytvoria aj dvojdimenzionálne grafy, ukazujúce rozdiely v objeme medzi dvoma dátami. Jedná sa iba o približné zmeny, aplikácia ku georeferencovaniu používa iba exif gps dáta dronu bez akýchkoľvek špeciálnych senzorov alebo pozemných kontrolných bodov.

Vizualizácia ako aj užívateľské prostredie beží pomocou klient-server aplikácie, ktorá je napísaná pomocou mikroframeworku Flask. Pre spúšťanie a komunikáciu sa využíva uwsgi socket a ako webový server sa používa Nginx. O vizualizáciu aplikácie sa stará jazyk Jinja2 za pomoci Bootstrapu pre štýly.

Využitie knižnice OpenDroneMap je riešenie cez komunikáciu medzi dvoma kontajnermi vo forme súrodeneckých kontajnerov, kedy má aplikácia k sebe pripojený nie len lokálny priečinok pre prístup k dáta

Výsledkom práce je nástroj, ktorý dokáže zrekonštruovať a odčítať od seba dve dáta z dronov. Navyše je to však aj postup výpočtu, ktorý sa dá použiť ako príručka pre získanie vedomostí o jednotlivých krokoch rekonštrukcie trojdimenzionálnych objektov z dvojdimenzionálnych dát.

Detection of changes in the scene captured by the drone at different times

Declaration

I declare that I have prepared this Masters's thesis independently, under the supervision of Ing. Vítězslav Beran, Ph. D. I listed all of the literary sources and publications that I have used.

. . . .
Martin Minárik
May 18, 2022

Acknowledgements

I would like to thank my supervisor Ing. Vítězslav Beran, Ph.D. for his help and patience during the creation of this thesis.

Contents

Chapter 1

Introduction

This thesis studies the possibilities of calculating volume using consumer-grade drones without any ground control points or additional equipment. It discusses the current trends of drones and how non-professional equipment can be used in professional fields like construction to safely, swiftly, and efficiently gain valuable information that would otherwise be dangerous, time-consuming, or even not possible. In this case, it is information about the volume and its history. The drone flies the same mission over the designated area multiple times capturing the information needed to reconstruct the location in 3D. This information is then reconstructed into a dense point cloud using structure from a motion algorithm, as well as deep multi-view stereo. The models are aligned to each other using drone metadata combined with a random sample consensus algorithm and can be visualized in the application. In the application, the changes in volume can be calculated. In the second chapter, I will cover general information about unmanned aerial vehicles, their specific sensors, current legislation, use cases of drones in hobby and professional fields, and current existing solutions.

In the next chapter, I will specify the algorithms and the architecture of the application. I will go over each algorithm and explain the concepts used in the application. The following chapter will discuss the proposed solution, the pipeline, and details concerning the implementation. This chapter will include problems that occurred during the implementation, their fixes, and their shortcomings. It will also contain a graphical user interface, its elements, and its usage.

In the last chapter, I will calculate the deviations of the solution, compare my solution with other existing solutions, and conclude the usability and effectiveness of the solution.

Chapter 2

Unmanned aerial vehicles

An unmanned aerial vehicle (UAV) is an aircraft without a human operator onboard and is commonly referred to as a drone, but also as a remotely piloted vehicle (RPV), remotely piloted aircraft (RPA), remotely operated aircraft (ROA), or, in the case of UAVs with specific combat roles, as an unmanned aerial combat vehicle (UACV). In contrast to conventional aircraft, UAVs are controlled either remotely by human operators or are guided by a computer program with various levels of automation and autonomy [24]. UAVs can be categorized into categories based on their aerial platform.

Firstly, there are UAVs using rotors, which are usually called drones. These are the most common UAVs as they can be affordable and easy to operate for hobbyists as well as professionals. UAVs using rotors can be further classified based on the number of rotors they have. Single rotor UAVs have a helicopter-like structure. These UAVs have one big rotor on the top and a small one on the tail of the vehicle. The small rotor is used to stabilize and make the vehicle more controllable just like a helicopter. When it comes to flying times, these vehicles are very efficient and can last a long time in the air. The downside of these UAVs is that they need high operational skills because of their complexity. Multi-rotor UAVs are the easiest to manufacture and that makes them the cheapest option available. Typically, they are named based on the number of the rotors. For example, a quadcopter has four rotors, a hexacopter has six rotors, et cetera. Unlike single rotor UAVs, multi-rotor ones use different rotations per minute of the rotors to maneuver. Thanks to the increased number of rotors, multi-rotor UAVs offer great balance, usually the higher the balance the bigger the number of rotors.

Secondly, UAVs with fixed-wing architecture are using wing designs like normal airplanes. These wings are enabling flight time of up to 16 hours or higher, which is significantly more than the rotor UAVs. Additionally, specific types of these vehicles can be loaded with larger cargo. Some of the downsides of fixed-wing aircraft include their high manufacturing cost, and more complicated taking off. Usually, these aircraft need a runway or a different launcher mechanism such as a catapult launcher. When in the air, management of these aircraft is more complex, but at the same time offers more control even if the aircraft's motor fails.

The last category is hybrid VTOL UAVs, offering the benefits of fixed-wing architecture with that of rotor-based aircraft. This idea has been implemented since the 1960s without much success. At the current time, the new generation of vehicles takes advantage of the improved gyroscope and accelerometer which have big potential. Sensors in the aircraft usually work autonomously stabilizing the aircraft in the air. The pilot's task is only to guide the aircraft on the desired course.

2.1 Drones and regulations

Historically, the first use of UAVs was in the army after World War I, but mostly in the Second World War where they were used to drop bombs and explore battlegrounds. In the early-2010s, the innovation of UAVs sparked commercial interest. As the micro-controllers, accelerometers, sensors, and other parts got cheaper, hobbyists started using four or more rotor-equipped aircraft for their accessibility and ease of use. Vehicles amongst hobbyists and in commercial use are primarily used for their cameras in photography or videography. This goes hand in hand with the rapid growth of smartphone technology as the smartphone is commonly used for the live camera feed and drone controls lessening the end cost.

However, drones have a wider area of use, with no one on board, it is safe to plan missions that would be dangerous or inaccessible for a human. An example of this is geographic mapping of inaccessible terrain and locations using photogrammetry. This could be done also on a vertical axis to perform safety inspections of, for example, electrical poles or other buildings and structures. Using additional sensors, such as thermal sensors, drones can be helpful in search and rescue operations. Specific professional fields have also adapted their uses of drones. In agriculture, drones are used to precisely monitor crops. Drones are also used on construction sites to monitor the progress and match the planning to reality.

Drone sensors and modules

To perform a range from basic tasks such as landing where a drone took off, to specific tasks like precisely measuring the distance to Earth, drones are equipped with sensors that monitor surroundings and gathered information is used to guide the pilot or drone's autonomous behavior.

Accelerometer - Determines position and orientation of the drone in flight. There are more ways how accelerometers could be implemented in drones. One type senses the movement in parts of the circuit and sends a small electrical current according to the change relative to gravity. Drones could also use technology based on thermal sensing, which has the advantage of no moving parts, instead, it is based on the movement of gas molecules [25].

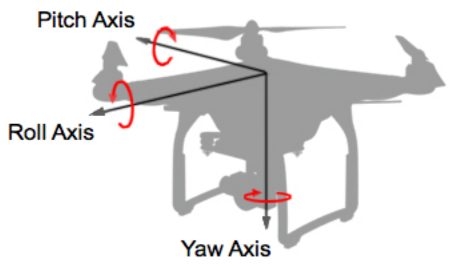


Figure 2.1: Different types of drone axis.[1]

Gyroscope - An essential part providing stabilization which results in maintaining level flight. Drones either use three-axis or six-axis gyro stabilization when using the ac-

[1]Taken from https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html

celerometer together with a gyroscope to measure the amount of static acceleration due to gravity. Three-axis gyros provide a calculation of rotation in three-axis using the coordinate right-hand rule: roll, pitch, and yaw (as seen in the figure 2.1) [7].

- The roll provides rotation around the front-to-back axis.
- The pitch provides rotation around the side-to-side axis.
- The yaw provides rotation around the vertical axis.

Global Positioning System (GPS) receiver - a global navigation satellite system that provides location, velocity, and time synchronization. Several satellites are orbiting the Earth sending microwave signals, which can be interpreted by GPS receivers as information about the distance from a satellite. If a GPS receiver has a signal from at least fours satellites at the same time, it can calculate its coordinate location. GPS can be used to determine 2D location data, take precise measurements of time, and it can be even used as an altimeter, although it is not accurate [4].

Global navigation satellite systems (GNSS) receiver - a type of satellite navigation that provides global coverage. This navigation system provides geo-spatial positioning information to the devices autonomously, allowing devices' receivers to determine their precise location on the surface of the Earth. Unlike GPS, GNSS has over 60 satellites available for viewing. While no additional information is available, the accuracy is improved which results in more precise location data [4].

Light detection and ranging (Lidar) sensor - a remote sensing method that uses light in the form of a pulsed laser to measure the ranges to the Earth. The pulses then get combined with other drone-recorded data resulting in precise 3D information about the shape of the Earth. There are two types of lidars, topographic and bathymetric. Topographic Lidar uses a near-infrared laser which works well with sensing land, whereas bathymetric Lidar uses green light, which penetrates the water, enabling it to measure riverbed elevations [16]. The downside of using Lidar is similar to using cameras and that results in lower visibility scenarios like smoke.

Thermographic camera - An imaging sensor that is sensitive to wavelengths in the infrared region of the electromagnetic spectrum to form an image. The infrared spectrum is not visible to human eyes. These cameras use special detectors for different wavelengths such as shortwave infrared region (SWIR), mid-wave infrared region (MWIR), and long-wave infrared region (LWIR) [23]. Thermographic cameras are sensitive to heat and reflect different temperatures with colors ranging from blue to red.

Gas detector - a detector using tunable diode laser absorption spectroscopy (TDLAS) to analyze the properties and constituents of gasses such as concentration, temperature, pressure, and flow velocity. it is working by penetrating the gasses by diode laser light and measuring the wavelengths with a photodetector. If the reduction of the light intensity is high, there is a high amount of gas in the air and vice versa [2].

Drone regulations in the European union

EU Regulations 2019/947[2] and 2019/945[3] set out a risk-based framework for the safe operation of civil drones. They distinguish between the weight of the drone, its specifications, and the operations that the civil drone is intended to conduct. These regulations introduce three categories of civil drone operations.

The open category - Includes lower-risk civil drone operations, where the civil drone operator is responsible for complying with the relevant requirements for its intended operation. There is no authorization necessary for flying a drone as the risks in this category are low. This category is subdivided into three categories, A1, A2, and A3:

- A1 - This category is divided into additional two subcategories; with the max weight of the drone under 250 grams. Everyone can fly these drones with no age restrictions or training required. The second subcategory applies to drones under 500 grams. With such drones, there is an age restriction of 16+ years and the pilot must have passed the exam for A1/A3 subcategory drones. Both of these subcategories have restrictions not to fly over uninvolved people and not to fly over assemblies of people.

- A2 - The drone weight limit of this category is up to 2 kilograms. The age and training requirements are the same as for the drones over 250 grams. Operational restrictions in this category are not to fly over uninvolved people as well as keep a horizontal distance of 50 meters from uninvolved people.

- A3 - This category covers all drones from 2 kilograms up to 25 kilograms. The same requirements apply when it comes to age and training. Operational restrictions include not flying near or over people and flying at least 150 meters away from residential, commercial, or industrial areas.

The specific category - Addresses civil drone missions, where safety risk is higher and the civil drone operator needs to have operational authorization by the national competent authorities before starting the mission. To get the required authorization, the operator must conduct a risk assessment for the safe flight operation.

The certified category - Covers considerably high safety risk operations; therefore the certification of the drone and the drone operator as well as licensing of the remote pilots is always required [1]. Since December 31, 2020, every drone operator must be registered in their resident country or place of business. The legislation distinguishes between drone operators and remote pilots. The drone operator is usually the owner of the drone, who does not necessarily have to fly the drone but is responsible for the operation. The remote pilot is the one who controls the drone. He must have undergone the appropriate training for the operation to be conducted. All of the photo and video footage made with the drone must comply with General Data Protection Regulation (GDPR). This means that it is prohibited to collect and use the personal data of uninvolved people.

[2]https://www.easa.europa.eu/document-library/regulations/commission-implementing-regulation-eu-2019947

[3]https://www.easa.europa.eu/document-library/regulations/commission-delegated-regulation-eu-2019945

2.2 DJI drones and applications

In this section, I will be talking about the most commercially popular drones, the drone that was used to create the dataset and application used for capturing as well as processing the data.

DJI drones

DJI is a Chinese technology company based in Shenzhen. Since its debut in the drone industry with Phantom in 2013, it has been known for its commercial-grade drones. In 2021, DJI made 54% of the commercial drone brand market, making DJI drones the most used drones in the world.

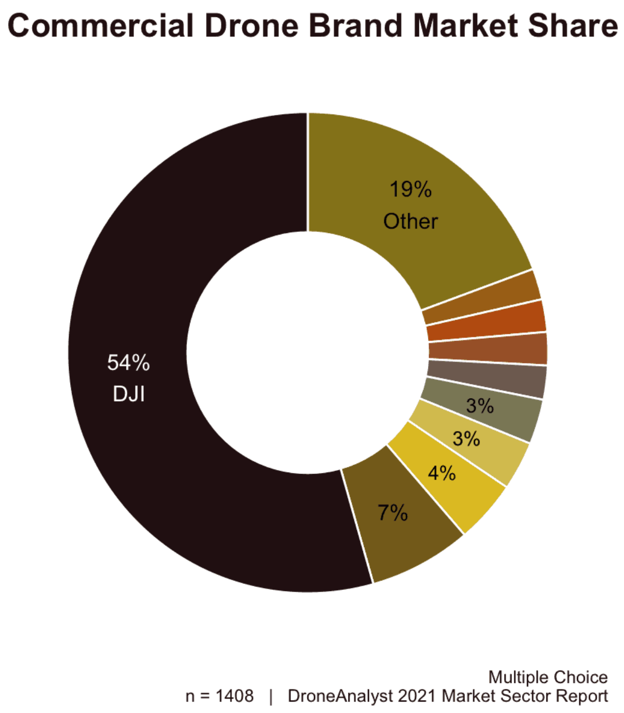


Figure 2.2: Commercial drone brand market share according to DroneAnalyst website.[4]

Because of the user base that DJI has, DJI offers developers its SDK to develop third-party applications that work with DJI drones. DJI has created SDKs for every sector of the drone-flying process. With mobile SDK, developers can get access to flight level control, access to aircraft state through telemetry and sensor data, as well as live camera and stored camera data on the drone, and create pre-defined missions in their mobile applications. UX SDK provides UI elements, including their core functionalities to complete the application. Payload SDK enables the communication and control of the additional modules used in their professional-grade drones. These drones can also use the capabilities of their Onboard SDK, which can extend the capabilities of DJI Matrice - a modular series of drones.

[4]Taken from https://droneanalyst.com/2021/09/14/how-has-2021-changed-the-drone-industry

This thesis is using data captured thanks to DJI Spark [5]. It has a takeoff weight of 300 grams, which puts it in the C1 [6] class (under 900 grams), meaning it can fly in the A1 subcategory of The "open" category under the European legislation.

It is equipped with a 1/2.3' CMOS camera sensor with an effective 12 megapixels. Spark can produce up to 4K photo resolution and very stable Full-HD 1920x1080 video at 30p. The gimbal, on which the drone's camera is located, has a pitch controllable range of -85° to 0°. This means that this drone is not well equipped for photogrammetry, as it can not get a full -90° angle. This, combined with the flight time of only up to 16 minutes, makes Spark not supported in photogrammetry applications. However, with the DJI GO 4 application, Spark's gimbal pitch can be calibrated up to 10° each way. This means that Spark with gimbal pitch calibrated to -5° can reach from -90° to -5° making it easier to use in missions involving photogrammetry.

Existing solutions for capturing and viewing drone data

Most of the applications that can offer photogrammetry are professional-grade applications that require a higher-tier drone than the DJI Spark. These applications tend to be aimed at professionals working in surveying, construction, and mapping, who use drones as a complementary addition to their usual work routine.

DJI GO 4

While not being professionally used, DJI GO 4 is the most used mobile application to fly a DJI drone [9]. Made by DJI, it is the essential flight app offering a variety of functions. The application provides a live video transmission feed from the drone to a smartphone. Many of the drone settings can be changed through the app, like altering the flying characteristics of the drone, calibrating the gimbal, and changing the camera settings. Some of the useful features include obstacle avoidance based on the sensors located built-in the drone, automating takeoff and landing options for safe flying.

Creating waypoints is a very useful feature for mission planning. In the map menu, the user can define various waypoints that can be shown with the actual position of the drone during the flight. These can then guide the pilot during the flight. If the drone is supported, there are flight modes called "Quick Shots" and "Intelligent Flight Modes". These modes allow the drone to orbit an object or track moving objects and follow them. Other than this, DJI GO provides important status indicators about the drone. In the top bar, the application provides system status, which depends on the strengths of the signals, flight mode, GPS signal, flight autonomy, remote controller signal strength, video transmission signal strength, and aircraft battery. The application is available for android and iOS.

[5] https://www.dji.com/spark/info

[6] https://www.easa.europa.eu/domains/civil-drones/drones-regulatory-framework-background/open-category-civil-drones

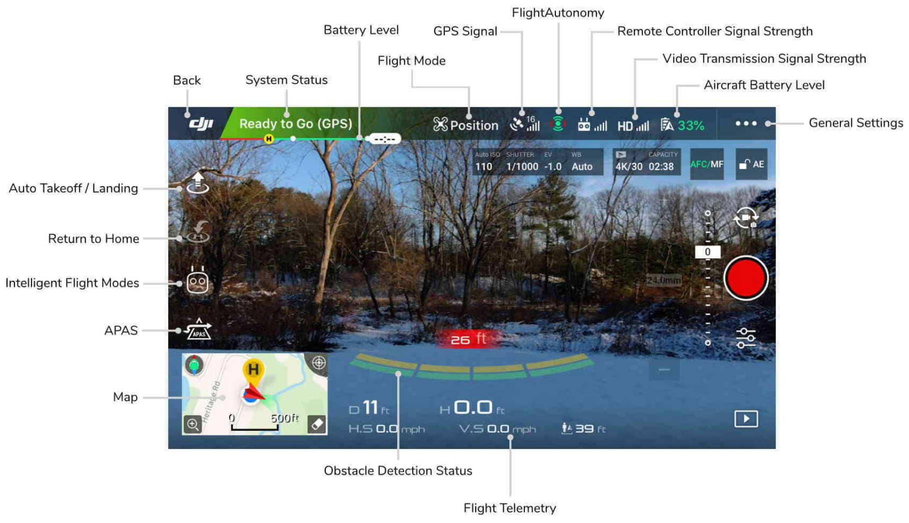


Figure 2.3: DJI Go 4 control elements.[7]

Pix4D

Is a software solution for photogrammetry using drones. The suite provides a mobile application used for mission planning and drone mapping. The application supports DJI drones. There are more options for planning missions used for 2D and 3D mapping provided. When planning a mission, the user creates a radius to be mapped and displayed over the map automatically calculating the best drone route. During this process, the flight height and approximate flight time are also calculated.

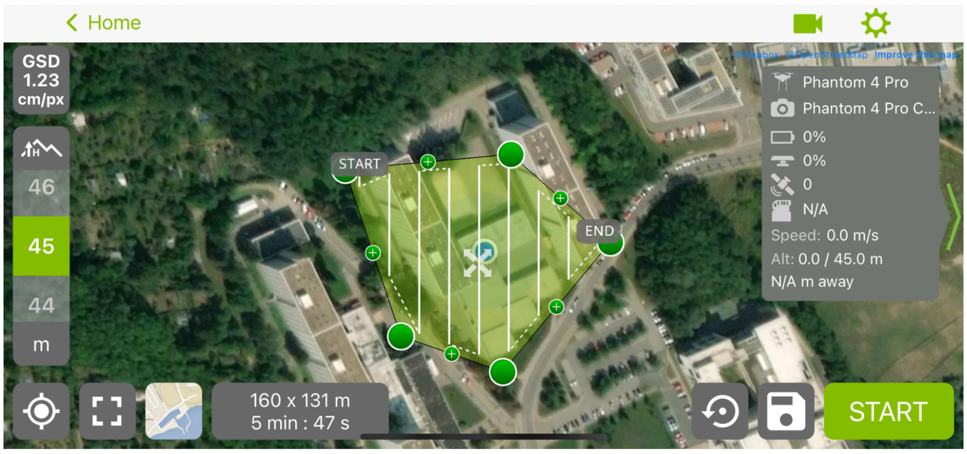


Figure 2.4: Mission planning using Pix4D

Pix4D suite also contains various computer software for processing the data from the drone. PIX4Dmapper can be used to generate a full-color point cloud, orthomosaic, digital surface models, index maps, and thermal maps. Using this software, the data from the drone could be transformed into a point cloud and manually measured and inspected in the software.

[7]Taken from https://www.dronegenuity.com/dji-go-4-app-tutorial/

DJI Terra

DJI Terra is a professional software made specifically for DJI drones used mostly in infrastructure and agriculture. It is a complex piece of software offering a wide range of functions. Terra offers waypoints for mission planning with visualization in 3D, with the ability to regulate gimbal pitch angle, speed, and more. What is more, area and oblique mission planning for photogrammetry are present, with automatic calculations for the flight estimates. If the 3D model of the surveyed structure is available, a detailed inspection can be planned, with specified points of focus. During the flight, a 2D orthomosaic model of the map can be created in real-time. After the data is gathered, 3D reconstruction can be done as well as Lidar data processing. The reconstructed data can be then analyzed, measured, and annotated and every photo creating the point cloud is inspected individually.

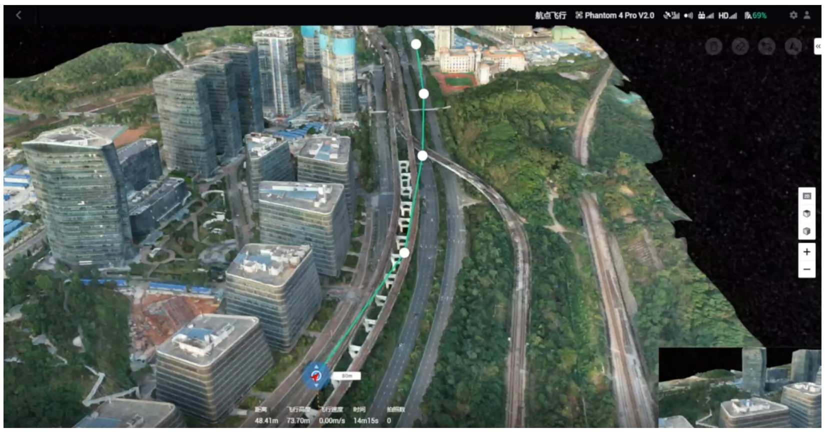


Figure 2.5: Waypoint mission planning in 3D using DJI Terra.[8]

Propeller

Is a survey data collecting tool, working on the PPK principle. This means that the data is collected through the supported drones, which the company lists on its website[9].

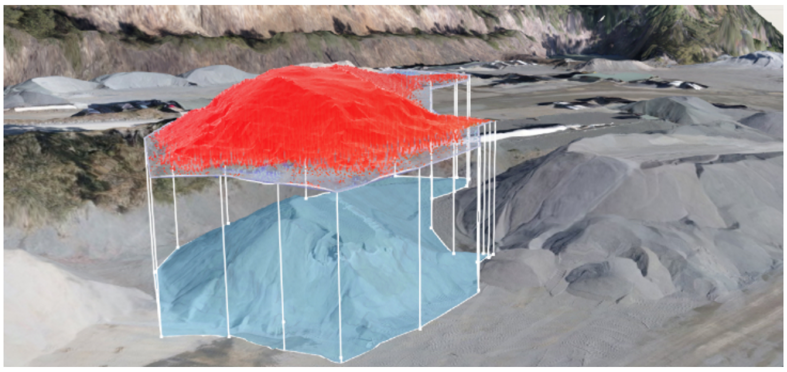


Figure 2.6: A visualization of the processed data using Propeller.

The data is geotagged using the AeroPoints, which are GCPs, that upload their position after the flight is done. After the flight, the data can be dragged and dropped into

[8]Taken from https://www.dji.com/dji-terra

the Propeller application, sent to the cloud, and processed on the cloud. This means that no powerful computing unit is necessary as the computing is done in the cloud. After up to 24-hours the processing of the data is done and the model is sent back. This can be then used to measure, inspect or even calculate the stockpile volume of the data as a part of the analysis.

[9] https://www.propelleraero.com/

Chapter 3

Acquiring accurate data

The surveyor needs to capture accurate data using UAVs. This step is crucial as not enough or poor quality of the data can cause problems during the recreation. An example of this can be found later in the thesis when talking about datasets used (as seen in section 6.3). This chapter is useful when the data is already made, as the person can look for clues of a bad dataset. Additionally, it communicates general information about the most commonly used drone data kinds.

3.1 Methods of acquiring precise data

As the development of the consumer-grade drones progressed, there was a transition from using ground control points (GCPs) to using more advanced global navigation satellite systems (GNSS). These methods can give safer and faster results while using fewer resources when used in the proper environment.

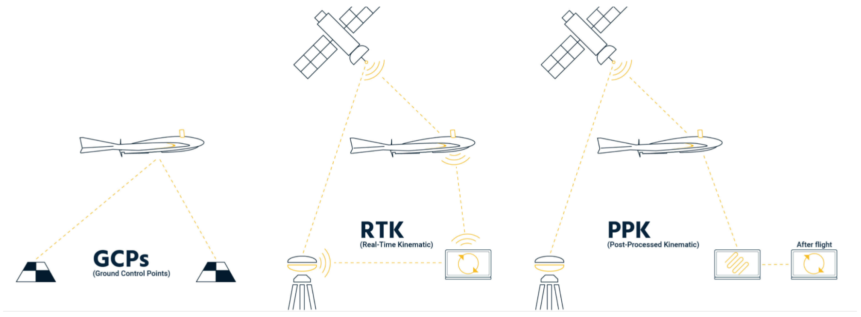


Figure 3.1: The difference between GCPs, RTK, and PPK.[1]

Ground control points (GCPs) - A location or object on the ground that has precisely known coordinates. The objects on the ground should be geo-referenced with a deviation of two to five centimeters to achieve absolute accuracy. It is the original method without using satellites that grants a consistent ground truth for the project's accuracy. Some of its cons are the fact that it needs additional equipment such as GPS rovers, VRS network licenses, and spray paint for the target. Placing the objects could be dangerous or impossible in certain environments.

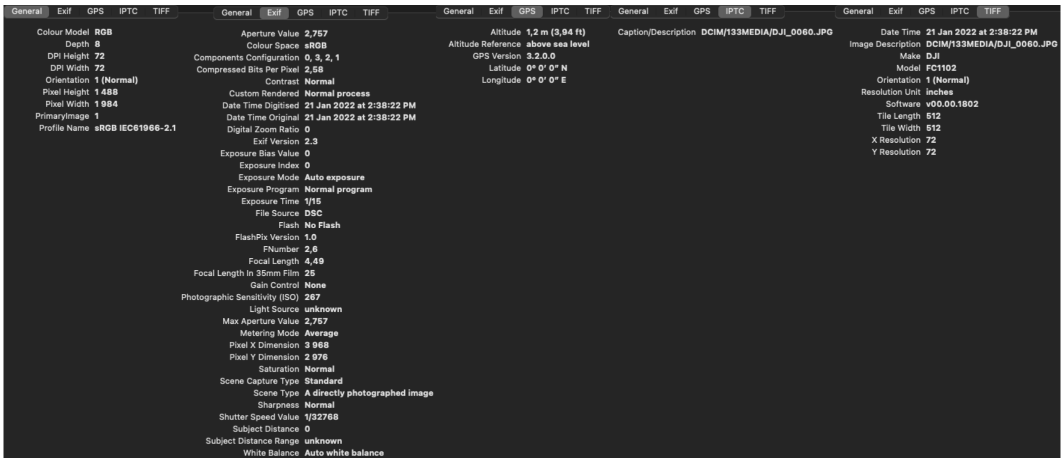
[1]Taken from https://www.sensefly.com/blog/gcps-rtk-ppk-when-what-why/

Real-Time Kinematic (RTK) - A technique used to enhance the precision of data derived from satellite-based positioning systems, which relies on a single reference station or interpolated virtual station to correct geotagged locations while in flight. Contrary to GCPs, RTK uses no marks or objects, only real-time GNSS data that improves accuracy. This means that there is no need to modify the data in post-processing. RTK works well in the terrains where there is a strong connection between the drone and the satellite as well as between the drone and the ground station. The cons of this approach are that it needs a capable drone equipped with special technology and consistent signal as malfunctioning during the flight can be present.

Post-Processing Kinematic (PPK) - An alternative to RTK. This method uses post-processing to correct geo-referencing after getting the data. Usually, the data is sent to a cloud, where post-processing is performed. Pros of this method are that this method is more universal and can be used in environments with a bad signal. Furthermore, it takes less time on-site to prepare than using an RTK connection. In comparison to RTK, PPK takes more time to use GNSS data in post-processing. What is more, using GNSS can introduce an error due to poor precision or a shift in geo-referencing.

3.2 Drone metadata

To achieve accurate results with or without using the GCPs, more detailed information about the flight is needed. The DJI drones store this information in exchangeable image file format (EXIF) data. EXIF is a standard that defines additional specific information about the image other than the pixels themselves. Other than the information about the picture itself, the metadata contains useful information used for photogrammetry and reconstruction of the 3D model. The information contains altitude (above sea level), GPS longitude and latitude as well as GPS version, drones orientation, name of the drone, software version, and others. This information is useful when aligning the photos or even point clouds.



Colour Model	RGB
Depth	8
DPI Height	72
DPI Width	72
Orientation	1 (Normal)
Pixel Height	1 488
Pixel Width	1 984
PrimaryImage	1
Profile Name	sRGB IEC61966-2.1

Aperture Value	2,757
Colour Space	sRGB
Components Configuration	0, 3, 2, 1
Compressed Bits Per Pixel	2,58
Contrast	Normal
Custom Rendered	Normal process
Date Time Digitised	21 Jan 2022 at 2:38:22 PM
Date Time Original	21 Jan 2022 at 2:38:22 PM
Digital Zoom Ratio	0
Exif Version	2.3
Exposure Bias Value	0
Exposure Index	0
Exposure Mode	Auto exposure
Exposure Program	Normal program
Exposure Time	1/15
File Source	DSC
Flash	No Flash
FlashPix Version	1.0
FNumber	2,6
Focal Length	4,49
Focal Length In 35mm Film	25
Gain Control	None
Photographic Sensitivity (ISO)	267
Light Source	unknown
Max Aperture Value	2,757
Metering Mode	Average
Pixel X Dimension	3 968
Pixel Y Dimension	2 976
Saturation	Normal
Scene Capture Type	Standard
Scene Type	A directly photographed image
Sharpness	Normal
Shutter Speed Value	1/32768
Subject Distance	0
Subject Distance Range	unknown
White Balance	Auto white balance

Altitude	1,2 m (3,94 ft)
Altitude Reference	above sea level
GPS Version	3.2.0.0
Latitude	0° 0′ 0″ N
Longitude	0° 0′ 0″ E

Caption/Description	DCIM/133MEDIA/DJI_0060.JPG

Date Time	21 Jan 2022 at 2:38:22 PM
Image Description	DCIM/133MEDIA/DJI_0060.JPG
Make	DJI
Model	FC1102
Orientation	1 (Normal)
Resolution Unit	inches
Software	v00.00.1802
Tile Length	512
Tile Width	512
X Resolution	72
Y Resolution	72

Figure 3.2: An example of DJI Spark metadata

When using multiple images for reconstruction, each has its combination of longitude, latitude, and altitude, but the problem arises when it comes to video. EXIF data from video usually contain general information such as drone manufacturer, that does not change during the flight. To record additional metadata information, DJI drones can produce a .srt file, which is a video caption file with information about the flight changing as the video frames change.

3.3 Data acquiring practices

This section discusses the best practices and essential things to look for when flying drones to gather data for the reconstruction. First of which is ground sampling distance.

Ground sampling distance (GSD)

The GSD is defined as the distance between neighboring pixel centers measured on the ground [19]. The value of GSD depends on altitude and the sensor used, knowing the GSD helps to determine the size of the object in the image. The lower the spatial resolution of the image and the less visible details.

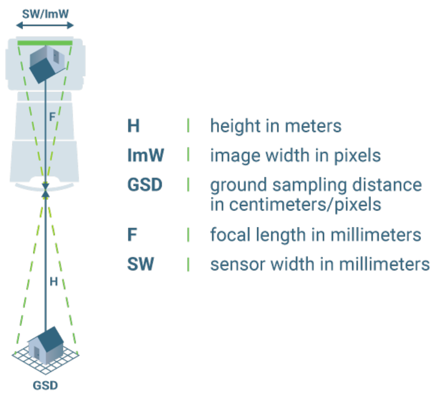

SW/ImW
F
H
GSD
H | height in meters
ImW | image width in pixels
GSD | ground sampling distance in centimeters/pixels
F | focal length in millimeters
SW | sensor width in millimeters

Figure 3.3: Variables of ground sample distance explained.[2]

The figure 3.3 shows the calculation of GSD. The flight height or the distance from the terrain or object is H. Other variables are camera specifications. It is important to calculate the GSD needed for the flight before the image acquisition as it is necessary to adjust the flight height as well as the camera specifications if needed. For example, for a detailed reconstruction of the area, a low GSD is required, meaning flying closer to the object of interest. On the other hand, while covering a large area, there is no need for precise object quality, therefore higher GSD is acceptable. This can significantly decrease the acquisition time as well as processing time. If an acquisition time is not important, then lowering the resolution before processing is always an option.

Mission planning

In addition to the GSD, the quality of the data lies in the mission planning. This means calculating the time that is needed to cover the area while looking at the drone battery. Flying and mapping the are over in more flies with replacing of taking the battery out can case minor issues because of the takeoff and landing time, together with changing

[2]Taken from https://support.pix4d.com/hc/en-us/articles/202559809-Ground-sampling-distance-GSD-in-photogrammetry

the battery. During this time, the weather conditions can change, there might be a new object in the area or it might not be possible to take off.

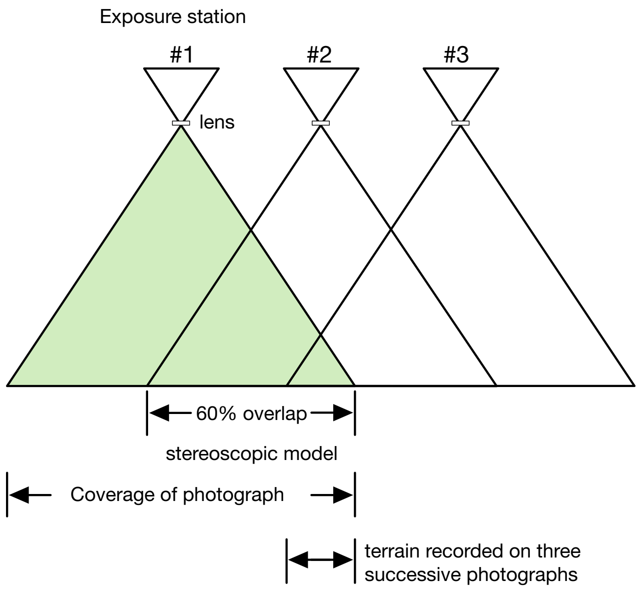


Figure 3.4: An example of a stereoscpic overlap.[3]

The overlap in the photographs taken is crucial. While pictures with 20%-30% overlap can be used to create orthophotography, these create what is called a block of orthophotography. To perceive the depth and volume, the same point must be been in the photographs at least two or three times. Each vertical line from the aerial photograph that overlaps into the next photograph by approximately 60% is referred to as stereoscopic overlap [3] (in the figure 3.4). Using stereoscopic overlap in the adequate GSD grants recognition of terrain in at least 3 successive photographs. This, together with correct GSD creates data good enough for the feature extraction.

For this application, mission planning or camera calibration is an optional step, that is dependent on the UAV, that the surveying operator is using. There are free applications like Pix4Dcapture [18] application, that can use mission planning features. Correct data can be gathered also with manual control over the drone or semi-automatically by recording video files or using automatic photograph capturing features.

[3]Inspired by https://www.edc.uri.edu/nrs/classes/NRS409509/RS/Lectures/409509RSClass2-UnderstandingAirPhotos.pdf

Chapter 4

Data processing algorithms

When adequate post-processing data is acquired for calculating the volume, there must be post-processing of the data must be done first. The data from the flight is usually not perfect as far as the alignment and position. However, this can be calculated only after the feature extraction and reconstruction of the model. There are several ways how to create such a model. In this chapter, I will be discussing the methods used to recreate an object using the data gathered from a drone in a form of a series of photographs, apiece mapping the same object only with a different position.

In section 4.1) I will explain an algorithm used to create a sparse point cloud as well as the camera positions that are then used by an algorithm in section 4.2 which uses the input and produces dense point cloud. This represents the final model of an object in the thesis.

4.1 Structure from motion

In this thesis, the structure from motion (SfM) principle is used to get three-dimensional (3D) data from the drone camera. Traditionally, stereophotogrammetry methods are based on binocular human vision. Consumer-grade drones, however, most do not have such capabilities. Therefore, the depth, volumes, and 3D features must be acquired through a single observing point. Such information can be gathered when either the observer or the object is moving.

Structure from motion is the photogrammetric method of reconstructing 3D structure from its projections into a series of images taken from different viewpoints [20]. SfM is used to get a 3D model from sequences of overlapping 2D images. Even if the dataset is unordered and heterogeneous without any prior knowledge of the image or camera parameters.

Three aspects differ SfM from other photogrammetry methods:

1. Feature extraction can be done automatically in images with different scales, viewing angles, or orientations. This can be useful in small or poorly acquired datasets.

2. No information about camera position or control points is needed, as the equations do not need it. Even though they can be added for better accuracy.

3. Camera calibration can be automatically solved or refined during the process. SfM can thus automatically deliver photogrammetric models without requiring rigorous homogeneity in overlapping images, camera poses, and calibrations.

SfM usually starts with feature extraction and matching followed by geometric verification. The result of strictly SfM is only a sparse point cloud even though the term SfM is commonly referred to as the entire reconstruction workflow [12].

Feature extraction - In this step of the process, each image Ii from the dataset sets local features $Fi = (Xj, fj) \mid j = 1...NFi$, where $Xj \in R2$ is the location of the feature and fj is the feature appearance descriptor itself. These features are invariant under geometric and radiometric changes, so the SfM would be able to recognize them across multiple images. Scale-invariant feature transform (SIFT) algorithm or its derivatives are used in this part to detect, describe and match local features. Binary features can be also used to provide better efficiency.

Matching - This step matches images that see the same part of the scene using the features Fi to detect unique features. Different approaches test a variety of images. Using the similarity metric, features of images are compared to the fj variable of the features. These approaches depend on the complexity, size of the dataset, and computing power of the machine. The result of this step is a set of image pairs with potentially overlapping features $C = Ia, Ib \mid Ia, Ib \in I, a < b$ and $Mab \in Fa \times Fb$ - their feature correspondences.

Geometric verification - Since in the previous step the potentially overlapping features are calculated solely based on their appearance, the features might be at different points in the scene. Therefore, a verification that the features belong to the same scene point is needed. The verification is done through a series of transformations. Homography describes rotating or moving the camera in a planar scene, and epipolar geometry describes the relation between a moving camera through the matrices. If a valid transformation exists, methods like Random sample consensus (detailed in section 4.3) (RANSAC) and or QDESAC for Quasi-degenerate data are used to find outliers in the data.

The camera position, which is used to calculate the projection matrix P_i, the 3D position X_j of the points using triangulation is computed using the matched points in the previous text. The initial triangulation process enhanced by the iterative non-linear optimization is following:

$$E(P, X) = \sum_{i=1}^{n} \sum_{j=1}^{m} d\left(u_{ij}, P_i X_j\right)^2,$$

where $d(x, y)$ denotes the Euclidean distance, n is the number of total images, and m is the number of 3D points. This minimization problem is known as bundle adjustment resulting in a point cloud [11].

4.2 Multi-view stereo

Multi-view stereo (MVS) is a reconstruction problem that creates a complete 3D object model from a collection of images taken from known camera viewpoints [21]. The process consists of iteratively updating the 3D representation with multi-view photometric consistency and regularization optimization. Over the last few years, there has been a big rise in **learning-based MVS** methods [5]. These were popularized by networks working on 2D reconstruction. However, it was not easy to take these principles into 3D. Stereo approaches were lacking contextual geometry knowledge. Then, an approach that made 3D cost volume build around the camera with 2D image features warped in the cost volume, so that 3D CNN can be applied. This approach grants better results than 2D, as using 3D space lowers the distortion of the image. Learning-based MVS, in contrast to non-learning-based MVS, can take advantage of the whole scene, semantic information about the environment, object structures, specularity, illumination, and more, to create a reconstruction more true to the real world. The MVS algorithms can be categorized into different categories. The first category is using a cost function on a 3D volume. This algorithm goes through the volume, computes costs, and reconstructs the volume using only voxels with costs below the threshold. The next category is iterative. At first, it either starts with a large initial volume and shrinks inwards, minimizing the cost function, or it can even locally expand to minimize the energy. The third category is the image-space category, which creates consistent results using a set of depth maps which are later merged into a 3D model. The fourth category extracts feature points and then fits a surface to the reconstructed surface. The final category consists of algorithms using CNN. These algorithms take advantage of semantic feature point extraction. The CNN algorithms try to combat the biggest downside of the MVS algorithms, the need for computing power.

Point-based multi-view stereo network

One of the methods using learning-based MVS is Point-MVSNet. This work directly processes the point cloud, using the PointFlow module to iteratively regress a more dense and still accurate point cloud from the initial input. This network works efficiently, as it adaptively samples potential points in the 3D space, using only valid information near the object, not using the whole 3D space. This can also be useful when only a portion of the whole point cloud is needed for the computation, as it saves a lot of computational power. Point-MVSNet needs the base image sources to boost the dense pixel correspondence quality. At first, the algorithm creates 3-scale feature pyramid $F_i = [F_i^1, F_i^2, F_i^3]$ for images I_i. This pyramid holds features of all the input images together. The features for each point in the scene can be sourced from multi-view feature maps using differentiable unprojection given corresponding camera parameters. a variance-based cost metric is used to aggregate the accurate set of features using multiple sets of views. Pyramid feature at level j, the variance metric for N views is defined as:

$$C^j = \frac{\sum_{i=1}^{N} \left(F_j^i - \overline{F^j}\right)^2}{N}, (j = 1, 2, 3),$$

These features with their normalized point coordinates are then concatenated to form the features for each 3D point. These coordinates are then updated in every iteration and the feature for each point is fetched. This means that the features can be fetched for part of the scene only, focusing on regions of interest rather than the whole scene. This process is called **dynamic feature fetching**.

The input point cloud has only limited accuracy, to improve this, PointFlow is performed. For each point of the point cloud, its displacement to the ground truth is calculated using all the available views. This is not trivial so a **point hypothesis** that attaches different displacements to an unprojected point along the camera direction is created. These displacements are crucial for working with information at different depths. The local neighborhood is important for robust depth prediction, so this work uses ***k*-nearest neighbors** to create the directed graph. This way, the local geometric structure can be used for the feature propagation of points. Using these, edge convolution can be calculated. Flow prediction is using three edge convolution layers to aggregate point features at different scales in the neighborhood. These are local point features and are used to create a probabilistic weighted sum of the displacement among all the point hypotheses using softmax. As this algorithm is also iterative, each iteration is upsampled. The training loss function for this network is:

$$Loss = \sum_{i=0}^{l} \left(\frac{\lambda^{(i)}}{s^{(i)}} \sum_{p \in P_{valid}} \left\| D_{GT}(p) - D^{(i)}(p) \right\|_1 \right),$$

Which is L1 loss measuring the absolute difference between the predicted depth map and the ground truth depth map. P_{valid} represents the valid ground truth pixel, l is the iteration number. The weight $\lambda^{(i)}$ is set to 1.0 in the training [5].

4.3 Optional algorithms

These algorithms are not needed for the 3D reconstruction. However, they are used when the the data has low quality and needs improvement. Random sample consensus can be used to further align the models assuming they do not correspond to the reality.

Random sample consensus

Random sample consensus (RANSAC) is an algorithm, first published by Fishler and Bolles, that is capable of interpreting input data that contains outliers and detecting them. It is an iterative algorithm, which means that it uses a small number of initial data points at first and then increases the number of consistent data points with iterations [10].

Algorithm 1 RANSAC - taken from „Overview of the RANSAC Algorithm“ [8]

```
1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit with a predefined tolerance $\epsilon$.
4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold $\pi$, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of $N$ times).
```

N must be chosen high enough to ensure that at least one of the sets of random samples does not include an outlier with desired probability, p. Let u represent the probability that any selected data point is an inlier and $v = 1 - u$ the probability of observing an outlier. N iterations of the minimum number of points denoted m are required, where

$$1 - p = (1 - u^m)^N,$$

which is equal to:

$$N = \frac{\log(1-p)}{\log(1-(1-v)^m)}.$$

Hausdorff distance

Hausdorff distance is a distance commonly used to calculate a distance between sets of points and a point. This makes it possible to use it as a tool to calculate the distance between two point clouds.

The classic Hausdorff distance is the distance between two sets of edge points, taking the longest distance from one set to the nearest point in the other set. It is very sensitive to outliers. That is why in point cloud geometry it would not be a good choice, even with minimal distortion of the point clouds. That is why a generalized Hausdorff distance was adopted. It measures the distance between two sets of points using K^{th} ranked distance, in contrast to maximum distance.

$$d_{GH-K}(A, B) = \mathrm{per}\, K^{th}_{a \in A} d(a, B),$$

where $\mathrm{per}\, K^{th}_{a \in A}$ is the K^{th} ranked distance such that $(K/N_A) \times 100 = \mathrm{per}\,\%$ and N_A is the total number of points in PC A. For example, the 480^{th} ranked distance in a PC with 600 points is the maximum distance obtained from the $per = (480/600) \times 100 = 80\%$ lowest distance values, after sorting all the distances in ascending order [13].

Chapter 5

Proposed solution

The problem that this thesis aims to solve is, the lack of a direct solution when it comes to tracking the difference between multiple flights. While there are many solutions when it comes to processing the data and creating 3D models or point clouds, there are not many, that focus on the difference between the flights. If there are applications that provide such solutions, they require expensive drone sensors and equipment used on-site as well as processing power. That shapes some of the requirements of this solution to focus on the difference between the flights, as photogrammetry applications are accessible. This application should provide basic functionality for users to sort the data into flights, transform the captured data into a 3D model, let the user choose, which flights he wants to calculate and visualize the result data information that the user easily understands. It is an important part of the solution, to make the calculations in a relatively short time. This is difficult, as it is hardware-related. The goal is for the user not to be dependent on cloud services that require one day and man-made corrections for the results.

5.1 Defining the use case

The typical user of this application wants to survey a small area, usually up to 100 meters squared. The user usually has more different sites that need surveying. He wants to keep track of the changes at each site by flying a commercial-grade drone on a weekly to bi-weekly bases for around 20 minutes, surveying areas from an altitude of 30 meters. After the tracking is finished, the user transfers the data into storage. Once a month, the user wants to evaluate the changes of the sites to write a report or advise some changes.

The application helps him with the process, after the data is obtained, the user can add information for that flight along with the path to the data and start a background process of recreating the 3D model of the scene. Then, when it comes the time for the evaluation, the user simply calculates the difference between desired flights which takes a few seconds, depending on the hardware, and views the result, allowing him to download the results to reference in his work. The application simplifies this workflow by creating a one-stop solution that can be done on-site if needed.

5.2 Application architecture

Based on the available tools I propose an application running in the container environment, with no need to install the third-party libraries and tools. These are part of the image and run the algorithms as follows:

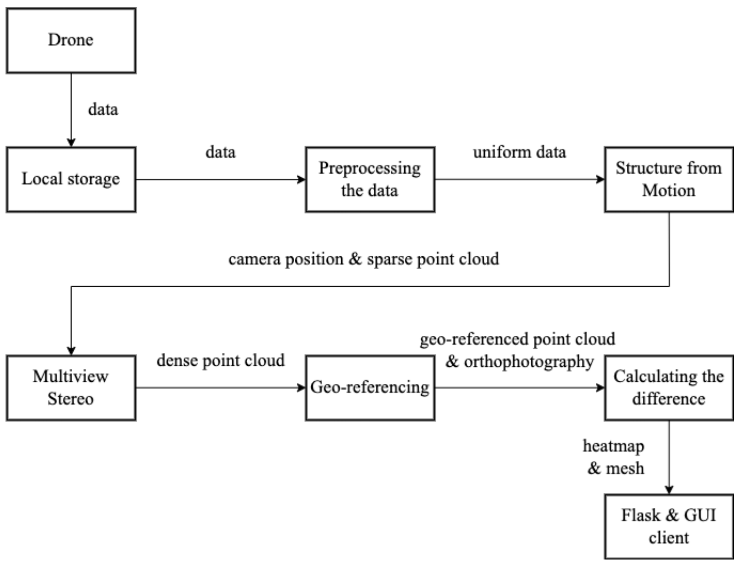


Figure 5.1: A process pipeline of the application.

The Structure from Motion, Multi-view Stereo, and Geo-referencing parts are run outside of the application docker container in the sibling docker container. To communicate the data, both containers have the same local folder mounted, which is the folder with data and also the result folder. This architecture is allowing user to run a following user process. A design of a user process:

1. The user captures drone data in form of a picture or video by flying the drone
2. The user transfers the data into a computer and binds the data folder to the application
3. The user runs the reconstruction task on the data turning it into a dense point cloud and an orthophotography
4. The user repeats the first three steps (flying the drone over the same area) to gather the data at different time
5. The user selects the two flights and runs a computation task to visualize the difference
6. The user uses information about the difference in his work

5.3 3D reconstruction

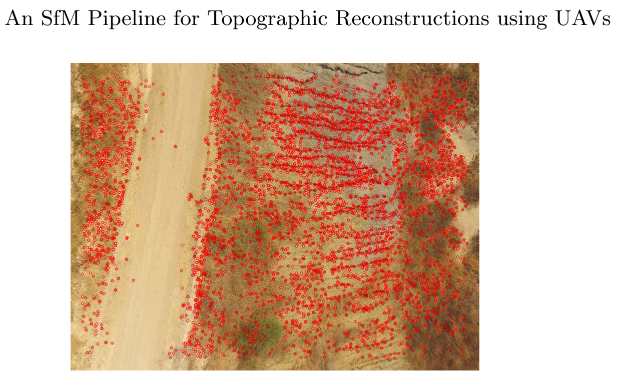
Regarding the structure of the data in section 3.2, drones produce still photographs and video. Acknowledging this, along with the fact, that this application aims to visualize the difference between individual flights, the data needs to be transformed into a 3D model first.

Data preparation

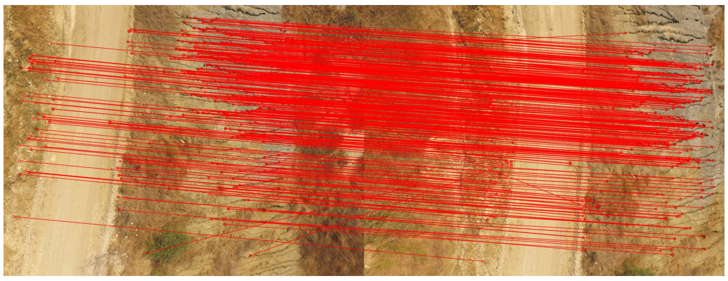
Before any reconstruction can be started, the video files need to be cut into individual still frames. The video files do not have any metadata GPS metadata during the flight, so the additional subtitle file containing GPS coordinates for each frame of the video is needed and used to recreate still frames with all the needed metadata. This creates uniform input across both forms of the data.

Creating a sparse point cloud

To reconstruct a 3D model, reconstruction algorithms need to have multiple views of the same point of interest from different positions. For an object to be reconstructed, it needs to be observable from at least three different photographs [15]. There is no checking for correct data before running the reconstruction algorithms, however, the number of the photographs should be over fifty.



(a) Feature detection using HAHOG algorithm.



(b) Matching of detected features in two photographs of the same scene.

Figure 5.2: An example of feature matching using HAHOG (a) and FLANN (b) algorithms inside OpenSfM.[2]

Barring the optional step of resizing the input data, the first step of reconstruction is the structure from motion algorithm (detailed in section 4.1), which is done by the OpenSfM library inside of the ODM toolkit. At first, the algorithm searches for the pattern in the photographs, that stands out from the surrounding area and therefore is a candidate to stand out in the other photographs as well (as seen in the figure 5.2). These feature points should be unique so that the points are not mistaken for one another. OpenSfM library offers HAHOG (alongside other algorithms for feature point extraction algorithms

[2]Taken from https://jhacsonmeza.github.io/papers/SfM_recons_2018.pdf

- AKAZE, ORB, SIFT), which is the combination of Hessian Affine feature point detector and HOG descriptor to detect feature points regardless of their orientation or scale thanks to being invariant to orientation and scale [14]. After the feature points are found for the whole dataset, there is a link between the images that are used to compute both, the position and orientation of the camera, and a 3D structure of an object.

Creating a dense point cloud

The sparse point cloud is not ideal for volume calculation. It can yield an estimate according to the number of points it detects, but this is not accurate. There are ways, as shown in the previous chapter, to get a finer, more precise point cloud - an MVS (detailed in 4.2) technique. The input in this algorithm is a dataset alongside the camera positions and orientation from the SfM algorithm in the step before. The type of MVS used in the application is multiple depth maps from the OpenMVS library, which creates a depth map for every input image in the dataset. These are then merged into a single scene, thus creating a dense point cloud.

5.4 Calculating the difference

A dense point cloud created using MVS was the last step of 3D reconstruction. All of the following steps will count to calculate the difference between the models.

Geo-referencing a point cloud

Up until this point, the constructed dense point cloud is in the local coordinate system. This is converted using the EXIF (detailed in section 3.2) data from the photographs by using a reference point created by averaging the GPS coordinate of all the photographs and setting it as a center. Other points are then computed.

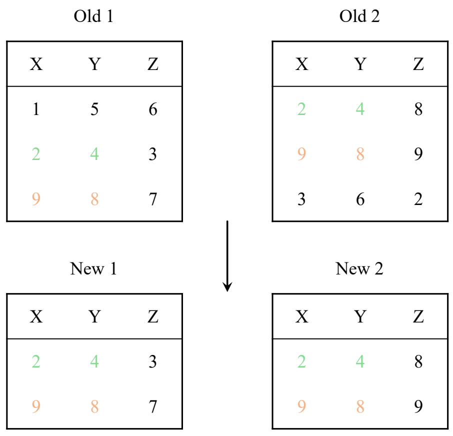
0	1	2	....	13	14	15
“X”,	“Y”,	“Z”,	....	“Red”,	“Green”,	“Blue”

Figure 5.3: An data structure of the geo-referenced point cloud.

The header in figure 5.3 represents a geo-referenced point cloud in the `.csv` format, which has sixteen columns, however, in this application only the columns 0-2 and 13-15 are used, as they represent XYZ coordinates and RGB components of the color. The other non-listed columns are not necessary, as they would provide valuable information if GCPs or specialized sensors were used in the data capture. These columns include intensity, information about the user, et cetera.

Substracting the clouds

With the two geo-referenced point clouds, assuming they are aligned, it can proceed to calculate the size difference. If they are not, a RANSAC algorithm can be used to align the clouds, which takes away the geo-reference out of the point cloud as it manipulates its coordinates. A mesh, created by merging the different colored point clouds representing the two flights is one of the results, that helps to better understand the difference in volume. The volume created by this mesh can also represent an approximation of the real change in the volume. Another way of displaying the changes is by putting the two flight point clouds into an **intersection** by their X and Y coordinates (as seen in figure 5.4). All of the points that do not have their X and Y pair in the other point cloud are deleted. This produces interesting statistics; the percentage of the points that have been changed between the two flights, and the average difference in the altitude. This data allows for the creation of a **heatmap**, that keeps the same X and Y coordinates, but the Z coordinate marks the difference in the altitude between the flights. There are more colors on the map, each representing a range of altitude values.


Old 1

X	Y	Z
1	5	6
2	4	3
9	8	7

Old 2

X	Y	Z
2	4	8
9	8	9
3	6	2

New 1

X	Y	Z
2	4	3
9	8	7

New 2

X	Y	Z
2	4	8
9	8	9

Figure 5.4: An example of intersection between the point cloud coordinates.

5.5 Graphical user interface

The graphical user interface (GUI) for the application does not have to be complicated. The basic user needs are to be able to add a flight, be able to differentiate between flights, recreate models from the flight data, compare the differences between the data and view the results in compendious form.

At first sight, the user is greeted with a message saying that there are no flights in the database prompting him to add the first flight by filling out the form for a new flight. The form contains inputs for the data path to the folder, where flight data is stored, the location of the flight, and an optional note. After the first flight is added a basic layout of the application is shown. On the left, there is a list of flights containing all added flights. They can be sorted or filtered by the date or location. Each separate flight has a checkbox next to it. After clicking on the flight name, a detail of the flight is shown on the main, right, side of the screen, with the ability for the user to update flight data or delete the flight. There is also a „Prepare“ button, that starts the background process that converts the input data if needed, and then recreates a 3D model of the flight. When a model is created, its orthophotography is shown in the flight detail.

Underneath the list, in the side panel, there is a „Compute“ button alongside „Plus“ and a „Gear“ button. The „Plus“ button allows the user to add another flight in the same fashion as in the initial flight. The „Gear“ button provides the change of settings for calculating the difference. This includes the quality, input image size, and an option to change the name of the default ODM container name. Ultimately, provided that exactly two flights are checked, the „Compute“ button computes a difference between two flights.

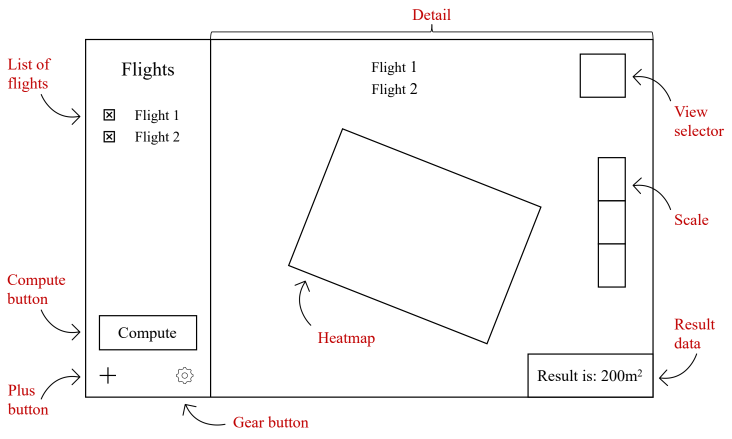


Figure 5.5: The mockup of the graphical user interface.

This computes the difference and after a few seconds, the user can see a result on the main side of the screen.

There are several ways of displaying the results. One of which is a heatmap. It takes a 2D aerial view of the point cloud and changes the color of the individual pictures. Each point is represented by the color from a color palette, portraying the change in that particular point between two flights. When the user hovers a mouse over the heatmap point, a text tooltip next to his cursor apprises him the height difference in meters. In the bottom right corner, there is information about how many points stayed at the same altitude, and how many have changed as well as an approximate estimate of the difference in volume. In the top right corner, there is a button used for changing the view from a heatmap to a mesh representing the change of a volume of a difference between the two flights. These views are useful to visually inspect the differences, as the numbers do not speak about the precise position of the differences.

Chapter 6

Implementation and experiments

The application is built as a toolkit with a webpage interface, with the details regarding functionality documented in chapter 5. Because the tools used for the computation are run in form of a Docker[1] container, the application is also run as a Docker container. This chapter talks about the details of the implementation and the challenges that it solves.

6.1 The tools used

This application workflow consists of numerous steps, which are not trivial, thus a number of specific tools and libraries is used to used in each step of the workflow. This section talks about tools used for computation with the drone data and does not include tools used to deploy a client-server interface or any other tools. The most used tool in the application is:

- **OpenDroneMap**[2] (ODM) - an open-source toolkit run in the docker container that is used for 3D reconstruction, encapsulating multiple libraries into one image.
- **OpenSfM**[3] - an open-source library used for the feature extraction and matching during the Structure from Motion algorithm.
- **OpenMVS**[4] - an open-source library used to create a dense point cloud during the Multi-view stereo algorithm.
- **OpenCV**[5] - an open-source library used to slice the input video into frames used for reconstruction
- **Open3D**[6] - an open-source library used to handle the point clouds and meshes

Other worthy libraries, that are used in the reconstruction are:

- **NumPy**[7] - used to handle fast calculations with large point clouds
- **SciPy**[8] - used for calculation of Delaunay triangulation

[1] https://www.docker.com
[2] https://www.opendronemap.org
[3] https://opensfm.org
[4] https://github.com/cdcseacave/openMVS
[5] https://opencv.org
[6] http://www.open3d.org
[7] https://numpy.org
[8] https://scipy.org

6.2 Web application

The application is built using the Flask[9] web microframework server. Flask was chosen because it is lightweight, it is a Python[10] framework, which directly supports the libraries used for computation, and it supports extensions to be easily run in a Docker container.

The web application runs the SQLite3[11] database, which is implemented primarily to keep track of the various flights. The Flight table in the database contains information about the class of the same name, Flight (as seen in the figure 6.1). The `data_route` is the input, but also the output route for the final results, meaning the whole result structure will be built in this folder. This is due to the privilege errors when running the Docker application.

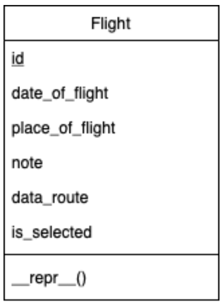


Figure 6.1: Object oriented class Flight.

To run the application GUI from the Docker container, and to enable threading for the background tasks that are done by the third party tools, a Web Server Gateway Interface - WSGI[12] is run. I used uWSGI to create an uwsgi Unix socket, that communicates between the web server and the Flask application. Alongside uwsgi runs Nginx[13], supplying a fast web server for communicating between the socket and the client (as seen in the figure 6.2). The server is configured to kill all the tasks if they reach the time of over 5 hours. This figure was estimated after testing multiple datasets, which the majority took approximately 40 minutes with the exception of 2 hours in the extreme case.

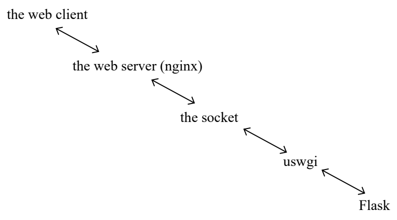


Figure 6.2: The server-client architecture.

When the user calls the compute function, firstly the video input is handled with the processor-oriented cv2 python library, which supports DJI video codecs. Using the Video I/O module, specifically changing the parameters in `cv2.VideoCapture` method, every 5

[9]https://flask.palletsprojects.com/en/2.1.x/
[10]https://www.python.org
[11]https://www.sqlite.org
[12]https://wsgi.readthedocs.io/en/latest/what.html
[13]https://www.nginx.com

seconds a single video still is taken and using `cv2.imwrite` method saved as photography to a disk.

A dockerfile was written for the application to create a new Docker image, which already contains all the important requirements and runs the application. This makes it easy for the end-user to initialize the application, as he needs to download the ODM image, and the application image and mount the application correctly to have access to all of the tools. This is possible thanks to the concept of sibling docker containers [6], which means, that the Docker command-line interface (CLI) is connected to a server via socket, which is representing a file on the filesystem. When the mentioned file is mounted as a volume inside a docker container and the Docker CLI is run there, it connects to the Docker Engine on the host. The CLI is connected to the Engine via the Docker APIs on a dedicated socket, most of the time located on the path `/var/run/docker.sock`. When the process inside the container sends requests to the Docker Engine, it connects to the inner socket, that is the Engine on the host, that is listening. Hence the request to run a new container is sent to the host, thus creating a sibling container.

The ability to create a sibling container, combined with the use of the Nginx and uwsgi enables the application to run a thread, that prepares the dataset by turning it into a point cloud. The thread calls the ODM container with a following command: `docker run -ti -rm -v path:/datasets opendronemap/odm -project-path /datasets dataset pc-csv`. This command will run the 3D reconstruction using OpenSfM, OpenMVS, and other libraries built into the ODM toolkit [17]. There are more, optional arguments, that can be used for more precise results. These arguments are:

- `feature-quality (ultra | high | medium | low | lowest)` - sets the feature extraction quality. Higher quality generates better features, but requires more memory and takes longer, the default is `high`
- `gps-accuracy <positive float>` - sets a value in meters for the GPS Dilution of Precision (DOP) information for all images. The default is `10`
- `ignore-gsd` - ignores the Ground Sampling Distance (GSD). Since GSD is an estimate, sometimes ignoring it can result in slightly better image output quality. The default is `false`
- `pc-quality (ultra | high | medium | low | lowest)` - sets the point cloud quality. Higher quality generates better, denser point clouds, but requires more memory and takes longer. Each step up in quality increases processing time roughly by a factor of 4x. The default is `medium`

The command above directly outputs the geo-referenced dense point cloud together with the orthophotography in the PNG and GeoTiff format. The logs of the preparation are in the `/tmp` folder. The results are in the structure as follows:

The process of the computation with point clouds uses the NumPy library to optimize the calculations, as the point clouds have tens of millions of points, which yields them usually more than a gigabyte in the size. Since the coordinates are in the UTM format, which uses a floating-point with two-digit precision, the individual points are loaded as `np.float32`. These are then always multiplied by 100, as the calculations with a floating-point were costly and not accurate.

The point clouds are stored in a form of Open3D point cloud `geometry.PointCloud()`. This allowed me to easily integrate the SciPy library, downsample the point clouds, and use

```
project/
├── odm_georeferencing/
│   ├── odm_georeferenced_model.csv     # XYZ format point cloud
│   ├── odm_georeferencing_log.txt      # Georeferencing log
│   └── odm_georeferencing_utm_log.txt  # Log for the~extract_utm portion
└── odm_orthophoto/
    ├── odm_orthophoto.png              # Orthophoto image (no coordinates)
    ├── odm_orthophoto.tif              # Orthophoto GeoTiff
    ├── odm_orthophoto_log.txt          # Log file
    └── gdal_translate_log.txt          # Log for georeferencing the~png file
```

Figure 6.3: The structure of the output data.

its `spatial.Delaunay()` for the triangulation of the point cloud. This results in a mesh illustrating the difference between the point clouds.

The 2D graphs generated by the application are made by calculating the point cloud distance using the library Matplotlib[14]. These could be turned to a heatmap by adding a color spectrum and dyeing the altitude the color from the appropriate range.

6.3 Acquiring own data

One of the biggest challenges of this thesis was to gather a relevant dataset to test the functionality and theories. The very first dataset used to recreate a model of a scene is the one of a Qatar desert taken by DJI Phantom 4. It was relevant when using photogrammetry methods to create orthophotography of a scene, regardless, a point cloud was not usable due to the small height differences and uniformity of the colors.

This led to obtaining own data by flying a drone. Nonetheless, the drone was the DJI Spark (detailed in section 2.2) that as mentioned above was not meant to be used for photogrammetry due to its low flight time and mounting of the camera. This, however, made for the best scenario as it represented the user with the lowest consumer tier drone from DJI.

With the drone, I created a dataset (in figure 6.4) that would represent a good case scenario. This dataset contains two flights shot back to back after each other on a lane, where one of the flights contains a car park in the middle and the other one does not. In the dataset, the lighting conditions were good, and there was no movement on the ground during the flights.

The dataset possesses more than 300 pictures per flight taken from an altitude of 30 meters with a minimal overlap of 60%. The differences in overlapping are due to the manual way of taking the pictures, as there were no photogrammetry mission planning applications for the Spark. The data from this set could not be used, as only the orthophotography could be created with reconstruction lacking enough feature points - possibly due to the structure of the snow.

The next dataset also incorporates two flights, this time with over 400 photographs in each flight, one with the car present and the other one without a car. The setting captures

[14]https://matplotlib.org

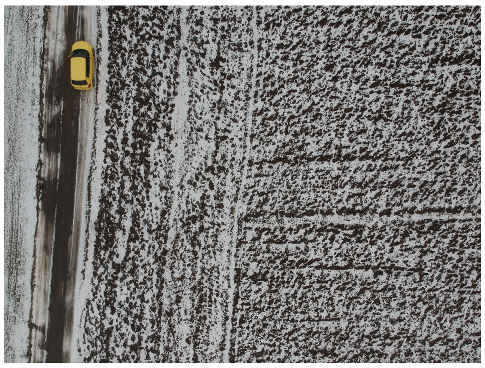


Figure 6.4: An example of the input photograph from the dataset with a lack of feature points.

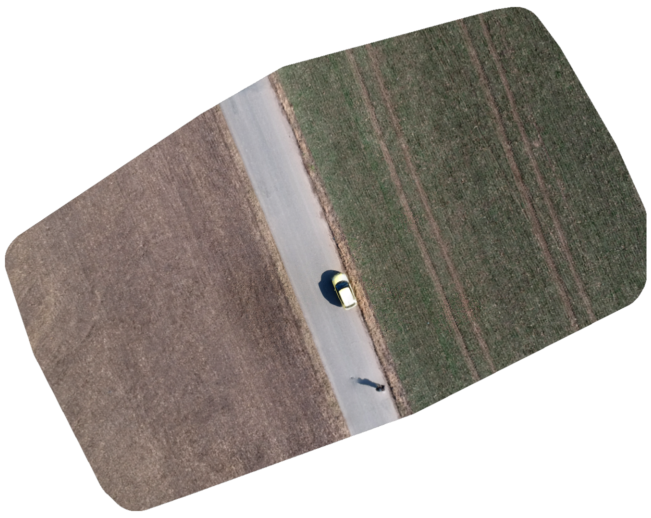


Figure 6.5: An orthophotography of the field dataset.

a road with a field on both sides of the road. Contrary to the previous dataset, this one possesses photographs from three different heights; the first is from an altitude of 30 meters, and the following from an altitude of 50 meters, with the last part containing drone shots orbiting around the car at a distance of 1-2 meters. In the dataset without a car, this portion is dedicated to the close-up photographs on the side of the road. Again, this dataset has a minimal overlap of 60%, due to the manual taking of the photographs. The results are not limited only to the orthophotography, this time there are enough feature points to be extracted to reconstruct a 3D model. The problem was, however, that since the road is straight, there are no curves and there is a uniform field on the side of the road, therefore no accurate alignment, when it comes to aligning the two models before calculating the volume as explained in the section 5.4.

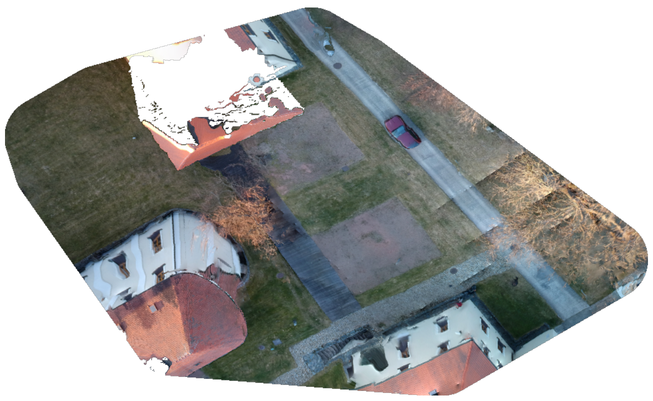


Figure 6.6: An orthophotography of the FIT dataset.

The last and final dataset was provided by my supervisor, which I am thankful for. This time, it contains three different flights in the same area, ranging from 63 to 84 photos, with an altitude of 30 meters, again with a car, now in two positions, and the third flight did not contain a car. What is more, there were two drones flying over each scene. The one was DJI Spark, which was photographing the scene, and the second one was DJI Maverick which captured a video flying over a similar perimeter as the Spark. Defiant to the previous dataset, this one was set near a faculty campus, in an environment with a number of different, to create a better alignment. This dataset will be referred to as FIT dataset.

6.4 Experiments with the quality

The application has optional arguments (as seen in section 6.2) that change the number of iterations and other parameters in the reconstruction algorithms to provide a finer result. These have a radical impact on the time and memory needed to finish the task.

According to the Steam hardware survey [22], only 14% of the users have more than 16GB of RAM in their computer. The majority of the users, 50,59% have 16GB of RAM. The application was implemented on Apple Mac Mini with an ARM M1 processor and 8GB of RAM, and no dedicated graphics card. Using lower-end hardware did not allow me to use the medium setting for the feature point extraction as well as the point cloud quality. Trying to run higher settings resulted in errors that no memory is available. This gave me an idea, to test, whether or not a higher-end computer grants better results.

With the FIT dataset on a medium preset, I could reconstruct the dense point cloud with a point count of 404,453 points in 6:57 for the first flight and 631,269 in 7:16 for the second flight. The higher-end computer is represented by a windows machine running Intel core i7 7700K, with Nvidia GTX 1050 and 32GB of RAM. The same FIT dataset run with ultra settings reconstructed the dense point cloud with a point count of 10,458,124 in 43:33 for the first flight and 10,452,858 in 59:57 for the second flight. The data in this experiment were not aligned, to establish an accurate representation, using unmodified data.

Both pairs of the point clouds were subtracted to illustrate the difference between them. The resulting point clouds were put into two-dimensional graphs. The X axis demonstrates a difference in the altitude between the first flight and the second flight. The Y axis denotes the number of points with the same difference in the altitude.

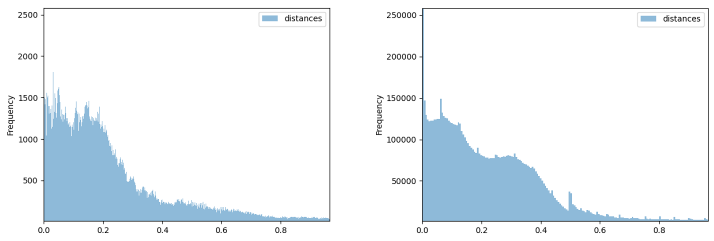


Figure 6.7: The difference between two flights with medium settings (left) and ultra settings (right).

The difference is noticeable, as seen in the graph 6.7. Both of the results look similar, with the higher-end dataset showing finer and more frequent points between the two flights. The graph on the right displays a spike between two flights around the 0.5 mark. This is likely due to a tree or bush particles being present in one of the flights and not in the other one. Overall, the more precise point cloud represents the real scene more accurately, which was mostly thanks to its better alignment.

My reasoning for the dissimilarity is that the geo-referencing is done after the dense point cloud is constructed, which indicates that the less dense point clouds end up being aligned geo-referenced slightly worse.

6.5 Function testing

During the pipeline recreation, the testing was done using `assert` functions inside of the source code insted of a unit test script. This was crucial, as it revealed important mistakes, when it comes to 3D recreation. The testing of the entire pipeline was done manually at first since there was a need to find working tools. Every output was inspected by reviewing its source code, attempting to open it in a relevant application, or modifying the data to have the proper format. For example, the input of the MVS algorithm requires to be images that were used as input in the previous step as well as the JSON file containing the position and orientation of the cameras.

Since the available dataset for the application is small, there was no complex testing on an extensive number of datasets, rather the application pipeline was tested with only a few different sized data. For the smaller point clouds cut regions of the point clouds were used to demonstrate if the app was still usable with the lower quality data, and if not, I manually checked if the error outputs were accurate. For the bigger data, a FIT dataset was used with the ultra settings containing 10,452,858 points. The outcome of this test found that the NumPy function `isin` with multiple queries did not work and the code had to be adjusted accordingly.

Testing of the proper alignment was done manually by opening the GeoTiff orthophotographs and pasting them into the GID application to see if the alignment is correct. The outcomes uncovered that the optional resizing of the input data correlates with the alignment. The resized data was misaligned compared to the unmodified data. The correctness of resulted GeoTiff orthophotographs was tested also by using the GID application and testing if the scale matched the one in the photograph.

The integration testing was improved by running the application as a docker image. Allowing to conduct interation testing once and deploy the application, while ensuring the functionality.

Chapter 7

Conclusion

This work aimed to implement user processes and a tool with a graphical interface allowing for the detection of changes in the scene captured by the drone at different times. Firstly, research on existing solutions of drone mapping and three-dimensional reconstruction was done. The research resulted in dividing the problem into smaller subproblems and identifying them.

Further investigation achieved understanding regarding specific algorithms and libraries that implement them. An initial pipeline was created, trying to reconstruct and visualize the scene manually, to encounter practical knowledge about the nuisances. To successfully recreate a scene, I had to understand and preprocess the data, to use them in the algorithms.

After succeeding in the reconstruction I gained access to a drone to create a usable dataset since there were not any available. The first automatic pipeline attempt was produced after succeeding in generating my data. The automatic tool was command-line based and needed to install the libraries to the user's computer manually. This created the final vision of the application architecture. The application is bundled in a container and can be distributed easily. I created a graphical user interface, to make the app aimed at a wider audience.

One of the main contributions of the application is, in my opinion, the complexity of the solution. It makes the solution a one-stop application for both, recreating a three-dimensional scene and also the ability to survey and monitor changes over time.

If I had my actual knowledge of the subject at the start, I would have implemented the application as an extension of the OpenDroneMap project, because it is open-source and already has most of the libraries used in this thesis bundled inside.

The result of the thesis is an application running a pipeline that uses state-of-the- art reconstruction algorithms, allowing the user to upload, organize, and reconstruct the data into three-dimensional models and view the difference in volume between the flights at different times.

Bibliography

[1] Anon.. *Civil drones (unmanned aircraft)* [online]. The European Authority for aviation safety [cit. 2022-01-03]. Available at: https://www.easa.europa.eu/domains/civil-drones.

[2] Anon.. *Tunable Diode Laser Absorption Spectroscopy* [online]. Zurich Instruments [cit. 2022-01-02]. Available at: https://www.zhinst.com/europe/en/applications/optics-photonics/tunable-diode-laser-absorption-spectroscopy.

[3] Anon.. *Understanding of Air Photos Photogrammetry* [online]. University of Rhode Island [cit. 2022-05-14]. Available at: https://www.edc.uri.edu/nrs/classes/NRS409509/RS/Lectures/409509RSClass2-UnderstandingAirPhotos.pdf.

[4] Anon.. *What is GPS?* [online]. Geotab, may 2020. Available at: https://www.geotab.com/blog/what-is-gps/.

[5] Chen, R., Han, S., Xu, J. and Su, H. Point-Based Multi-View Stereo Network. *CoRR* [online]. arXiv. 2019, abs/1908.04422. DOI: 10.48550/ARXIV.1908.04422. Available at: http://arxiv.org/abs/1908.04422.

[6] Colangelo, A. *Sibling Docker Containers* [online]. Medium, july 2019 [cit. 2021-05-13]. Available at: https://medium.com/@andreacolangelo/sibling-docker-container-2e664858f87a.

[7] Corrigan, F. *Drone Gyro Stabilization, IMU And Flight Controllers Explained* [online]. DroneZon, may 2020 [cit. 2022-01-02]. Available at: https://www.dronezon.com/learn-about-drones-quadcopters/three-and-six-axis-gyro-stabilized-drones/.

[8] Derpanis, K. G. *Overview of the RANSAC Algorithm* [online]. Rmozone, may 2010 [cit. 2021-12-12]. Available at: http://rmozone.com/snapshots/2015/07/cdg-room-refs/ransac.pdf.

[9] DJI, S. *DJI GO 4 Manual: The Pilot's Handbook - DJI Guides* [online]. DJI, october 2017 [cit. 2022-11-02]. Available at: https://store.dji.com/guides/dji-go-4-manual/.

[10] Fischler, M. A. and Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. june 1981, vol. 24, no. 6, p. 381–395. DOI: 10.1145/358669.358692. ISSN 0001-0782. Available at: https://doi.org/10.1145/358669.358692.

[11] FURUKAWA, Y. and PONCE, J. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE transactions on pattern analysis and machine intelligence*. august 2010, vol. 32, p. 1362–76. DOI: 10.1109/TPAMI.2009.161.

[12] IGLHAUT, J., CABO, C., PULITI, S., PIERMATTEI, L., O'CONNOR, J. et al. Structure from Motion Photogrammetry in Forestry: a Review. *Current Forestry Reports* [online]. Springer. september 2019, vol. 5, no. 3, p. 155–168. DOI: 10.1007/s40725-019-00094-3. ISSN 2198-6436. Available at: https://doi.org/10.1007/s40725-019-00094-3.

[13] JAVAHERI, A., BRITES, C., PEREIRA, F. and ASCENSO, J. A Generalized Hausdorff Distance Based Quality Metric for Point Cloud Geometry. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 2020, p. 1–6. DOI: 10.1109/QoMEX48832.2020.9123087.

[14] LINDEBERG, T. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*. september 1998, vol. 30, p. 77–116. DOI: 10.1023/A:1008045108935.

[15] MEZA, J., MARRUGO, A., SIERRA, E., GUERRERO, M., MENESES, J. et al. A Structure-from-Motion Pipeline for Topographic Reconstructions Using Unmanned Aerial Vehicles and Open Source Software: 13th Colombian Conference, CCC 2018, Cartagena, Colombia, September 26–28, 2018, Proceedings. In:. January 2018, p. 213–225. DOI: 10.1007/978-3-319-98998-3_17. ISBN 978-3-319-98997-6.

[16] OCEANIC, N. and ADMINISTRATION, A. *What is lidar?* [online]. National Oceanic and Atmospheric Administration, february 2021. Available at: https://oceanservice.noaa.gov/facts/lidar.html.

[17] ODM. *OpenDroneMap's documentation (version 2.8.4)*. Available at: https://docs.opendronemap.org.

[18] PIX4D. *Pix4Dcapture (version 4.13.1)*. Available at: https://www.pix4d.com/product/pix4dcapture.

[19] PIX4D, S. *Ground sampling distance (GSD) in photogrammetry* [online]. Pix4D [cit. 2022-01-02]. Available at: https://support.pix4d.com/hc/en-us/articles/202559809-Ground-sampling-distance-GSD-in-photogrammetry.

[20] SCHÖNBERGER, J. L. and FRAHM, J.-M. Structure-from-Motion Revisited. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. June 2016, p. 4104–4113 [cit. 2021-10-24]. DOI: 10.1109/CVPR.2016.445. Available at: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Schonberger_Structure-From-Motion_Revisited_CVPR_2016_paper.pdf.

[21] SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D. and SZELISKI, R. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: IEEE. *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. 2006, vol. 1, p. 519–528. DOI: 10.1109/CVPR.2006.19. Available at: https://vision.middlebury.edu/mview/seitz_mview_cvpr06.pdf.

[22] Steam. *Steam Hardware & Software Survey: April 2022* [online]. Available at: https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam?platform=combined.

[23] Tarin, M. *Thermal Infrared Imaging explained!* [online]. MoviTHERM [cit. 2022-01-02]. Available at: https://movitherm.com/knowledgebase/thermal-infrared-imaging-explained/.

[24] Wagner, M. September 2014. Available at: https://opil.ouplaw.com/view/10.1093/law:epil/9780199231690/law-9780199231690-e2133.

[25] Winkler, C. *How Many Sensors are in a Drone, And What do they Do?* [online]. Fierce Electronics, july 2016 [cit. 2022-01-02]. Available at: https://www.fierceelectronics.com/components/how-many-sensors-are-a-drone-and-what-do-they-do.

Appendix A

Storage media structure

The file structure in the attached storage media:

- **`src`** – a folder containing the source files of the application
- **`dataset`** – a folder containing the FIT dataset
- **`latex`** – a folder containing the source code of the master's thesis
- **`media`** – a folder containing a poster and a video
- **`README.md`** – a file containing the storage media structure