Enhanced Reconnaissance using Deep Learning and Computer Vision

Project Report submitted in partial fulfillment

Of

Bachelor of Technology

In

Electronics and Telecommunications Engineering

By

Aniruddh Chandratre (70061016009)

Kunal Amin (70061016002)

Jahnavi Doneria (70061016012)

Under the supervision of

Dr. Vaishali Kulkarni

Associate Dean, MPSTME

SVKM's NMIMS University

(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY

MANAGEMENT & ENGINEERING

Vile Parle (W), Mumbai-56

2019-20

CERTIFICATE

This is to certify that the project entitled " **Enhanced Reconnaissance using Deep Learning and Computer Vision**" has been done by **Mr. Aniruddh Chandratre** under my guidance and supervision. The report has been submitted for Phase-1 evaluation of the degree of Bachelor of Technology in Electronics and Telecommunication of Mukesh Patel School of Technology Management and Engineering, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

Dr. Vaishali Kulkarni	Examiner
(Internal Guide)	(External)
Date :	
Place : Mumbai,	Dr. Manoj Sankhe
India	(HOD)

Abstract

A drone survey refers to the use of a drone, or unmanned aerial vehicle (UAV), to capture aerial data with downward-facing sensors, such as RGB or multispectral cameras. During a drone survey with an RGB camera, a drone captures multiple images from multiple geo-coordinates at different angles. For a highly accurate aerial mapping mission, the drone is flown using an autopilot mission at a fixed height. This thesis proposes dedicated hardware, an autonomous unmanned aerial vehicle and software capable of generating orthomosaics, 3d models, digital surface models & digital terrain models. Unlike manned aircraft or satellite imagery, drones can fly at a much lower altitude, making the generation of high-resolution, high-accuracy data, much faster, less expensive and independent of atmospheric conditions such as cloud cover. The work done in this thesis focuses on a military application of aerial mapping - Reconnaissance. Current methods of military reconnaissance is not safe as it involves sending personnel behind enemy lines to gather information. This can potentially lead to loss of life. The work adds support for two additional pipelines over aerial mapping, (1) face clustering & (2) weapon detection and recognition to accommodate for military reconnaissance requirements.

In this thesis, we propose an autonomous UAV and an image processing toolbox that is suitable for aerial mapping in military reconnaissance missions.

Acknowledgement

The project has been highly intricate. However, with constant support from faculty and friends, the project was made possible. It would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them. We are highly indebted to Dr. Vaishali Kulkarni for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project. We would like to express our gratitude towards Mr. Amey Raut and Mr. Dattatray Sawant for their kind cooperation and encouragement which helped us with any electronic parts we required in the prototyping of this project.

Thank you,
Aniruddh Chandratre (70061016009),
Kunal Amin (70061016002),
Jahnavi Doneria (70061016012)

Contents

List of Figures

List of Tables

List of Listings

xi

List of Acronyms

UAV	Unmanned Aerial Vehicle
MMI	Mission Management Interface
RPN	Remote Processing Node
GPS	Global Positioning System
DEM	Digital Elevation Model
DSM	Digital Surface Model
DTM	Digital Terrain Model
REST	Representational state transfer
IMU	Inertial Measurement Unit
FC	Flight Controller
ROS	Robot Operating System
MAVLink	Micro Air Vehicle Link
RTL	Return-to-Launch
QoS	Quality of Service
TCP	Transmission Control Protocol
GIS	Geographic Information System
SLAM	Simultaneous Localisation and Mapping
SfM	Structure from Motion

xii

MVS Multi View Stereo

GSD Ground Sampling Distance

xiii

1 Introduction

This chapter introduces a general notion of the work – the problem statement, background, motivation and scope. Furthermore, this chapter lays down the organisation of the manuscript.

1.1 Background of the project topic

Before conducting any military mission, it is important to perform reconnaissance in order to have a proper understanding of the base (site for attack) and the artilleries used by the enemy. Attack sites are often dangerous for humans to travel to or inaccessible because of the terrain, and can often lead to loss of life and information. This work aims to provide an autonomous systematic pipeline for enhanced reconnaissance which involves a step by step extraction of crucial information from images. We make use of an autonomous surveying drone to capture high resolution images of the target area. The proposed mapping software generates high-resolution orthomosaics and detailed 3D models of areas where low-quality, outdated or even no data, are available. They thus enable high-accuracy cadastral maps to be produced quickly and easily, even in complex or difficult to access environments.

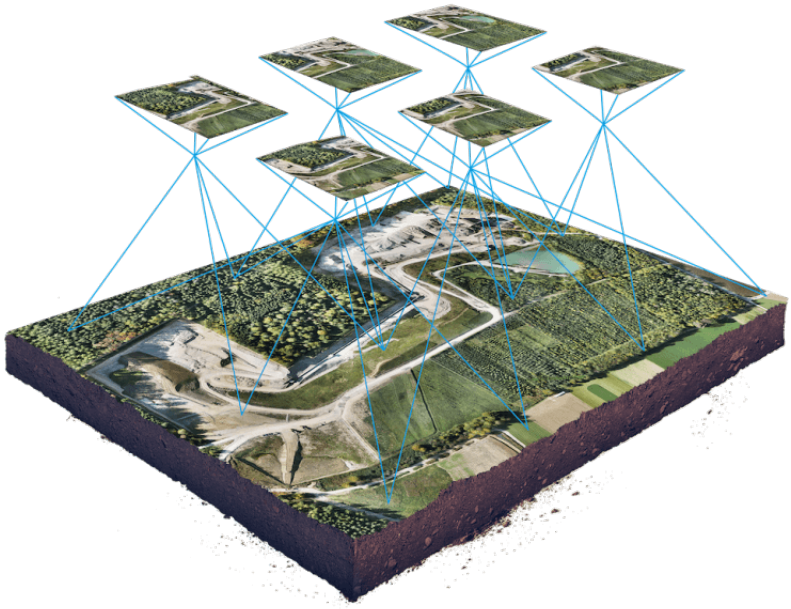

Fig. 1.1.: Aerial Mapping

1.2 Problem Statement

The intent of this project is to build dedicated hardware and software capable of auonomous aerial mapping and reconnaissance. To achieve this goal, the hardware must capable of completing the following objectives.

1. Autonomous flight based on Global Positioning System (GPS) waypoints
2. Autonomous take-off and land
3. Return-to-launch (failsafe)
4. Gyroscopic image stabilisation
5. Geo-tagged image acquisition & storage

On the other hand, the software must provide functionality for the following :

1. 3D reconstruction (point cloud & mesh)

2. Orthophoto generation
3. Digital Surface Model (DSM) generation
4. Digital Terrain Model (DTM) generation
5. Face clustering
6. Weapon recognition
7. Mission Management (Creation & Modification interface)
8. Flight Planner
9. Mission browser (For viewing information about old missions)
10. Orthophoto, DSM, DTM visualiser
11. 3d model visualiser
12. AR playground
13. Face & weapons database viewer

1.3 Motivation and scope of the report

Our main goal is to try to prevent the loss of human life by finding a safer and faster alternative to gather intelligence behind the enemy lines. Using an autonomous UAV provides a much safer alternative to sending personnel for gathering data. Using a drone instead for aerial mapping provides added advantages :

1. Capturing topographic data with a drone is up to five times faster than with land-based methods and requires less manpower. With PPK geo-tagging, you also save time, as placing numerous GCPs is no longer necessary.

2. Total stations only measure individual points. One drone flight produces thousands of measurements, which can be represented in different formats (orthomosaic, point cloud, DTM, DSM, contour lines, etc). Each pixel of the produced map or point of the 3D model contains 3D geo-data.

3. An aerial mapping drone can take off and fly almost anywhere. You are no longer limited by unreachable areas, unsafe steep slopes or harsh terrain unsuitable for traditional measuring tools. You do not need to close down highways or train tracks.

1.4 Organization of report

The project has been divided into six chapters and each one conveys the following information :

1. **Chapter 1 - Introduction** : This chapter introduces a general notion of the work – the problem statement, background, motivation and scope. Furthermore, this chapter lays down the organisation of the manuscript.

2. **Chapter 2 - Literature Survey** : This chapter breaks down the monolithic problem statement into multiple objectives and provides an introduction to related work with respect to each objective.

3. **Chapter 3 - Methodology & Implementation** : This chapter first introduces a holistic overview of the complete pipeline, outlines the functions of each sub-system, and provides an understanding of how everything is connected. The chapter also explains the in-depth working of each node within the pipeline.

4. **Chapter 4 - Results & Analysis** : This chapter shows the results of our implementation by demonstrating the features of application. Additionally, it

presents some bench marking results and performance numbers that can be used to prove the usability of our system.

5. **Chapter 5 - Advantages, Limitations and Applications** : This chapter discusses the advantages of using our system over any other. It also describes the limitations of our software and how they can be overcome. This chapter also provides an overview about the different application areas for project.

6. **Chapter 6 - Conclusion and Future scope** : This chapter concludes the thesis and outlines areas where the project can be improved in the future.

2 Literature Survey

This chapter breaks down the monolithic problem statement into multiple objectives and provides an introduction to related work with respect to each objective.

2.1 Understanding the problem statement

Section 1.2 provides a general understanding of the problem statement. However, as a whole, the problem statement is compounded. In order to to understand and solve the problem statement, the monolithic problem has to be broken down into a nexus of smaller, achievable objectives.

The overall system comprises of three sub-systems – (1) Unmanned Aerial Vehicle (UAV), (2) Mission Management Interface (MMI) & (3) Remote Processing Node (RPN). Each of the sub-system is strategically designed to provide the functionalities stated below.

2.1.1 Functions of Unmanned Aerial Vehicle (UAV)

The UAV sub-system is a quadcopter running a customised version of the PX4 autopilot stack. As defined in the section 1.2, the UAV must be capable of autonomous take-off, land and flight. Along with this, the problem statement also defines that the UAV must have an on-board high resolution camera capable of capturing geo-tagged images on

6

manual and automatic electronic trigger. The UAV should be capable of the following functions.

1. Autonomous flight based on GPS waypoints
2. Autonomous take-off and land
3. Return-to-launch (failsafe)
4. Gyroscopic image stabilisation
5. Geo-tagged image acquisition & storage

2.1.2 Functions of Mission Management Interface(MMI)

The MMI sub-system defines a web based interface for creation, modification and archival of missions. The MMI extends its functionality by providing a set of visualisation tools for the products generated by the RPN. Main functions of the MMI are as follows.

1. Mission Management (Creation & Modification interface)
2. Flight Planner
3. Mission browser (For viewing information about old missions)
4. Orthophoto, DSM, DTM visualiser
5. 3d model visualiser
6. AR playground
7. Face & weapons database viewer

2.1.3 Functions of Remote Processing Node (RPN)

The RPN sub-system defines a Representational state transfer (REST) API build around three main nodes - Aerial Mapping Node (for generating 3d point cloud, orthophoto, DTM, DSM and 3d mesh ; Face Clustering Node (for clustering and identification of faces with a common database) ; Weapon Recognition Node (for identifying weapons (if any) in the sequence of images. Main functions of the RPN are as follows.

1. 3D reconstruction (point cloud & mesh)
2. Orthophoto generation
3. DSM generation
4. DTM generation
5. Face clustering
6. Weapon recognition

2.2 Exhaustive Literature Survey

As defined by the section 2.1, the system architecture has been divided into three main sub-systems.

2.2.1 Unmanned Aerial Vehicle

[@44], [@40] defines a UAV, commonly known as a drone, as an aircraft that can perform flight missions autonomously without a human pilot on board, or can be tele-operated by a pilot from a ground station. The UAV's degree of autonomy can vary with

Tab. 2.1.: Common UAV Types

UAV								
Heavier-than-air							Lighter-than-air	
Wing Type			Rotor Type				Blimp	Balloon
Fixed	Flying	Flapping	Mono	Quad	Hex	Octo		

implementation, however, a UAV often has basic autonomy features such as "self-leveling" using an Inertial Measurement Unit (IMU), "position-holding" using a GPS sensor, and "altitude-holding" using a barometer or a bottom-firing ranging sensor. UAVs with higher degrees of autonomy can offer more functions like automatic take-off and landing, waypoint navigation & obstacle avoidance.

Depending on the flying principle, UAVs can be classified into several types. 2.1 illustrates a common classification method, where UAVs are first classified according to their vehicle mass, "Heavier-than-air" & "Lighter-than-air". "Heavier-than-air" UAVs normally have substantial vehice mass and rely on propulsive thrust to fly. On the other hand, "lighter-than-air" UAVs like blimps and balloons rely on buoyancy force to fly. The "Heavier-than-air" UAVs can be further classified into Wing-type UAV and Rotor Type UAV. "Wing" type UAVs rely on their wings to generate aerodynamic lift, and can be classified into fixed-wing, flying-wing & flapping-wing. "Rotor" type UAVs rely on a single motor (Mono) or multiple rotors (Quad, Hex and Octo) that are pointing upwards to generate propulsive thrust.

[10] surveys 28 different fixed-wing, flapping-wing, and rotor type UAVs and categorizes UAVs with two principal components – flight time versus UAV mass. A simplified version of their original plot is represented in figure 2.1.

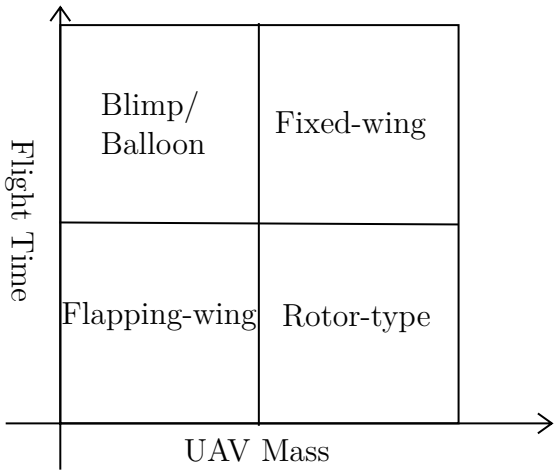

Fig. 2.1.: Flight time versus UAV mass (inspired by Floreano and Wood [10]).

In general, flapping-wing UAVs are usually small and have a short flight time. Blimp/Balloon UAVs are lightweight and have longer flight ime. Rotor-type and fixed-wing UAVs are usually heavy and rely on motor based propulsion thrust. Assuming the same UAV mass and optimal design, fixed-wing UAVs have longer flight time than rotor-type UAVs due to their higher aerodynamic efficiency factor.

[20] focuses on a quadcopter platform and specifies research outcomes regarding dynamic modelling, trajectory planning and state estimation. In addition, [20] outlines several challenges and opportunities of formation flight. [22] presents a survey of the publicly available open-source flight controllers such as Arducopter, Multiwii, Pixhawk, OpenPilot, Aeroquad, and Paparazzi. In addition to the hardware details, [22] also discusses the state estimation method and controller structure of each Flight Controller (FC).

[24] presents a novel, deeply embedded robotics middleware and programming environment. It makes use of a multi threaded, publish-subscribe design architecture and provides a Unix-style software interface for micro controller applications. [24] demonstrates the system modularity and its suitability for novel and experimental vehicle platforms with a vertical takeoff and landing (VTOL). They also show how the system architecture allows a direct interface to Robot Operating System (ROS) and to run individual processes

Tab. 2.2.: Comparison of Open source flight controller software platforms [9]

Platform	Running Processor	Programming Language	Website
ArduPilot	32-bit ARM	C++	www.ardupilot.org
MultiWii	8-bit ATmega 328	C	www.multiwii.org
AutoQuad	32-bit ARM	C	www.autoquad.org
LibrePilot	32-bit ARM	C++	www.librepilot.org

either as native ROS nodes on a linux based companion computer or on a microcontroller, maximizing interoperability.

[9] presents an extensive survey on publicly available open-source flight controllers that can be used for academic research. [9] explains the main components of an autopilot system (Flight controller, propulsion system, sensors, communication systems), open source hardware development platforms such as Phenix Pro, OcPoC, PixHawk, Paparazzi, Naza, CC3d, Ardupilot Mega, FlyMaple, Erle-Brain, AeroQuad, Mikrocopter and MatrixPilot. & open source software platforms such as Ardupilot, AutoQuad, LibrePilot, Multiwii, Dronecode, JAviator and OpenPilot. [9] establishes a clear comparison between open source flight controller hardware platforms (refer Table 2.3) & OSS flight controller software platforms (refer Table 2.2).

The Micro Air Vehicle Link (MAVLink) is a communication protocol for unmanned systems. It specifies a comprehensive set of messages exchanged between unmanned systems and ground stations. This protocol is used in major autopilot systems, mainly ArduPilot and PX4. [19] provides a survey regarding the communication and transport protocols, message types and structures. [19] categorizes MAVLink messages into two classes : (1) State messges, and (2) Command Messages. [19] also surveys security threats to the MAVLink protocol and explains different attacks possible - (1) Eavesdropping, (2) Identity Spoofing, (3) Traffic Analysis, (4) Man-in-the-middle attack, (5) Hijacking

Tab. 2.3.: Comparison of Open source flight controller hardware platforms [9]

Platform	Processor	Sensors	Interfaces	Weight (g)
Phenix	Xilinx Zync SoC (ARM Cortex A9)	HUB, IMU, GPS, LED	CAN, HDMI, LVDS, BT1120-PL	64
OcPoC	Xilinx Zync SoC (ARM Cortex A9)	IMU, Barometer, GPS, Bluetooth, WiFi	PWM, I2C, CAN, Ethernet, SPI, JTAG, UART, OTG	70
PixHawk	ARM Cortex-M4F	IMU, Barometer, LED	PWM, UART, SPI, I2C, CAN, ADC	38
Pararazzi	STM32F767	IMU, Barometer	UART, SPI, I2C, CAN, AUX	-
CC3D	STM32F	Gyroscope, Accelerometer	SBus, I2C, Serial	8
Atom	STM32F	Gyroscope, Accelerometer	SBus, I2C, Serial	4
APM	Atmega 2560	IMU, Barometer, LED	UART, I2C, ADC	31
FlyMaple	STM32	IMU, Barometer	PWM, UART, I2C	15

(Unauthorized command injection) and (6) Denial of Service (DoS/DDoS) by jamming and flooding. [2] proposes MAVSec, a MAVLink enhanced version with cryptographic mechanics to mitigate the vulnerabilities presented in the MAVLink protocol in terms of confidentiality. [2] states that asymmetric ciphers pose a challenge in static communication and outlines the results by using the CTR, CBC, ChaCha20 and RC4 cipher algorithms on the MAVLink protocol.

[@8] lays down fundamental rules regarding drone operations in India and outlines some mandatory equipment requirements which include – (1) GPS, (2) Return-to-Launch (RTL), (3) Anti-collision light, (4) ID plate, (5) RF ID and Sim / No permission No take off.

[15] (ISO/TC20/SC16) specifies a standard in the field of unmanned aircraft systems (UAS) including, but not limited to, classification, design, manufacturing, operation and safety management. The [15]/AG4 specifies general standards and operational procedures for collision avoidance sub-systems. [15]/WG [1 - 6] describe the general classification, product manufacturing and maintenance, operations and procedures, UAS traffic management, testing and evaluation, and UAS subsystems.

[27] presents the mathematical modelling of a quadcopter along with a modified PID control algorithm and establishes the optimum system behavior in a virtual environment. [27] outlines four inputs – (1) Altitude control (U_1), (2) Roll control (U_2), (3) Pitch Control (U_3) & (4) Yaw Control (U_4) and six output states – (1) X, (2) Y, (3) Z, (4) $\theta(roll)$, (5) ψ (Pitch) & (6) ϕ (Yaw).

[27] defines the quad copter dynamics based on four equations [2.1 - 2.4] and outlines a control algorithm as stated in figure 2.2.

1. Throttle

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{2.1}$$

2. Roll

$$U_2 = bl(\Omega_4^2 - \Omega_2^2) \tag{2.2}$$

3. Pitch

$$U_2 = bl(\Omega_3^2 + \Omega_1^2) \tag{2.3}$$

4. Yaw

$$U_4 = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \tag{2.4}$$

On transformation using inverted matrix method,

1.

$$\Omega_1^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \tag{2.5}$$

2.

$$\Omega_2^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{2.6}$$

3.

$$\Omega_3^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 - \frac{1}{4d}U_4 \tag{2.7}$$

4.

$$\Omega_4^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{2.8}$$

2.2.2 Mission Management Interface

The MMI provides a graphical user interface for creation, modification and archival of missions. The MMI is also responsible for maintaining connectivity between all the three sub-systems - (1) UAV (2) MMI & (3) RPN. Transmission of different type of data requires different Quality of Service (QoS). For example, a real-time video feed does not

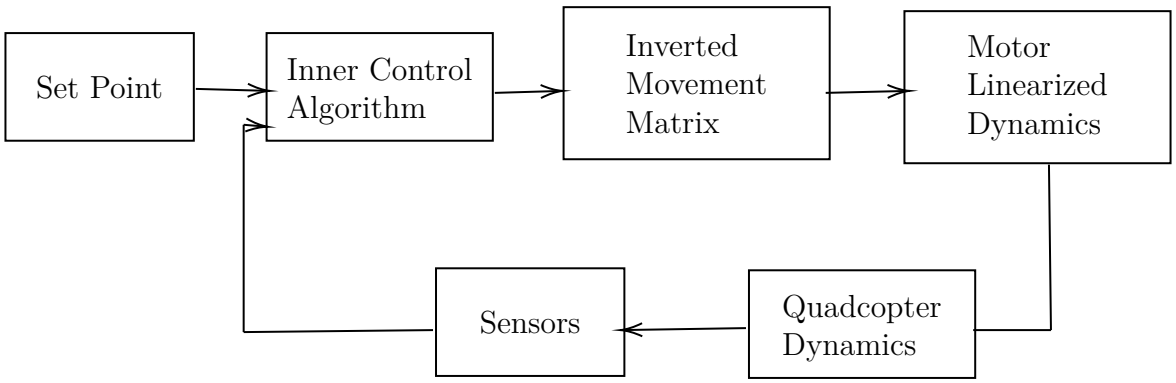

Fig. 2.2.: Figure: Block diagram of control block [27]

make use of a connection oriented communication protocol because at long telemetry distances, the number of retries to transmit a frame without errors will grow, making the system unfit for transmission of real-time video signals. However, when the UAV has completed a mission, the geo-tagged still photographs have to be transmitted without any errors whilst maintaining data integrity, and thus a connection oriented protocol (Transmission Control Protocol (TCP)) is used.

[38] defines a simple connection-less communication model with a minimum of protocol mechanisms. It provides check sums for data integrity and port numbers for addressing different functions at source and destination of datagram. The communication standard does not have handshaking dialogues and this exposes the users program to any unreliability of the underlying network – there is no guarantee of delivery, ordering or duplicate protection.

Applications can use datagram sockets to establish host-to-host communications. An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a port. In this way, UDP provides application multiplexing. A port is a software structure that is identified by the port number, a 16 bit integer value, allowing for port numbers between 0 and 65535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

[37] defines a connection-oriented communication model. TCP is connection-oriented, and a connection between client and server is established (passive open) before data can be sent. Three-way handshake (active open), retransmission, and error-detection adds to reliability but lengthens latency. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that prioritizes time over reliability [38]. TCP employs network congestion avoidance. However, there are vulnerabilities to TCP including denial of service, connection hijacking, TCP veto, and reset attack.

Though TCP is a complex protocol, its basic operation has not changed significantly since its first specification. TCP is still dominantly used for the web, i.e. for the HTTP protocol [13], and later HTTP/2 [14], while not used by latest standard HTTP/3.

[19] defines a communication protocol that is exclusive for communication between a companion computer and a flight controller. More information on this has been dicussed in para 2.2.1.

[@29] defines an orthophoto, orthophotograph or orthoimage as an aerial photograph or satellite imagery that is geometrically corrected ("orthorectified") such that the scale is uniform: the photo or image follows a given map projection. Unlike an uncorrected aerial photograph, an orthophoto can be used to measure true distances, because it is an accurate representation of the Earth's surface, having been adjusted for topographic relief, lens distortion, and camera tilt. Orthophotographs are commonly used in Geographic Information System (GIS) [@42] as a "map accurate" background image.

GIS is a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data. [@42]. [@41] defines a Digital Elevation Model (DEM) as a 3D CG representation of a terrain's surface – commonly of a planet (e.g. Earth), moon, or asteroid – created from a terrain's elevation data. A "global DEM" refers to a discrete global grid.

A DEM is required to create an accurate orthophoto as distortions in the image due to the varying distance between the camera/sensor and different points on the ground need to be corrected.

MMI provides tools for visualising the DEM (DSM & DTM), and the orthophoto. However, the size of high resolution rasters (2-5 cm/pixel) is large. This poses a problem in rendering them on a browser interface. [32] provides a systematic method for displaying a portion of map on a mobile device by making use of a tile map service. Tile Map Service or TMS, is a specification for tiled web maps, developed by the Open Source Geospatial Foundation. The definition generally requires a URI structure which attempts to fulfill REST principles [@43].

2.2.3 Remote Processing Node

Section 3.1.3 outlines the main functions for the RPN. The RPN sub-system architecture has been strategically divided into three nodes – (1) Aerial Mapping Node, (2) Face Clustering Node & (3) Weapon Recognition Node.

In 3D reconstruction, a space is mapped by fusing the data from a moving sensor into a representation of the consistent surfaces it contains, allowing precise viewpoint-invariant localisation. Typically, Simultaneous Localisation and Mapping (SLAM) systems are deployed for generation of a real-time 3D point-cloud of the environment. This technique also offers the potential for detailed semantic scene understanding. Real-time 3-dimensional scene reconstruction poses a huge challenge as the real-time operation struggles when the sensor makes movements which are both of extended duration and often criss-cross loop back on themselves. Such a trajectory is typical if an UAV is used. This problem is resolved by using a dense vision [39] front-end, in which the number of points matched and measured at each sensor frame is much higher than feature based systems.

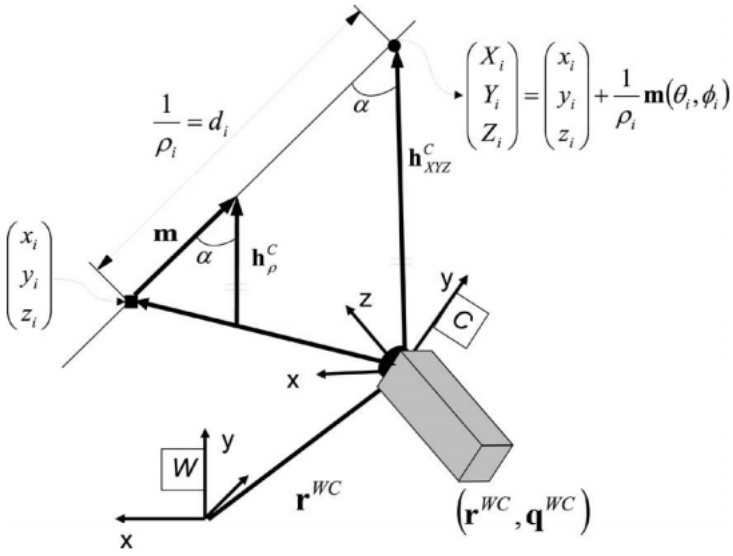

Fig. 2.3.: Feature Parametrization of each point [7]

The first SLAM model was proposed by [7]. [7] presents new parametrization techniques for point features within monocular (SLAM) that allows efficient and accurate representation of uncertainty the time of operation, all within the standard extended Kalman filter (EKF). The key concept is direct parametrization of the inverse depth of features relative to the camera locations from which they were first viewed, which produces measurement equations with a high degree of linearity. The inverse depth parametrization remains well behaved for features at all stages of SLAM processing, but has the drawback in computational terms that each point is represented by a 6-D state vector as opposed to the standard three of a Euclidean XYZ representation.

[7] uses the following method for paramaterization of each point. The standard representation for scene points i in terms of Euclidean XYZ coordinates (Figure 2.3) is given by equation 2.9.

$$x_i = (X_i \; Y_i \; Z_i)^T \tag{2.9}$$

A new 3D point i is given by Equation 2.10.

$$y_i = (x_i \; y_i \; z_i \; \Theta_i \; \Phi_i \; \rho_i)^T \tag{2.10}$$

The model of the 3D point is given as 2.11.

$$x_i = \begin{pmatrix} X_i \\ \\ Y_i \\ \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ \\ y_i \\ \\ z_i \end{pmatrix} + \frac{1}{\rho_i} m(\Theta_i, \Phi_i) \tag{2.11}$$

$$m = (\cos \Phi_i \sin \Theta_i - \sin \Phi_i, \cos \Phi_i \cos \Theta_i)^T \tag{2.12}$$

[26] makes an attempt to overcome the problems faced by [7] by using a feature based monocular SLAM system. [26] computes special features, termed *FAST Corners*, and performs *RANSAC* transform in two consecutive frames. The transform returns the camera translation matrix between two consecutive frames, *i.e*, it provides correlative information about each pixel and its translated position in the new frame. [26] is a feature-based monocular SLAM system that operates in real time, in small and large, indoor and outdoor environments. This method makes use of a single camera. The algorithm works on three threads, a tracking thread, a local mapping thread and a loop closing thread. Functioning of [26] is given in the block diagram given in figure 2.4.

1. **Initialization of Map** To initialize the map starting by computing the relative pose between two scenes, Two geometrical models are calculated in parallel in parallel. A score-metric system is used for selection of model. The score of each model is computed using equation 2.13. Using the selected model they estimate

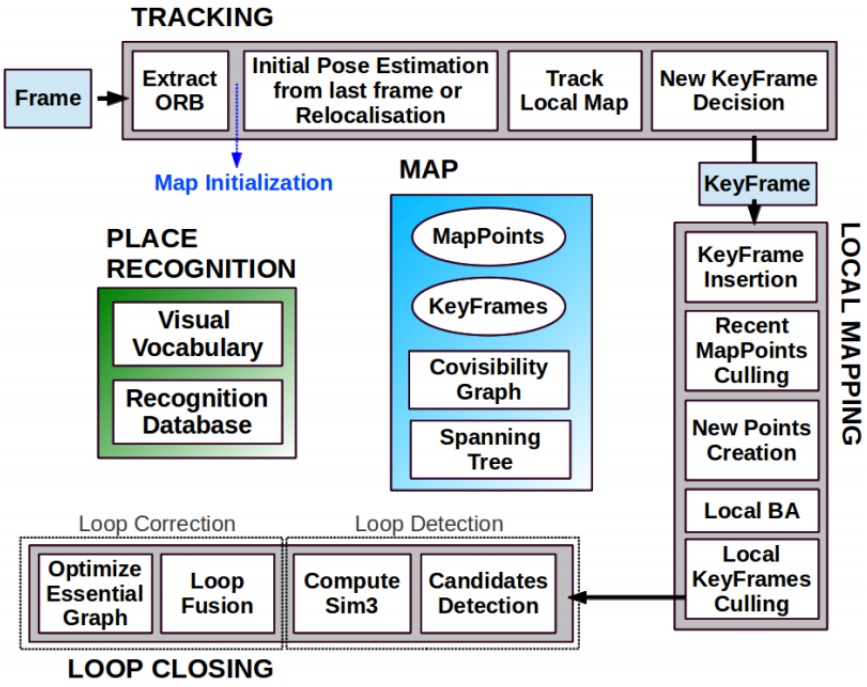

Fig. 2.4.: Block Diagram for ORB SLAM [26]

multiple motion hypotheses and see if one is significantly better than the other, if so, a full bundle adjustment is done, otherwise the initialization starts over.

$$S_M = \sum(\rho_M(d_c^2 r(x_c^i, x_r^i, M)) + \rho_M(d_{rc}^2(x_c^i, x_r^i, M))) \tag{2.13}$$

2. **Tracking**

The tracking part localizes the camera and decides when to insert a new keyframe. Features are matched with the previous frame and the pose is optimized using motion-only bundle adjustment. The features extracted are FAST corners. (for res. till 752x480, 1000 corners should be good, for higher (KITTI 1241x376) 2000 corners works). Multiple scale-levels (factor 1.2) are used and each level is divided into a grid in which 5 corners per cell are attempted to be extracted. The initial pose is estimated using a constant velocity motion model. If the tracking is lost,

the place recognition module kicks in and tries to re-localize itself. As stated above, The tracking process runs in parallel with the local mapping process.

3. **Local Mapping**

   First the new keyframe is inserted into the co-visibility graph, the spanning tree linking a keyframe to the keyframe with the most points in common. New map points are created by triangulating ORB from connected keyframes in the co-visibility graph. The unmatched ORB in a keyframe are compared with other unmatched ORB in other keyframes. The new map points first need to go through a test to increase the likelihood of these map points being valid. They need to be found in more than 25% of the frames in which it is predicted to be visible and it must be observed by at least three keyframes. To detect possible loops, they check bag of words vectors of the current keyframe and its neighbors in the co-visibility graph. The min. similarity of these bag of words vectors is taken as a benchmark. RANSAC iterations are performed to find them and these are then optimized after which more correspondences are searched and then again an optimization is performed. If the similarity is supported by having enough inliers, the loop is accepted.

[26] makes use of monocular images, and thus the depth mapping capabilities of the system are limited. [21] overcomes this by making use of Red-Blue-Gree-Depth (RGBD) frames. [21] is split in two steps: the front-end (odometry) and the back-end (loop closure detection and graph optimization). The main difference between [26] and [21] is that [21] works with RGBD images, that is, RGB + Depth image. [21] makes use of Robot Operating System (ROS) as a communication link between the sensors and the main node. The block diagram of the ROS node for [21] is represented by figure 2.5.

1. **Front End**

   The front-end algorithm has constant time complexity The required input frequency of images is 20 Hz Similar to ORB-SLAM, the RTAB-Map extracts visual

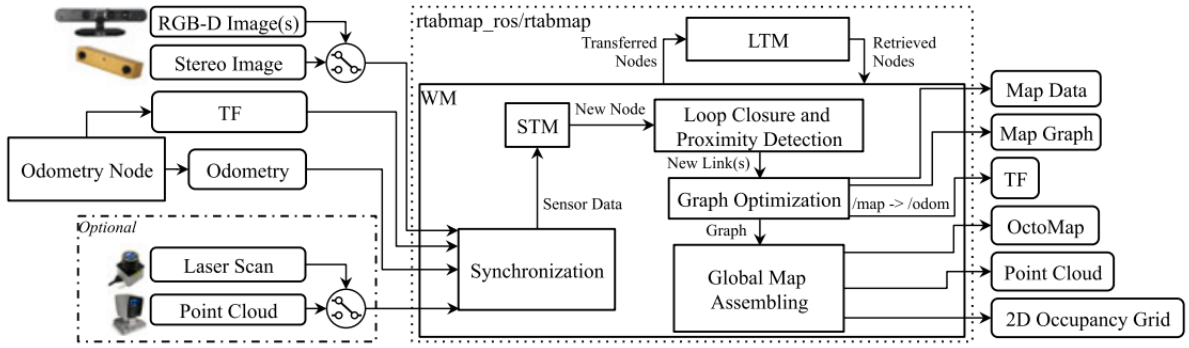

Fig. 2.5.: Block Diagram for RTAB-Map ROS Node [21]

features. Extract visual features in the RGB image, get the depth of these features using the Depth image. Then a RANSAC rigid transformation estimation is performed in the images with the previous image using the corresponding 3D features (correspondences are found by matching 2D visual features between the RGB images).

2. **Backend**

   The backend function is related to loop closure and graph optimizations. Linear complexity (when memory management of RTAB-Map is disabled). Don't have to be done at high frame rate, so 1 Hz is used by default. The graph is created here, where each node contains RGB and depth images with corresponding odometry pose. The links are transformation between each node. When the graph is updated, RTAB-Map compares the new image with all previous ones in the graph to find a loop closure. When a loop closure is found, graph optimization is done to correct the poses in the graph.

3. **Visualization** For each node in the graph, we generate a point cloud from the RGB and depth images. This point cloud is transformed using the pose in the node. The 3D map is then created.

[46] goes one step ahead by combining geometric registration of scene fragments with global optimization based on line processes. [46] divides the dataset into fragments with N fragments. By taking stereo pairs and computing camera translation vectors, [46] first computes the 3D model of each fragment. Then by applying a global registration algorithm on to all the fragments, the system computes a 3D reconstruction of the entire scene. [46] heavily relies on the work done in [6] for the reconstruction framework.

The functioning of reconstruction pipeline by [46] takes place in 4 stages.

1. **Fragment construction** Individual range images are noisy and incomplete. To derive more reliable information on local surface geometry, the system partitions the input RGB-D video into k-frame segments (k = 100 in for the current project), use RGB-D odometry to estimate the camera trajectory. The images have a larger footprint in the scene than individual images without suffering from significant odometry drift. Fragments are analogous to submaps, which are used in a number of robotic mapping systems.

2. **Geometric Reconstruction** Due to odometry drift, simply using the transformations to localize the fragments yields broken reconstructions in which non-consecutive fragments that cover overlapping parts of the scene are misaligned. So, each pair of fragments is tested to find overlapping pairs. A geometric registration algorithm is run on each pair (Pi , Pj). If the algorithm succeeds in aligning the fragments with sufficient overlap. The geometric registration is performed with reference to equation 2.14.

$$\frac{1}{\left|K_{ij}^{*}\right|}\sum \|T_{ij}p^{*}-q^{*}]\|^{2}<\tau^{2} \tag{2.14}$$

   A fairly small threshold is used (0.2meters).

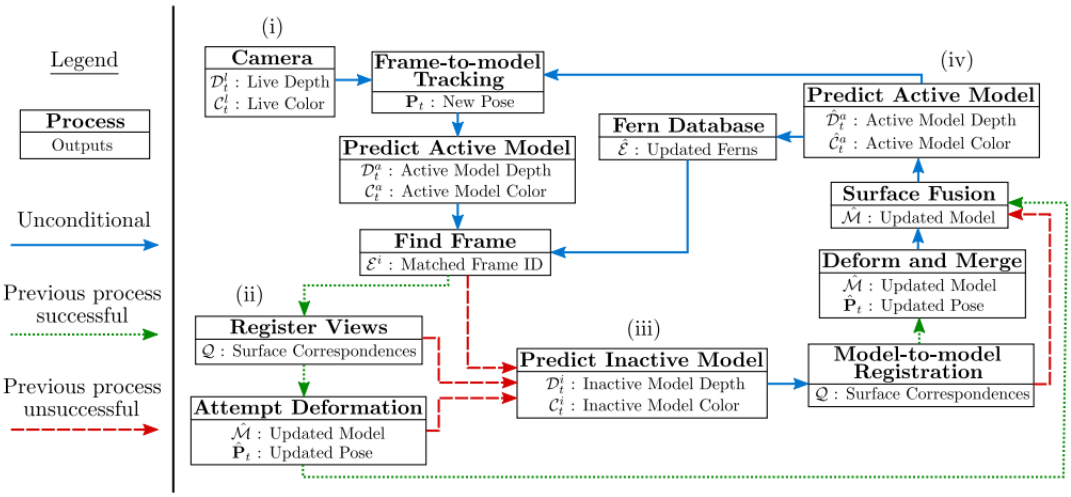

Fig. 2.6.: Block Diagram for Elastic Fusion [39]

3. **Robust Optimization** Many of the loop closure pairs can be false-positives. [46] identifies these spurious loop closures by optimizing a dense surface registration objective augmented by a line process over the loop closure constraints.

4. **Final Model** Optional nonrigid refinement can be used to further improve the registration. The registered fragments are fused into a global mesh model by volumetric integration.

[39] recognizes that refined algorithms such as [46] pose a great challenge if run in real-time, as with every possible solution for a loop-closure, the odometry and the pose graph has to be optimized, which is quite computationally expensive.

[39] makes use of a dense vision frontend and avoids pose graph optmization entirely. The basis of [39]'s functionality can be given by the block diagram represented in figure 2.6.

1. Estimate a fused surfel-based model of the environment. This component of the method is inspired by the surfelbased fusion system defined in [16].

2. While tracking and fusing data in the area of the model most recently observed (active area of the model), segment older parts of the map which have not been observed in a period of time t into the inactive area of the model (not used for tracking or data fusion).

3. Every frame, attempt to register the portion of the active model within the current estimated camera frame with the portion of the inactive model underlaid within the same frame. If registration is successful, a loop has been closed to the older inactive model and the model is non-rigidly deformed into place to reflect this registration. The inactive portion of the map which caused this loop closure is then reactivated to allow tracking and surface fusion (including surfel culling) to take place between the registered areas of the map. The geometric pose estimation for [39] is given by 2.15

$$E_{icp} = \sum((v^k - exp(\varepsilon)Tv_t^k).n^k)^2 \tag{2.15}$$

4. For global loop closure, add predicted views of the scene to a randomised fern encoding database.

[36] outlines that ideally, for a SLAM systems to meet real-time requirements, the system should be able to process at-least 30 image frames per second. Along with this, most of the depth camera's available in the market (ZED Camera, Intel Realsense, Kinect, etc) provide a maximum range of 10-30m for accurate depthmaps making them unsuitable for large scale out-door mapping using UAV.

[1] introduces a reconstruction and pipeline based on photogrammetry algorithms called Structure from Motion (SfM) and Multi View Stereo (MVS). It is a method for estimated 3D structure and camera motion out of a series of 2D images. While SfM only reconstructs a sparse scene structure, MVS is a technique used to refine this sparse structure in order to receive a dense scene reconstruction.

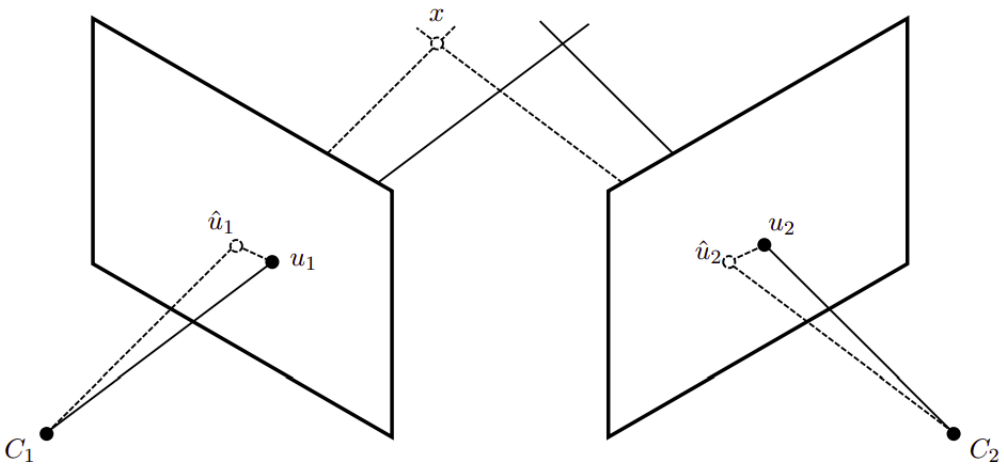

Fig. 2.7.: Triangulation in structure-from-motion algorithms [@34]

SfM in combination with MVS is such a low-cost passive photogrammetric method used to generate large point datasets. It refers to the process of estimated three-dimensional structures out of two-dimensional image sequences. These sequences can even be captured with conventional consumer-grade digital cameras. A SfM engine detects corresponding image points in the given input images and reconstructs three-dimensional points by means of triangulation as depicted in figure 2.7. The projection matrices, representing the extrinsic and intrinsic camera parameters, needed for point reconstruction can be computed simultaneously with the reconstruction. Additionally, bundle adjustment algorithms are used to refine the outcome and minimize an appropriate cost function [@34]. Furthermore, MVS is used to generate a dense reconstruction using the camera information computed by SfM.

Triangulation in SfM algorithms : A 3D point x can be computed form the viewpoints $(u_1, u_2, ...)$ measured from different views (C_1, C_2) by intersection their back-projected viewing rays. Due to measurement errors, these rays don't intersect exactly. Therefore x is a 3D point, which minimises sum of squared errors between the measured and calculated viewpoints.

To do triangulation, the extrinsic and intrinsic camera parameters that make up its projection matrix are needed. These parameters can either be computed by calibrating the cameras before doing the reconstruction or retrieved on the go during the reconstruction process with SfM [1].

Camera calibration is the process of determining the extrinsic and intrinsic parameters of an image sensor and its lens, in order to create its camera matrix or projection matrix P. P is used to compute the projected position x of a 3D point X on an image plane. While the extrinsic parameters describe the cameras position and orientation in the world, the intrinsic values stand for its internal characteristics like the focal length of the lens, its skew, optical centre and distortion. The camera matrix can be described by equation 2.16, which is derived from a pinhole camera model (refer figure 2.8) [12].

$$x = PX = K[R|t]X = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 1 & r_1 2 & r_1 3 & t_1 \\ r_2 1 & r_2 2 & r_2 3 & t_2 \\ r_3 1 & r_3 2 & r_3 3 & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.16}$$

Equation 2.16 consists of the intrinsic matrix K and the extrinsic matrix $[R|t]$, containing the rotation R and translation t. x_0 and y_0 are the coordinates of the principle point p. The parameters α_x and α_y describe pixel scale factors in x- and y-direction, which are not equal if the pixels on a camera sensor are non-square (distorted). These parameters are defined using the focal length f and the number of pixel per unit distance m_x and m_y in x- and y-direction. s defines the skew between the sensor axes.

The algorithm can be executed using the following steps [@34] :

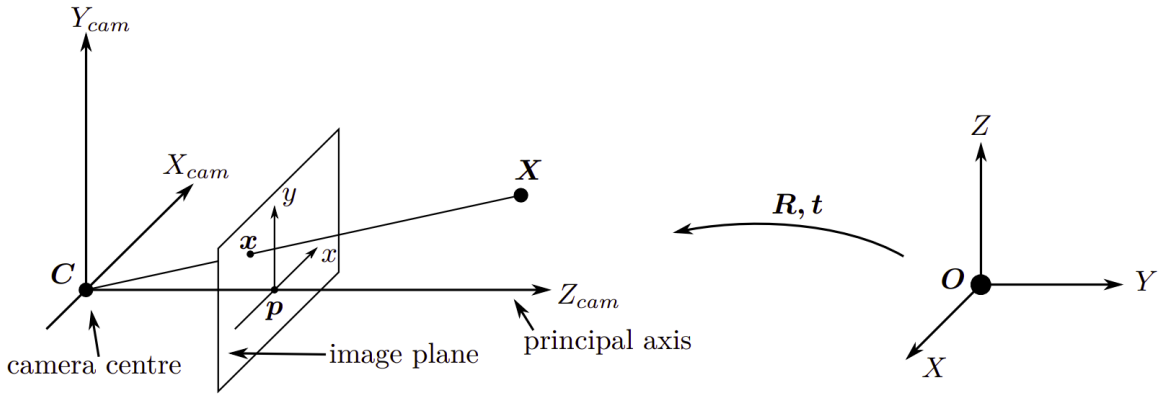

Fig. 2.8.: Pinhole Camera Model: The left part depicts the geometry of a pinhole camera, where C is the optical centre. The line from C perpendicular to the image plane is called the *principal axis* and the point where image plane and this axis intersect is known as principle point p. The right side shows the Euclidean transformation consisting of a rotation R and translation t, between the world and camera coordinate system.

1. Feature Detection : The first step of the pipeline is the feature detection step, where distinctive image interest points are recovered, that can be subsequently used in the next step to find relations between different images. The most prominent [4] detectors is the Scale-invariant feature transform as described by [23].

2. Finding correspondences, motion and structure : After identifying the features, they can be used to find feature correspondences between the images, which are then used to recover the relative camera positions and orientations(motion) and the locations of the 3D points (structure). The pair-wise matches are optimised using a RANSAC (Random Sample Consensus) loop where each iteration, called as fundamental matrix is calculated describing the relation between the two views.

3. Dense reconstruction using MVS : Multi-view stereo approaches are able to produce a dense scene reconstruction out of stereo correspondences and camera parameters.

Along with the generation of a 3D textured mesh, the aerial processing node also generates DEM. Digital Elevation Models (DEMs) are used to derive information from the morphology of a land. The topographic attributes obtained from the DEM data

allow the construction of watershed delineation useful for predicting the behavior of systems and for studying hydrological processes. Imagery acquired from Unmanned Aerial Vehicles (UAVs) and 3D photogrammetry techniques offer cost-effective advantages over other remote sensing methods such as LIDAR or RADAR. In particular, a high spatial resolution for measuring the terrain microtopography. [25] proposes a Structure from Motion (SfM) pipeline using UAVs for generating high-resolution, high-quality DEMs for developing a rainfall-runoff model to study flood areas. SfM is a computer vision technique that simultaneously estimates the 3D coordinates of a scene and the pose of a camera that moves around it. The result is a 3D point cloud which we process to obtain a georeferenced model from the GPS information of the camera and ground control points. [25].

The proposed pipeline has four stages. The first stage, camera calibration stage is optional [@34] since [1] provides a way to compute the extrinsic and intrinsic projection matrix on the fly. The pipeline proceeds as follows.

1. Flight Plan stage : Image Acquisition : Three-dimensional reconstruction algorithms require a set of overlapping, offset images of the object or scene of interest acquired from different positions. The first stage consists in setting the flight path for the drone to carry out the image acquisition. More of this on current implementation can be found in Chapter 3. [45] provides an in-depth analysis of effect of different overlap values on the end result.

2. SfM Stage : The second stage consists of performing the 3D reconstruction, as explained above with reference to [1], [@34]. The outputs are used as a first estimation and are refined in an iterative non-linear optimisation process known as Bundle Adjustment [25].

3. Post-processing : With the sparse and dense reconstruction with SfM in PLY format, we post-process the point cloud to generate a geo-referenced point cloud in LAS format, a geo-referenced 3D textured mesh and an orthophoto mosaic in the

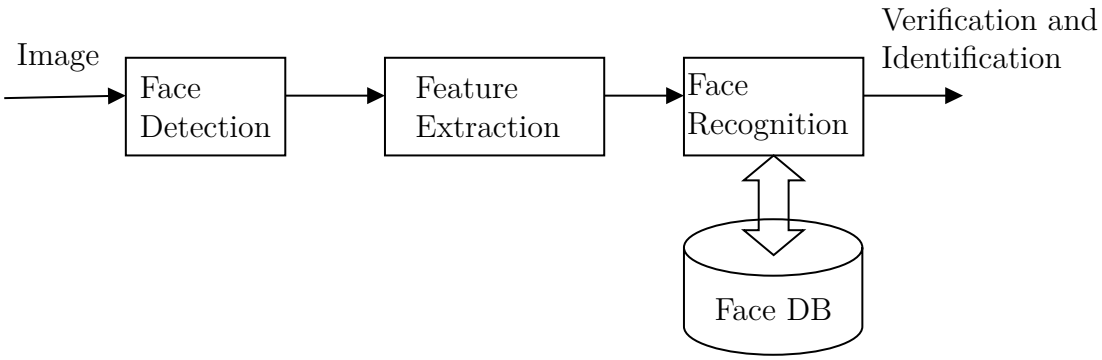

Fig. 2.9.: Block diagram of a general face recognition system. Redrawn as seen in [28]

GeoTIFF format. The LAS format is a standard binary format for the storage of LIDAR data and point clouds.

4. Generation of DTM & DSM : With the point cloud in LAS format, we have ground and non-ground points of scene so that our output point cloud from SfM pipeline represents a DSM of the scene. In order to generate a DTM using the LAS point cloud, the point cloud is filtered to remove the non-ground points that represent objects like platns or buildings. The filtering is carried out with an open-source library called PDAL [31], using the Extended Local Minimum Method [5] to identify low noise points and the Simple Morphological Filter approach [30] implementing nearest neighbour void filling to segment ground and non-ground points.

RPN provides a dedicated node for face detection and recognition. Three basic steps are used to develop a robust face recognition system – (1) face detection, (2) feature extraction & (3) face recognition as depicted in figure 2.9.

[18] presents a classification and comparison of face recognition systems. [18] classifies the systems into three approaches based on their detection and recognition method – (1) Local, (2) Holistic (subspace) & (3) Hybrid. The first approach is classified according to certain facial features, not the whole face. The second approach employs the entire face as input data and then projects into a small subspace or in a correlation plane. The third

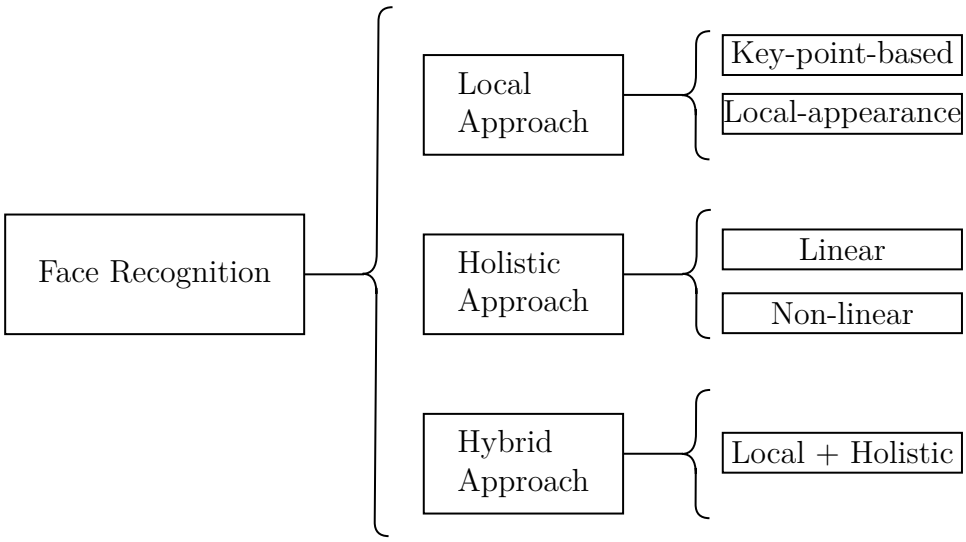

Fig. 2.10.: Face recognition methods (classification) [18]

approach uses local and global features in order to improve face recognition accuracy. This system of classification can be seen in the figure 2.10.

1. Local Approaches : In context of face recognition, local approaches treat only some facial features. They are more sensitive to facial expressions, occlusions and pose. The approach is classified into two types : (1) Local appearance-based techniques : The local appearance based techniques are used to extract local features while the face is divided into small regions (patches) & (2) Key-points-based techniques : where points of interest are detected in the face image, after which the features localised on these points are extracted.

2. Holistic Approach : Holistic or subspace approaches are supposed to process the whole face, that is, they do not require extracting face regions or feature points. The main function of these approaches is to represent the face image by a matrix of pixels, and this matrix is often converted into a low dimensional space. However, holistic or subspace techniques are sensitive to variations, and these advantages make these approaches widely used. The holistic approaches can be further classified into two types – (1) Linear : Linear techniques such as Principal component analysis (PCA), linear discriminate analysis (LDA) and independent component analysis

(ICA) are applied for dimensionality reduction. (2) Non-linear : Different non-linear algorithms are applied such as Kernel linear discriminant analysis (KDA), Gabor-KLDA, Wavelet Transform (WT), Radon Transform (RT), Convolutional Neural Network (CNN), Kernelized maximum average margin, SVM, and kernel fisher discirmininant analysis.

3. Hybrid : Makes use of techniques from both, local approaches as well as holistic approaches.

Traditional face recognition system have a common pipeline, the faces are first detected and later on the region of interest from the frame is passed onto a neural network for classification. However, traditional face recognition systems use convolutional layers with generic kernels, and thus do not perform well with facial data. [3] moves from generic kernels to face feature vectorization with the help of [17]. [3] makes use of a Support Vector Machine for classification. However, our findings have concluded that an approximate nearest neighbors search algorithm outperforms the SVM in this particular usecase. [3], given an input image with multiple faces, face recognition systems typically first run face detection to isolate the faces. Each face is preprocessed and then a low-dimensional representation (or embedding) is obtained. The general block diagram for [3] is given in figure 2.11.

1. **Preprocessing** [3] uses a simple 2D affine transformation to make the eyes and nose appear in similar locations for the neural network input.The 68 landmarks are detected with dlib's face landmark detector[17]. Given an input face, our affine transformation makes the eye corners and nose close to the mean locations. The affine transformation also resizes and crops the image to the edges of the landmarks so the input image to the neural network is 96 x 96 pixels. The transform is as shown in 2.12.

2. **Sampling** [3] makes use of a special sampling function - the triplet-loss sampling function. [3] maps unique images from a single network into triplets. The gradient

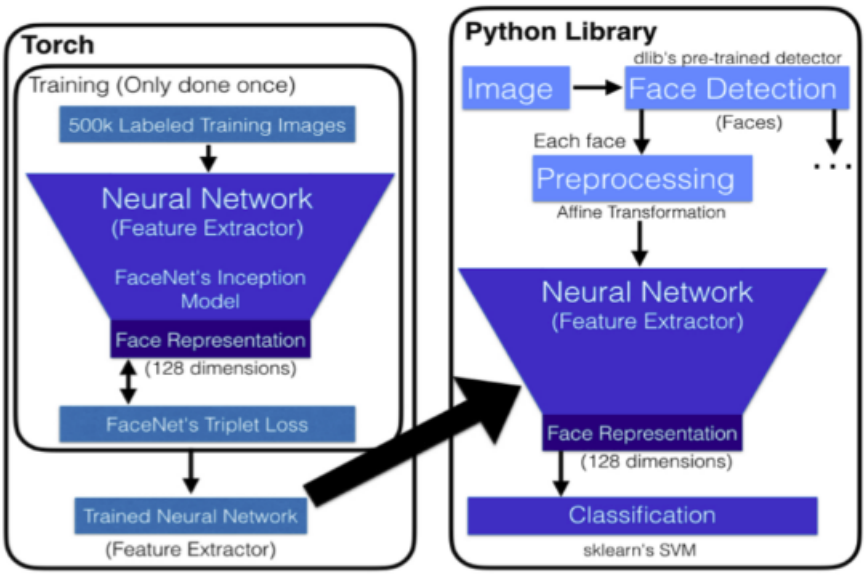

Fig. 2.11.: Block Diagram for OpenFace [3]

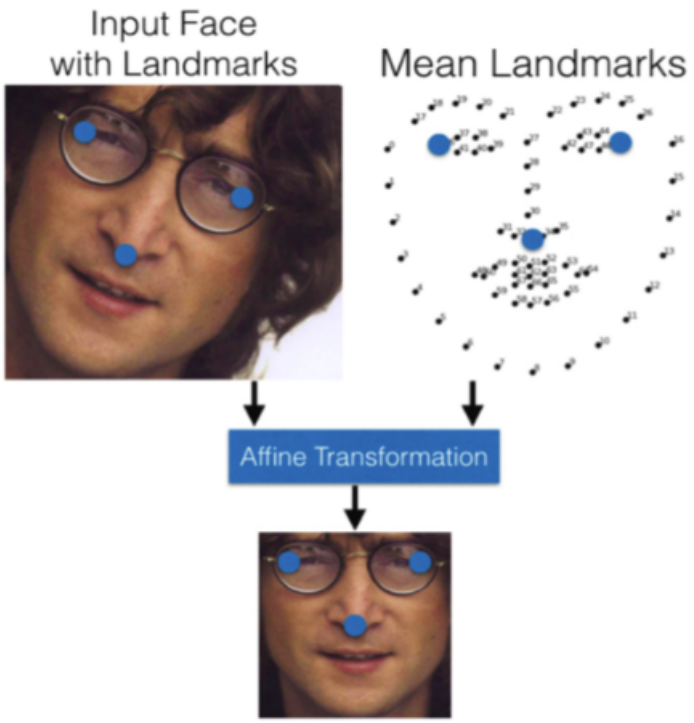

Fig. 2.12.: Affine Transformation for Face Image [3]

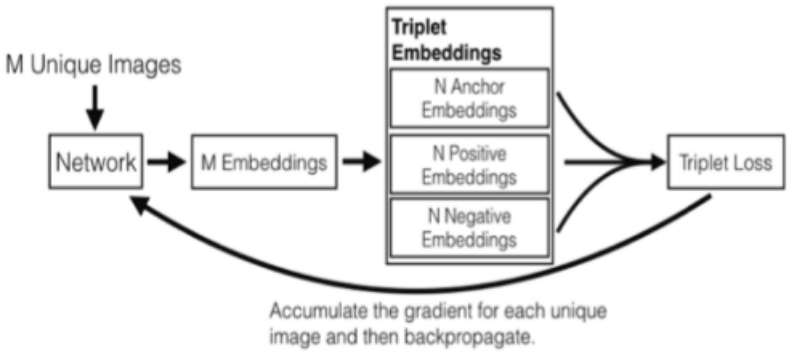

Fig. 2.13.: Triplet loss sampling

of the triplet loss is propagated back through the mapping to the unique images. In each mini-batch, we sample at most P images per person from Q people in the dataset and send all M = P Q images through the network in a single forward pass on the GPU to get M embeddings. Currently P = 20 and Q = 15 is used. All anchor-positive pairs are taken to obtain triplets. The triplet loss is computed and the derivative is mapped back through to the original image in a backwards network pass. If a negative image is not found within the margin for a given anchor-positive pair, we do not use it. (Refer figure 2.13.

3. **Training** The feature vectors generated after triplet loss sampling are then fed into a Support Vector Machine

The RPN also provides a comprehensive pipeline for weapon recognition based on [33]. It defines a neural network capable of detecting what is in an image and where each object is, in one pass. It gives the bounding boxes around the detected objects, and it can detect multiple objects at a time, see this sample image. The major innovation YOLO brought when it came about was the fact that it is capable of performing the detections in one go, which is why it is quite fast and performant. It works by performing a regression - it predicts the bounding boxes and the class probabilities for each, doing so with a single network pass (hence the name). Other approaches usually employ a pipeline of tasks,

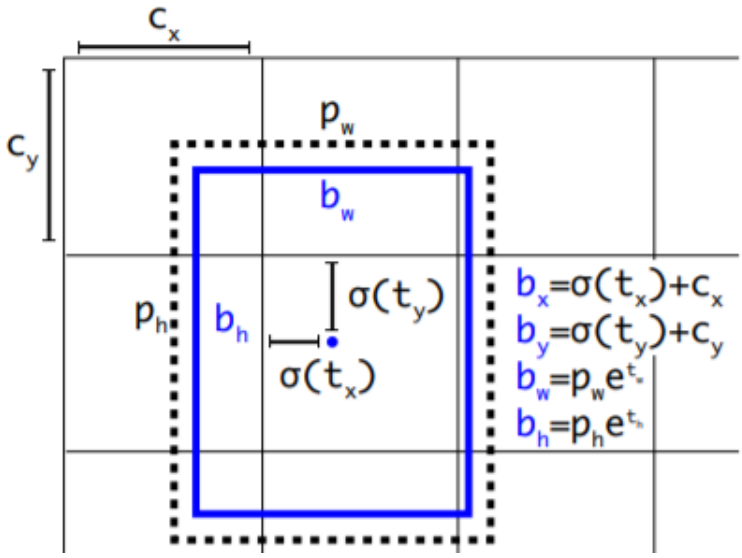

Fig. 2.14.: Yolo V3 Working [33]

like passing on the image some classifier(s) to detect stuff in different locations and/or utilising some other added methodologies. YOLO looks at the image once.

[33] has 24 convolutional layers working as feature extractors and 2 dense layers for doing the predictions. The architecture it works upon is called Darknet, a neural network framework created by [33]. The algorithm works off by dividing an image into a grid of cells; for each cell bounding boxes and their confidence scores are predicted, alongside class probabilities. The confidence is given in terms of an IOU (intersection over union), metric, which is basically measuring how much a detected object overlaps with the ground truth as a fraction of the total area spanned by the two together (the union). The loss the algorithm minimises takes into account the predictions of locations of the bounding boxes, their sizes, the confidence scores for said predictions and the predicted classes.

Following YOLO9000 the system predicts bounding boxes using dimension clusters as anchor boxes. The network predicts 4 coordinates for each bounding box, tx, ty, tw, th. If the cell is offset from the top left corner of the image by (cx, cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to the following. Refer to figure 2.14.

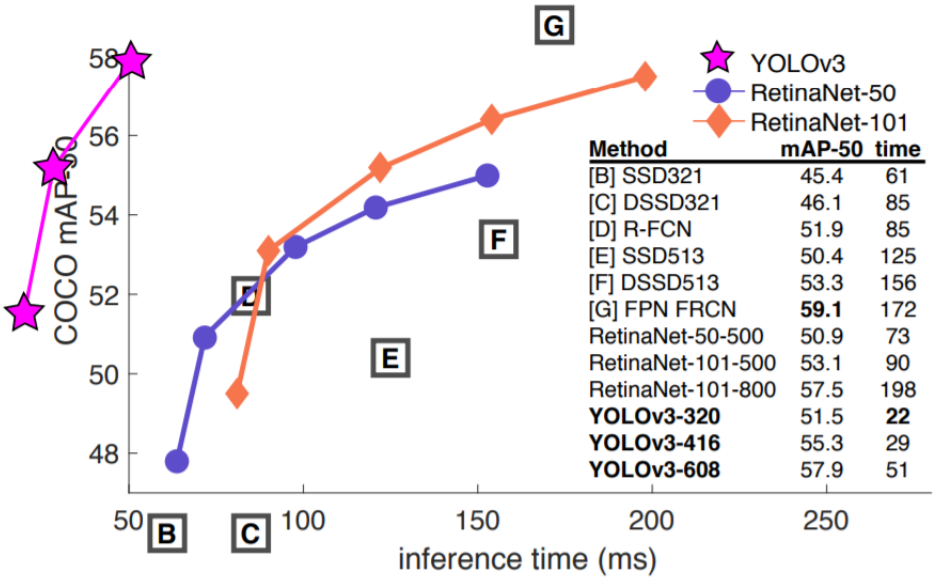

Fig. 2.15.: Yolo V3 Performance Comparison [33]

$$b_x = \sigma(t_x) + c_x \tag{2.17}$$

$$b_y = \sigma(t_y) + c_y \tag{2.18}$$

$$b_w = p_w e^{t_w} \tag{2.19}$$

$$b_h = p_h e^{t_h} \tag{2.20}$$

During training sum of squared error is used as loss metric. The ground truth value can be easily computed by inverting the equations above. YOLOv3 predicts an object's score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. The prediction of the width and height of the box is done as offsets from cluster centroids. The

prediction of the center coordinates of the box relative to the location of filter application is done using a sigmoid function. The performance comparison of YOLO v3 with respect to other object recognition models is given in figure 2.15.

3 Methodology & Implementation

This chapter first introduces a holistic overview of the complete pipeline, outlines the functions of each sub-system, and provides an understanding of how everything is connected. The chapter also explains the in-depth working of each node within the pipeline.

3.1 Holistic Overview

Section 2.1 outlines the the functions of all the three sub-systems – (1) Unmanned Aerial Vehicle (UAV), (2) Mission Management Interface (MMI) & (3) Remote Processing Node. Further sections discuss the design and implementation of each of these sub-systems.

Figure 3.1 depicts the functions of each sub-system and provides with an idea about the information that is exchanged between the subsystems. The following sub-sections describe each of the function in detail – the inputs required, the expected output and technologies used. In further sections within the software implementation, working model and implementation of each function will be discussed.

3.1.1 Understanding the Unmanned Aerial Vehicle (UAV)

The resolution, and quality of the outputs can be significantly increased by minimising noise. Flying a drone is a tedious task and requires proper training to achieve stable flights. The reconstruction system not only requires stable flight, but also a minimum of

38

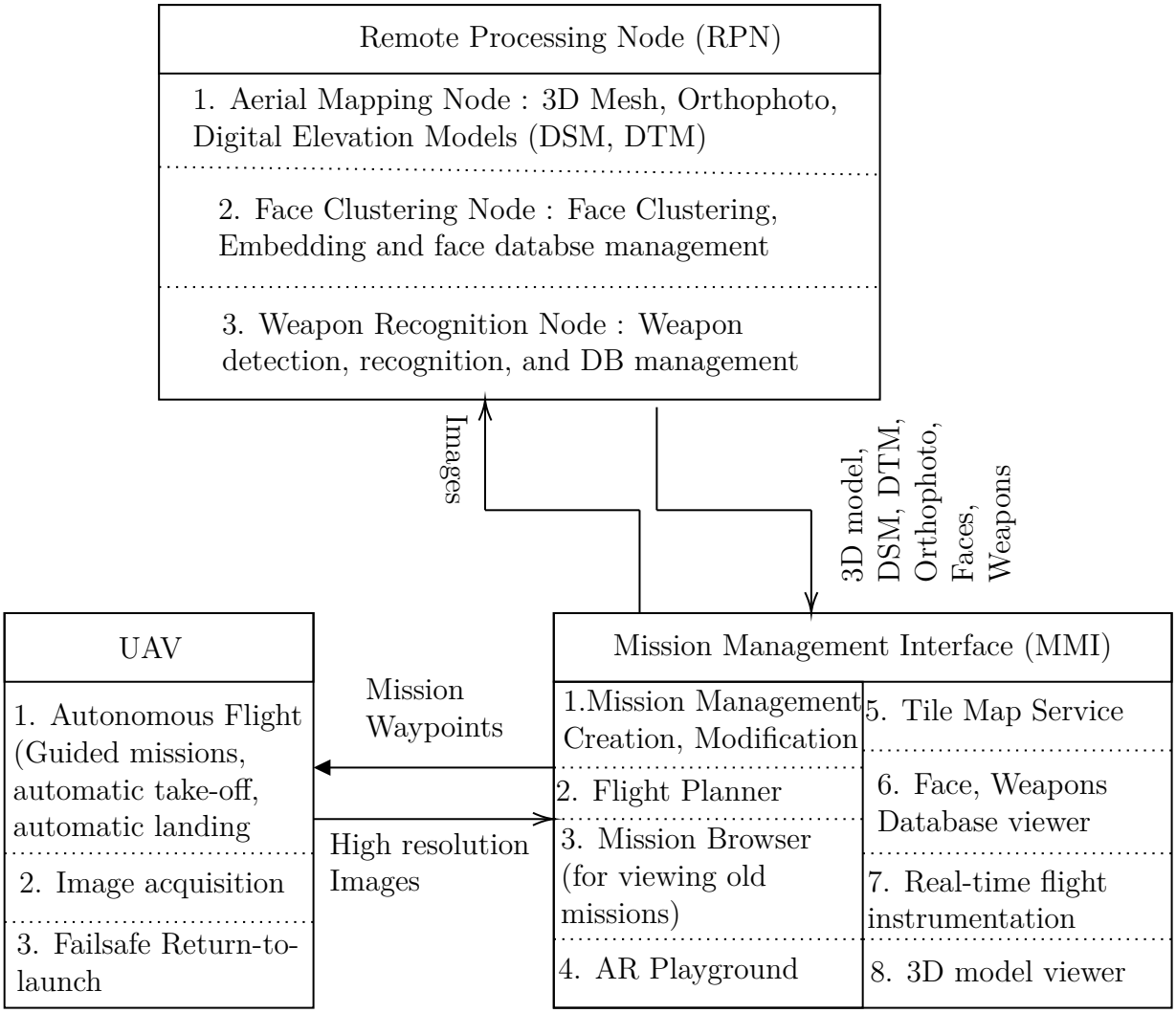

Fig. 3.1.: Functions of each sub-system and the data transmitted among each of the sub-system

70% forward overlap and a minimum of 65% side overlap for accurate mapping. Thus, it is recommended to have an autopilot in charge of flying and capturing data. Ideally, a matured autopilot system must be capable of accomplishing the following tasks :

1. Autonomous take-off and land : During the configuration of the mission plan, the operator can set the home position (GPS Coordinates) and the home altitude (relative to the starting point). The operator can choose if the drone should return to launch after the surveying mission. If yes, the operator can make a decision if the drone should land autonomously on the configured home position or it should loiter over the home position.

2. Autonomous guided missions : Surveying a particular area autonomously requires a properly defined flight plan. The operator has to draw a polygon over the geological area to be mapped in the MMI. The operator also has to provide basic camera parameters during the initial configuration of the mission. Based on the camera parameters – (1) Image resolution, (2) Sensor dimensions, (3) Focal length of the lens, & (4) Flight height, the Ground Sampling Distance (GSD) is computed. Once the GSD is known, a proper route can be computed, along with an estimation of the total time for the mission to complete and the total number of images that will be captured within the same duration. The guided mission algorithm also computes the distance at which an image should be clicked to maintain the overlap requirements.

3. Return-to-launch (failsafe) : The UAV should be capable of returning back to the configured home position in case of any system failure or communication losses. The UAV has been configured to return-to-launch in case of low-battery situations where the battery level falls below a threshold of 25%. The RTL altitude for the UAV has been configured at +15m with respect to the starting point.

4. Gyroscopic image stabilisation : The on-board camera should be stabilised with the help of the gyroscope on board. Image stabilisation allows us to increase the

aperture by a few points, thereby increasing the detail captured by the camera lens. This, however, is not required if the camera supports image stabilisation as its core functionality.

5. Geo-tagged image acquisition & storage : Geo-location of each image plays a very important role in the to-scale reconstruction algorithm that has been employed. Without the geo-location tags within an image, the reconstruction algorithm may suffer during the calculation of odometry and the geo-referenced pointcloud that is used to generate the orthophoto, and the digital elevation models.

3.1.2 Understanding the Mission Management Interface(MMI)

The MMI sub-system defines a web based interface dedicated to creation, modification and archival of missions. The MMI extends its functionality by providing a set of visualisation tools for the products generated by the RPN. The core functionalities of the MMI are listed below.

1. Mission Management (Creation & Modification interface) : A graphical user interface dedicated to organising missions and to modify mission parameters during or after the the mission is under progress. The mission creation module offers two types of missions – (1) Full Mission : Full missions require a a complete flight for image acquisition. (2) From Image : The software can process images that have been captured on separately conducted flights.

2. Flight Planner : A comprehensive flight planner : The flight planner provides an option to configure the home location and guides the user to draw a polygon around the area to be mapped. The flight planner then computes an optimum flight route and image capture distance and uploads the mission to the drone.

3. Mission browser (For viewing information about old missions) : The mission browser provides a GUI interface to view the photos, orthomaps, 3d models, digital elevation models, faces and weapons for any mission that has completed processing. If a particular mission is under processing, the mission browser shows the processing log until the processing is complete. The options to view the data in a 3d model viewer or on a map are unlocked once the processing is completed. Likewise, when a particular mission is in the "FLIGHT" stage, the mission browser shows flight instruments that are updated in real-time with respect to the actual flight.

4. Orthophoto, DSM, DTM visualiser : The MMI hosts the orthophoto as a set of geological tiles. The tiles are loaded using a Tile Map Service [@43] which hosts the files in a manner to leverage the REST features.

5. 3d model visualiser : The MMI makes use of Google's `model-viewer` JavaScript library to render 3D mesh models within a browser. The operator does have an option to load the model into an advanced object viewer such as Blender or MeshLab.

6. AR playground : Similar to the 3D model visualiser, the AR playground is generated by the `model-viewer` library. However, for AR models to load, the `model-viewer` library requires them to be hosted via a certified secure server (certified HTTPS).

7. Face & weapons database viewer : The face and weapons database viewer provides a GUI tool for viewing all the faces and weapons that have been clustered in the previous missions. The database viewer adds an option to add a label to the face detected (for example, a real name can be associated to an automatically generated unique ID for a face).

3.1.3 Understanding the Remote Processing Node (RPN)

The RPN sub-system defines a REST API build around three main nodes - Aerial Mapping Node (for generating 3d point cloud, orthophoto, DTM, DSM and 3d mesh ; Face Clustering Node (for clustering and identification of faces with a common database) ; Weapon Recognition Node (for identifying weapons (if any) in the sequence of images. Main functions of the RPN are as follows. The RPN is built separately from the client interface such that the RPN can be hosted separately on a much more powerful server than the client interface.

1. REST API : A Representational State Transfer Application Program Interface built around the pipeline to provide remote access to all the algorithms to generate the products listed below.

2. 3D reconstruction (point cloud & mesh) : The 3D reconstruction pipeline makes use of a sequence of overlapping 2D images. The pipeline first performs SIFT feature extraction and then runs a RANSAC transform loop between consecutive image pairs to generate a 3D projection of each point in the 3D space. The set of 3D projects are then converted into one large pointcloud using the multi-view stereo. The Poisson mesh reconstruction algorithm is then applied on the pointcloud to generate a textured mesh.

3. Orthophoto generation : With the sparse and dense reconstruction with SfM in PLY format, we post-process the point cloud to generate a geo-referenced point cloud in LAS format, a geo-referenced 3D textured mesh and an orthophoto mosaic in the GeoTIFF format. The LAS format is a standard binary format for the storage of LIDAR data and point clouds. The geo-referenced pointcloud, along with the original views is used to compute the orthophoto.

4. DSM generation : The pointcloud in LAS format is directly used to generate a 2D projection of the 3D points creating the Digital Surface Model.

5. DTM generation : The LAS pointcloud is filtered for objects such as buildings, trees, humans, cars, etc in order to obtain a precise terrain model.

6. Face clustering : A special face recognition algorithm is implemented which runs on the principle of face clustering rather than face identification. The fundamental difference between a standard classification algorithm and a clustering algorithm is that the classification algorithm does provide an online mechanism to add new identities.

7. Weapon recognition : A standard object detection framework [33] has been retrained to classify between two different weapons – (1) Pistol & (2) Knife. The dataset consists of 2,000 annotated images and a new model has been trained by completing 20,000 iterations using the darknet training API.

3.2 Hardware Description

Carrying out multiple tasks in real-time requires enormous amount of computing power. The project introduces several layers of hardware. The hardware requirements for the proposed pipeline can be categorized into three main types – (1) Vehicles, (2) Controllers & (3) Sensors.

3.2.1 Vehicles

The project proposes the use of a quad-copter running on auto-pilot to perform the information extraction procedure. The entire project setup will be present on a quad-copter. For powering the drone, four 930KV brush less DC motors have been used (2 clockwise, and 2 anti-clockwise) along with 10" propellers. The motor setup is as shown in *Figure 3.3*. A few technical specifications with respect to the functioning of the quad copter are as follows.

1. **Communication Systems** There are three main communication systems on board. Each wireless communication link serves a distinct purpose.

    1.1. **FS-iA6B 2.4Ghz communication link** The FS-iA6B serves as the main link between the PixHawk 2.4.8 Flight controller and a control remote dedicated to the quad copter. The specified range for this module is up to 1km.

    1.2. **433Mhz secondary telemetry module**

    The secondary telemetry serves as a failsafe communication link in case the primary 2.4Ghz link is lost. This telemetry link also provides support for in-flight logging and diagnostics. Additionally, the secondary telemetry link is used for uploading mission flight plans on-to the flight controller.

    1.3. **WiFi @ 2.4Ghz**

    An additional WiFi module aids the commmunication between the Jetson Nano Developer Kit and the host computer. This link is useful for diagnosing any issues with 3D scene reconstruction, weapon recognition or face recognition.

2. **Power Control and Distribution**

    To power the motors, four 30A Electronic Speed Controllers are used. The quadcopter makes use of a printed Power Distribution Board (PDB) for distribution of power to each motor. The in-flight stabilization and radio control is provided by the on-board PixHawk 2.4.8 flight controller. An APM module has been used for distributing power from battery to motors and the flight controller. Figure 3.2 depicts the schematic for power control and distribution.

3. **Positioning systems** The quad copter also makes use of Neo 7m Global Positioning System module for auto-pilot mission modes. A few more sensors and their utilities have been discussed in the *D.2.3 Section*.

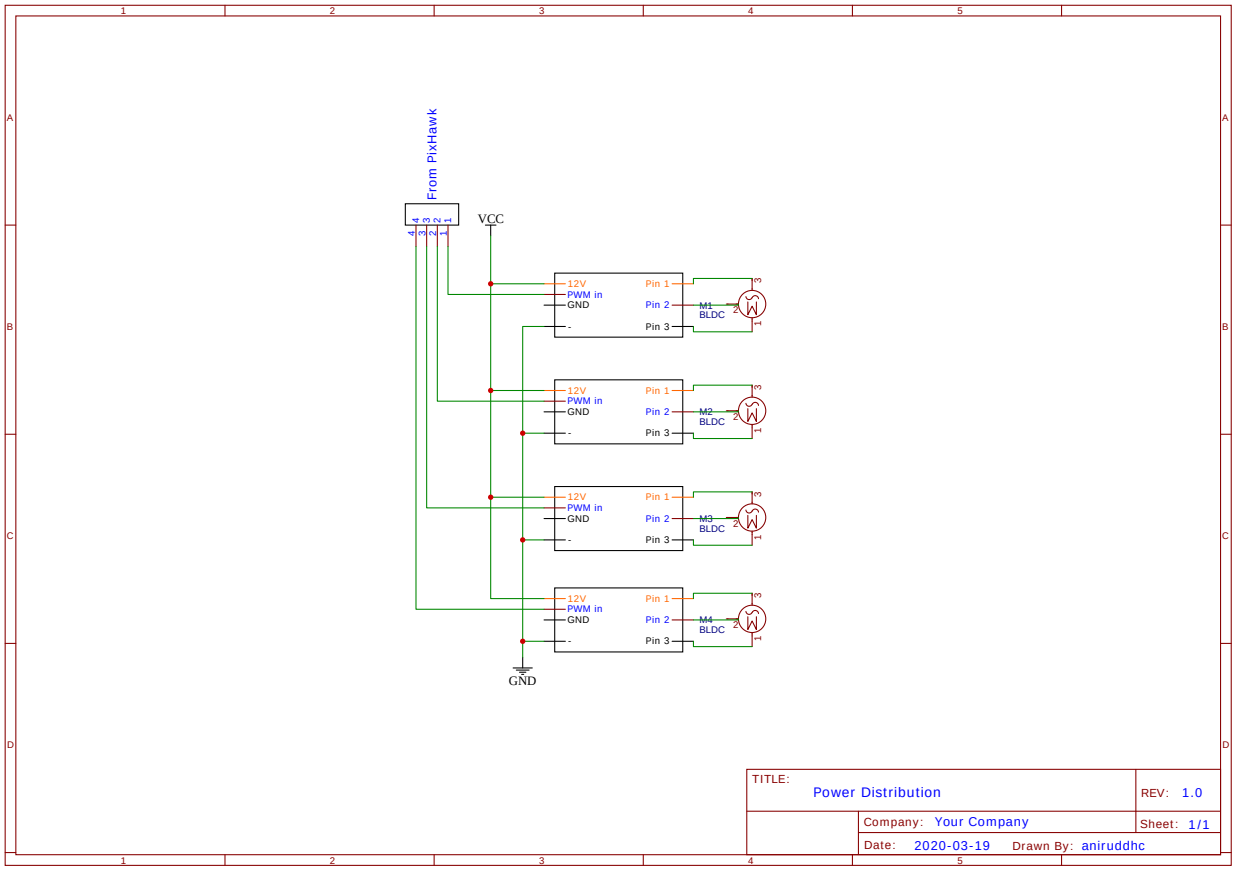

Fig. 3.2.: Power control and distribution board

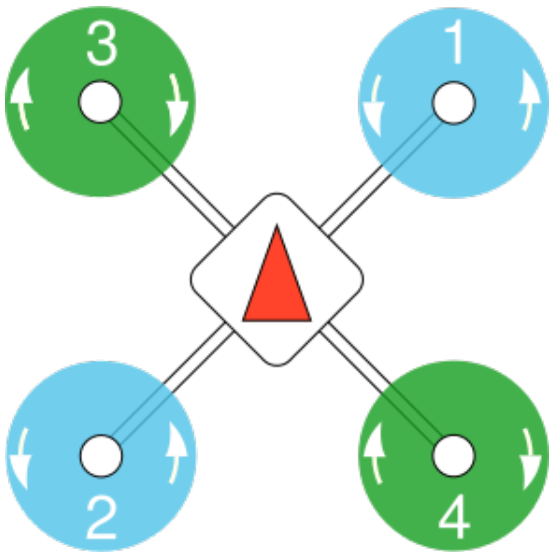

Fig. 3.3.: Quad Copter Motor Setup

4. **Quad Copter Dimensions** The *Figure 3.4* provides with dimensional specifications of the UAV. Currently, the UAV weighs around 1.0kg along with all peripherals.

3.2.2 Controllers

The controllers are responsible for all the computing processes on board. The project makes use of two controllers, working together in order to achieve the goal. One of the controllers is responsible for running 3D reconstruction, weapon recognition and face recognition, while the other controller is used for stabilizing and flying the UAV in autonomous mode.

1. **Jetson Nano Developer Kit**

   The Jetson Nano developer kit is a small, powerful computer that lets us run multiple deep learning networks in parallel. The Jetson Nano development board used for performing 3D reconstruction, Weapon Recognition as well as Face Recognition. Some technical specifications are as follows.

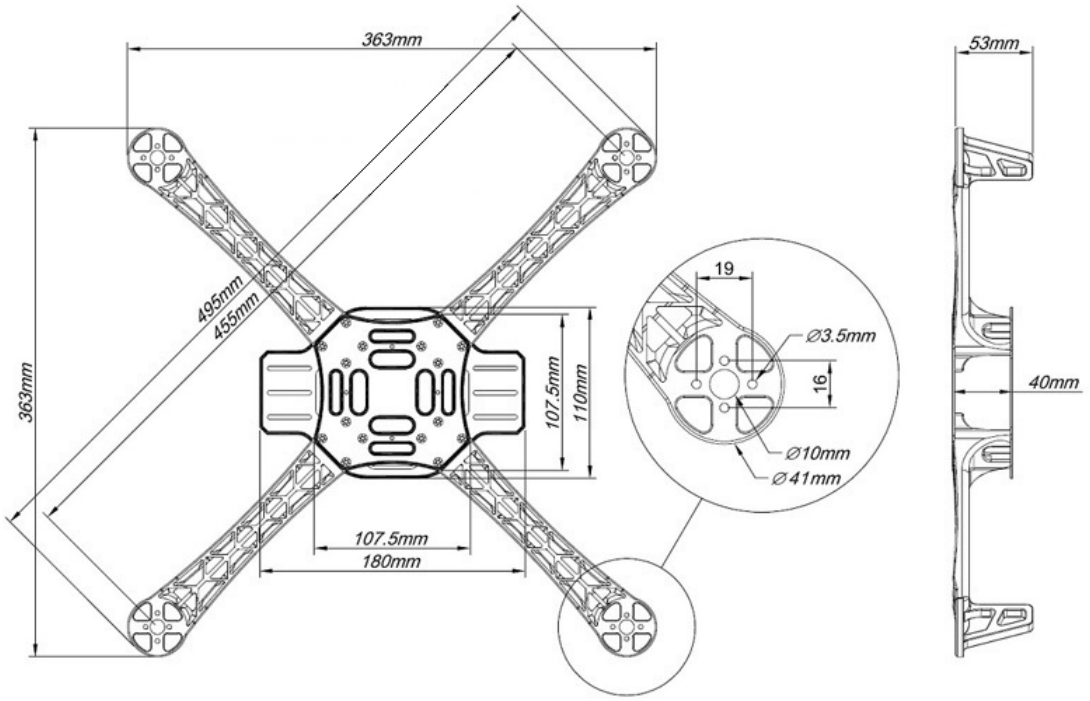

Fig. 3.4.: Quad copter dimensions

1.1. **CPU** : Quad-core ARM A57 @ 1.43 GHz

1.2. **GPU** : 128-core Maxwell architecture

1.3. **Memory** : 4 GB 64-bit LPDDR4 25.6 GB/s

1.4. **Video Encode** : 4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)

1.5. **Ports** : GPIO, I2C, I2S, SPI, UART

Jetson Nano does not come with an inbuilt WiFi adapter. However, we have used a D-Link 150N USB WiFi adapter for performing wireless communication. Jetson Nano only supports an SD-Card as its primary disk. However, the tasks that are to be performed in this project are CPU, GPU and Disk intensive and an SD card can wear out in the process. To overcome this, we have modified the kernel image on the Jetson Nano to boot from SDCard initially, and then to switch over to an external SSD for the remaining tasks. Overall, the Jetson Nano can offer upto 472 Giga Flops of computing power, which is sufficient for this project.

2. **PixHawk 2.4.8 32bit Flight Controller** Quadcopters are mechanically simple. By controlling the speeds of the four propellers, the quad-copters can roll, pitch, yaw and accelerate along the common orientation. Quad-copters are extremely agile, but the agility comes at a cost - quad-copters are inherently unstable, and require automatic feedback control in order to be able to fly. The PixHawk 2.4.8 provides estimation and control algorithms for the quad-copter to maintain stable flight. A few technical specifications for PixHawk are as follows.

    2.1. **Processor** : 32bit STM32F427 Cortex M4 core with FPU

    2.2. **Co-Processor** : The 32-bit STM32F103 failsafe Co-processor

    2.3. **Firmware** : PX4 RTOS

    2.4. **On-board sensors** : 3-axis gyrometer, Accelerometer, Magnetometer, Barometer

    2.5. **Input Voltage** : 7V

    The PixHawk Flight controller can be used with ArduCopter: Multirotors as well as the PX4 Flight Stack. However, the project proposes the use of PX4 flight stack due to support for extensibility.

3.2.3 Sensors

The proposed system requires multiple sensors in order to capture crucial information and maintain stable flight throughout the operation. The sensors can be mainly categorized into two types.

1. **Internal Sensors**

The PixHawk 2.4.8 flight controller comes with a set of sensors build right into the mainframe. The functionality of each sensor is as follows.

1.1. **3-axis gyroscope** Gyroscope sensor detects angular velocity in three axis. So it can detect rate of change of angle in pitch, roll and yaw. Gyroscope is a critical sensor even in regular craft. The change in angle information is used to provide stability to drone and to prevent it from wobbling. The information from gyroscope is fed to motor control drivers to control the speed of motors dynamically to provide the stability to motor. Gyroscope also ensure that drones rotate at exact angle which is expected by user controls.

1.2. **Accelerometer** Accelerometer is used to provide the acceleration force which the drone is subjected to in all three axis X, Y and Z. It also determines the tilt angle of drone in stationary position.

1.2.1. If the drone is stationary in horizontal position then its X and Y axis will give 0g output whereas z-axis will give 1g output. 1g is the gravity which is experienced by every object on earth.

1.2.2. If the drone rotate by 90deg on X axis then X and Z will give 0g and Y axis will start will start giving 1g. During the tilt X, Y and Z will give output which lies between 0 and 1g. The values can then be applied to trigonometry formulas to arrive at tilt angle of drone.

Accelerometer are also used to give linear acceleration in horizontal and vertical direction. This data can be used to calculate velocity, direction and rate of change of altitude of the drone. Accelerometer is also used to detect the vibration which the drone is experiencing.

1.3. **Barometer** Barometer working principle is to convert atmospheric pressure into altitude. Pressure sensor can detect earth's atmospheric pressure. The data from Barometer helps in drone navigation and achieve desired altitude.

Very good estimation of ascend and descend speeds is very vital for drones flight control. STMicroelectronics has introduced pressure sensors, LPS22HD, with 200Hz of data rate to address this requirement of altitude estimation.

1.4. **Magnetometer** Magnetic compass as the name suggests gives the sense of direction to the drone. It gives data of magnetic field in three X, Y and Z which the device is subjected to. This data is then fed into algorithm in the microcontroller to give heading angle w.r.t magnetic north. This information is then used to detect geographical directions.

2. **External Sensors** The proposed system also requires a few external sensors for the auto-pilot and the 3D reconstruction.

2.1. **Global Positioning System** A GPS receiver offers methods for computing the real-time geological location of the UAV. In the current context, the GPS module plays an essential role as it is actively used by the autopilot to navigate itself in the physical environment during a guided mission and during the return to launch mode. The reconstruction system also requires GPS to geo-tag images that will be used for mapping and reconstruction.

2.2. **Pi Camera V2** The reconstruction system requires a high resolution camera for the best results. Ideally, a camera with a much larger sensor should be used for mapping, however, due to project costs, we chose a Pi Camera V2. The PiCamera offers the following specifications :

2.2.1. Still Resolution : 8MP

2.2.2. Sensor Resolution : 3280 x 2464

2.2.3. Sensor Image area : 3.68 x 2.76 mm

2.2.4. Sensor (Pixel) Size : 1.12 µm x 1.12 µm

2.2.5. Focal length : 3.04mm

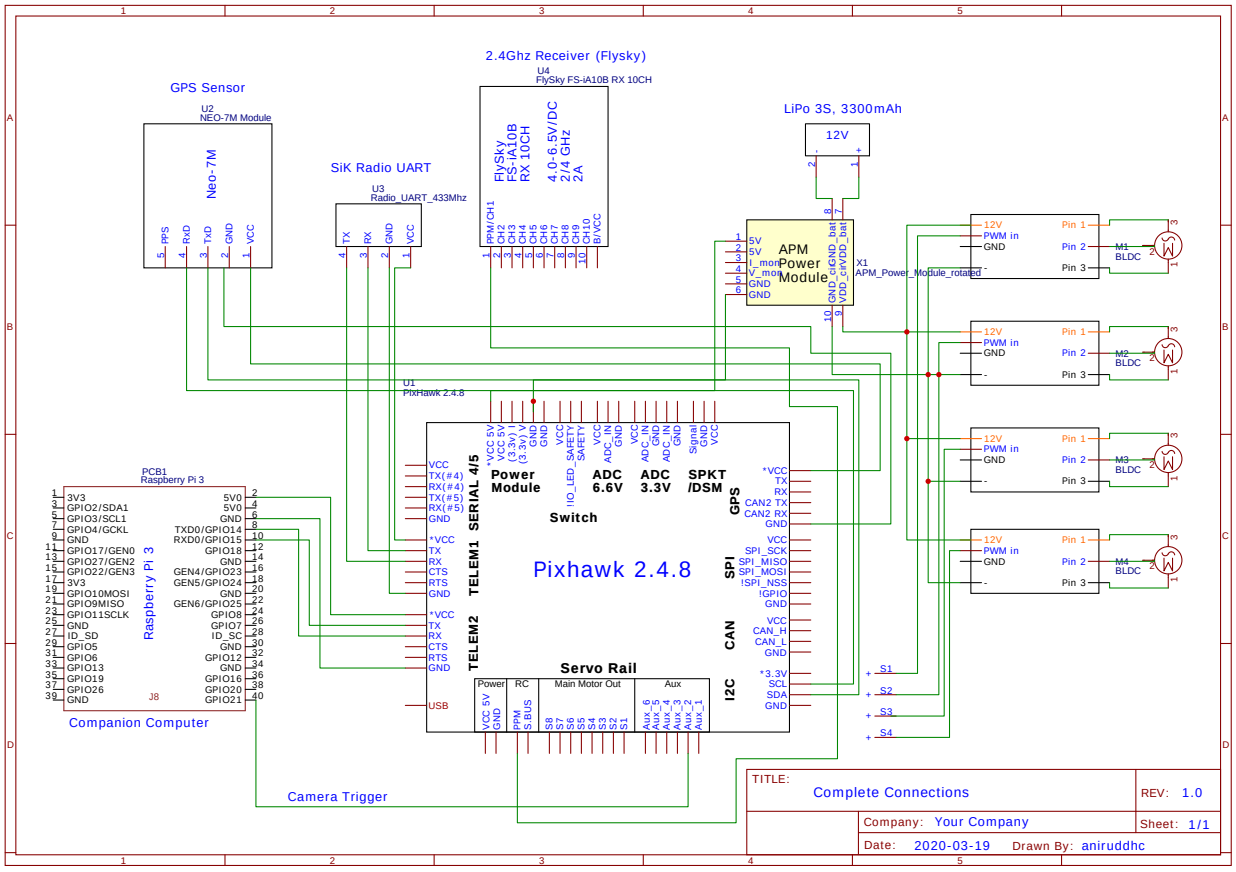

Fig. 3.5.: Schematic : Complete electronic connectivity on the UAV

2.2.6. Horizontal field of view : 62.2 degrees

2.2.7. Vertical field of view : 48.8 degrees

3.2.4 Full Schematic (UAV)

Figure 3.5 shows a detailed schematic depicting the connectivity of all the electronic components on board the UAV. PixHawk 2.4.8 is the master controller on board and is equipped with the PX4 autopilot firmware stack to run control and estimation algorithm to maintain stability throughout all the flights. The PixHawk FC has been configured to make use of the "TAKEOFF", "GUIDED", "AUTO", "RTL" and "LAND" flight modes for executing any mission.

The PixHawk flight controller has an inbuilt accelerometer and gyroscope for pitch and roll adjustments. An inbuilt barometer is used for altitude measurements. A external Ublox Neo-7m GPS module is connected via I2C interface to the main FC. The PixHawk offers 2 telemetry ports and one serial port for communication using the MavLink protocol. The TELEM 1 port has been connected to a 3DR SIK Radio for transmitting real-time telemetry information to any ground control application. The SIK radio makes use of the 433Mhz radio frequency band for communication.

A 3 cell (series) [3S] Lithium-Polymer battery with a capacity of 3300mAh has been used to enable longer flight time. However, the PixHawk controller requires a stable 5V supply for optimum operations. In high performance UAVs, the motors are supplied with burst current at peak capacity for providing as much thrust as possible. However, a sudden burst in the current can cause serious damage to the flight controller. To maintain a constant voltage and constant current line to the flight controller, the APM Power module with XT60 connector is used. The APM power module takes in a 12V supply as an input and provides a 5V power supply and a 12V power supply. The 5V power supply line is protected from any current burst that may occur during the flight operation.

The UAV makes use of Brushless DC (BLDC) motors for providing the required thrust. A brushless DC motor (known as BLDC) is a permanent magnet synchronous electric motor which is driven by direct current (DC) electricity and it accomplishes electronically controlled commutation system (commutation is the process of producing rotational torque in the motor by changing phase currents through it at appropriate times) instead of a mechanically commutation system. BLDC motors are also referred as trapezoidal permanent magnet motors. BLDC motor works on the principle similar to that of a conventional DC motor, i.e., the Lorentz force law which states that whenever a current carrying conductor placed in a magnetic field it experiences a force. As a consequence of reaction force, the magnet will experience an equal and opposite force. In case BLDC motor, the current carrying conductor is stationary while the permanent magnet moves.

An electronic speed controller circuit is required to energize appropriate motor winding by turning transistor or other solid state switches to rotate the motor continuously. Usually, Hall-effect sensors are used to gain precise information about the position and speed as a feedback. In the current context, the UAV is equipped with four 30A SimonK Electronic Speed controllers. Each of the ESC is dedicated to controlling the speed and direction of each of the brushless motor.

One of the fundamental functions of the UAV is to capture high resolution images. For this purpose, a companion computer is required, since the PixHawk does not provide for any on-board image storage or acquisition solution. The project makes use of a Raspberry Pi v3 B+ as a companion computer. The Pi Camera V2 is connected to the Raspberry Pi using the CSI Camera Lanes. The Raspberry Pi receives real-time telemetry information from the PixHawk using the TELEM 2 port. The VCC and GND supply lines from the TELEM 2 port are used to power the Raspberry Pi without the need of an external power bank.

A GPIO pin on the Raspberry Pi is dedicated to function as the camera trigger. During the boot-up process of the Raspberry Pi, a script is run as a daemon which locks the camera and makes it ready for use. The same script monitors for electronic triggers on the GPIO21. During the mission, the PixHawk sends several electronic triggers to Raspberry Pi to capture pictures. Whenever a trigger is used, the telemetry data from the PixHawk is read serially and the captured image is Geo-Tagged.

3.3 Software Description

Section 3.1 provides a holistic overview and understanding of each objective to be achieved. The software architecture strategically segregates these objectives into a nexus of smaller nodes. The holistic overview lists down three main sub-systems. Each of these sub-systems runs a dedicated web-server implemented in Python (Flask) to provide

a REST API around the algorithms applied. The following sub-sections provide an in-depth explanation regarding the working and implementation of the three sub-systems, along with screenshots and flow diagrams wherever applicable.

3.3.1 Software implementation on the Unmanned Aerial Vehicle (UAV)

Section 3.2, figure 3.5 outlines the hardware connectivity between the PixHawk flight controller and the companion computer, Raspberry Pi. The Flight Controller FC communicates with the Companion Computer (CC) using a serial interface at a baud rate of 921600 (921Kb/s). On this link the FC transmits the real-time telemetry data to the CC. The telemetry data includes the values of all senors, such as the altitude, GPS coordinates, compass heading, pitch, roll, etc. However, due to software limitations, a serial socket can only be read by one program at a time.

To solve the socket limitaion issue, a MAV Proxy server is used. MAVProxy is a fully-functioning GCS for UAV's. The intent is for a minimalist, portable and extendable GCS for any UAV supporting the MAVLink protocol. MAVProxy was first developed by CanberraUAV, to enable the use of companion computing and multiple datalinks with ArduPilot.

Some of the features of MAV Proxy are as follows :

1. Is a command-line based router application.
2. Provides with an option to host the incoming data on several UDP ports.
3. Completely portable and can run on any POSIX OS using python and pyserial.

The `Dronekit-Python` library is then used to make a connection from the driver code to the MAXProxy UDP port. This establishes a Command/Response interface to communication with the FC using Python Code.

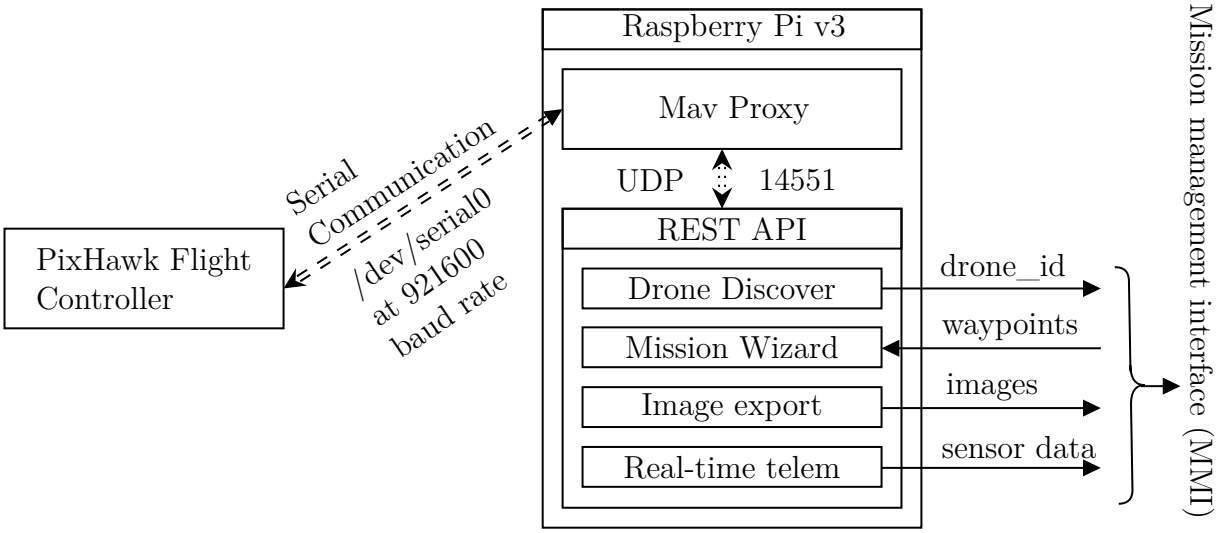

Fig. 3.6.: Software implementation on UAV

The UAV can now be controlled with a set of Python Commands. However, it is necessary to create a comprehensive interface to perform the following tasks.

1. Load and fly missions that are created using the MMI
2. Acquire Images and store them with reference to the mission unique identifier
3. Upload the images from a particular mission back to the MMI after landing at the RTL position
4. Transmit real-time telemetry and sensor information back to the MMI for visualisation in the Virtual Cockpit.

Figure 3.6 depicts the different blocks in the UAV software implementation. The REST API has been implemented in Python using the `Flask` micro-framework. The REST API is run on a TCP port `4000`. The web-server is bound to the `0.0.0.0:4000` in order for it to be accessible from outside localhost.

1. Drone Discovery Protocol (DDP) : (API Route : `/`, Method : `GET`) During the configuration process of an UAV, it is assigned an unique ID. Whenever a full mission is to be planned (on the MMI), a `nmap` process (on the MMI) scans for all

the devices on the local network and shortlists the devices with the port `4000` open. A `GET` request is sent to each of the device with an open `4000` port. Each drone responds to the request with its own ID. The response body format is :

```
{'drone_id': '<unique_drone_identity_here>'}
```

Listing 3.1: Response structure of the Drone Discovery Protocol

2. Mission controller : (API Route : `/start_mission`, Method: `POST`)

   The mission controller is used to start a particular mission from the MMI. The MMI has to send a `POST` request with the following payload.

```
{
    'uuid': '5ea35d78-26cd-4185-82dd-29112b1935ec', // mission id
    'waypoints':  [
        [
          "72.83658485518518", //longitude
          "19.109608142280223", //latitude
          10 //altitude
        ],
        [
          "72.83656876193095",
          "19.109040434971803",
          10
        ],
        [
          "72.83713739024202",
          "19.10911139849145",
          10
        ]
      ], // mission waypoints
}
```

Listing 3.2: Request payload for starting mission flight

3. Mission Flight : The mission flight is run as a concurrent thread and is connected to the drone via MAVProxy. Mission flight completes the flight in the GUIDED flight mode using the waypoints that are received from the mission controller. Once the UAV lands on the RTL position, the mission flight manager reports the status to the mission controller and all the images are uploaded from the Raspberry Pi to the MMI under the correct `mission_uuid`.

4. Telemetry export : (API Route : `/get_drone_location`, Method : `GET`)

   The telemetry export module responds to any request with the real-time sensor parameters from the Flight Controller. The response format is as follows.

```
{
    'lon': FloatField(),
    'lat': FloatField(),
    'alt': FloatField(),
    'heading': IntField(),
    'groundspeed': FloatField(),
    'roll': FloatField(),
    'pitch': FloatField()
}
```

Listing 3.3: Response from the telemetry export

3.3.2 Software implementation on the Remote Processing Node (RPN)

Figure 3.7 depicts a general block diagram of the Remote Processing Node. The core functionality of the RPN is to generate an ortophoto, DSM, DTM, 3D Mesh, a cluster of faces and a list of weapons detected. generation of an orthophoto, 3D mesh and DEM are inter-related.

Figure 3.8 depicts the stages of a reconstruction pipeline, which all images have to pass in order to get a high quality, dense 3D reconstruction presented as a point cloud. The

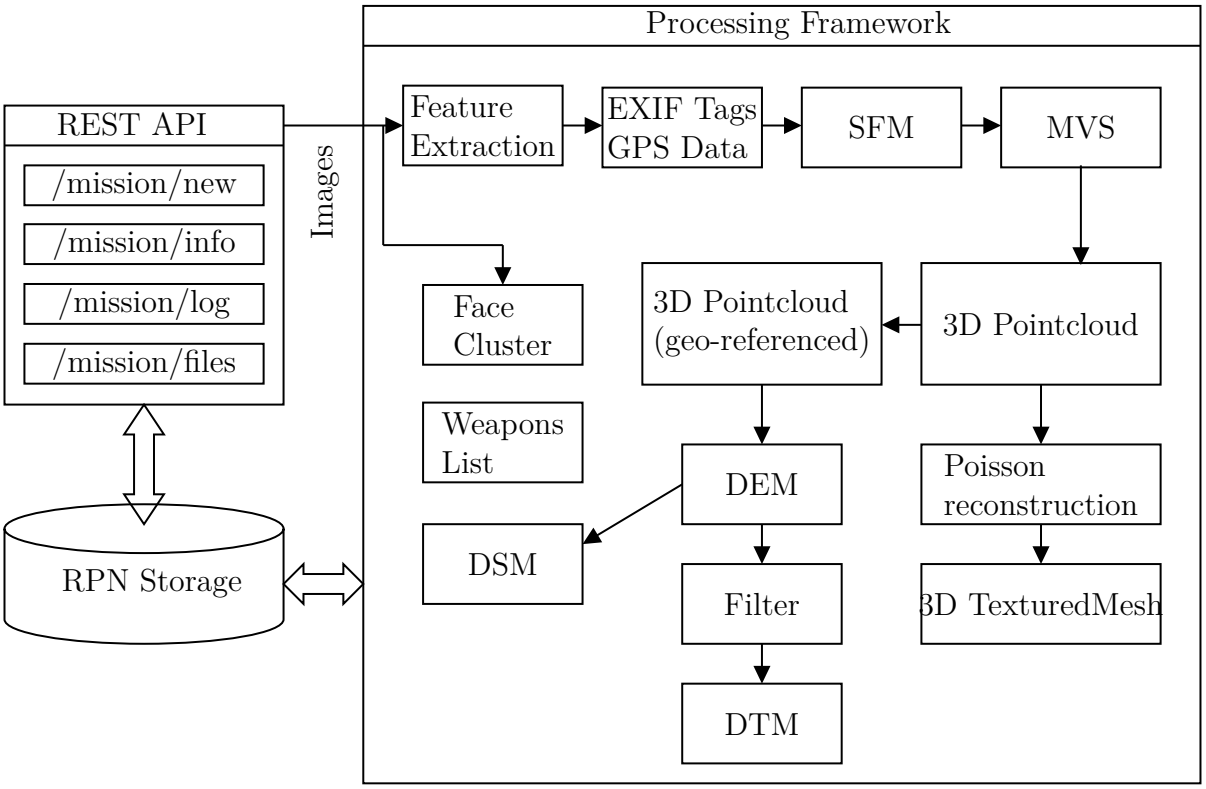

Fig. 3.7.: Block diagram of the Remote Processing Node

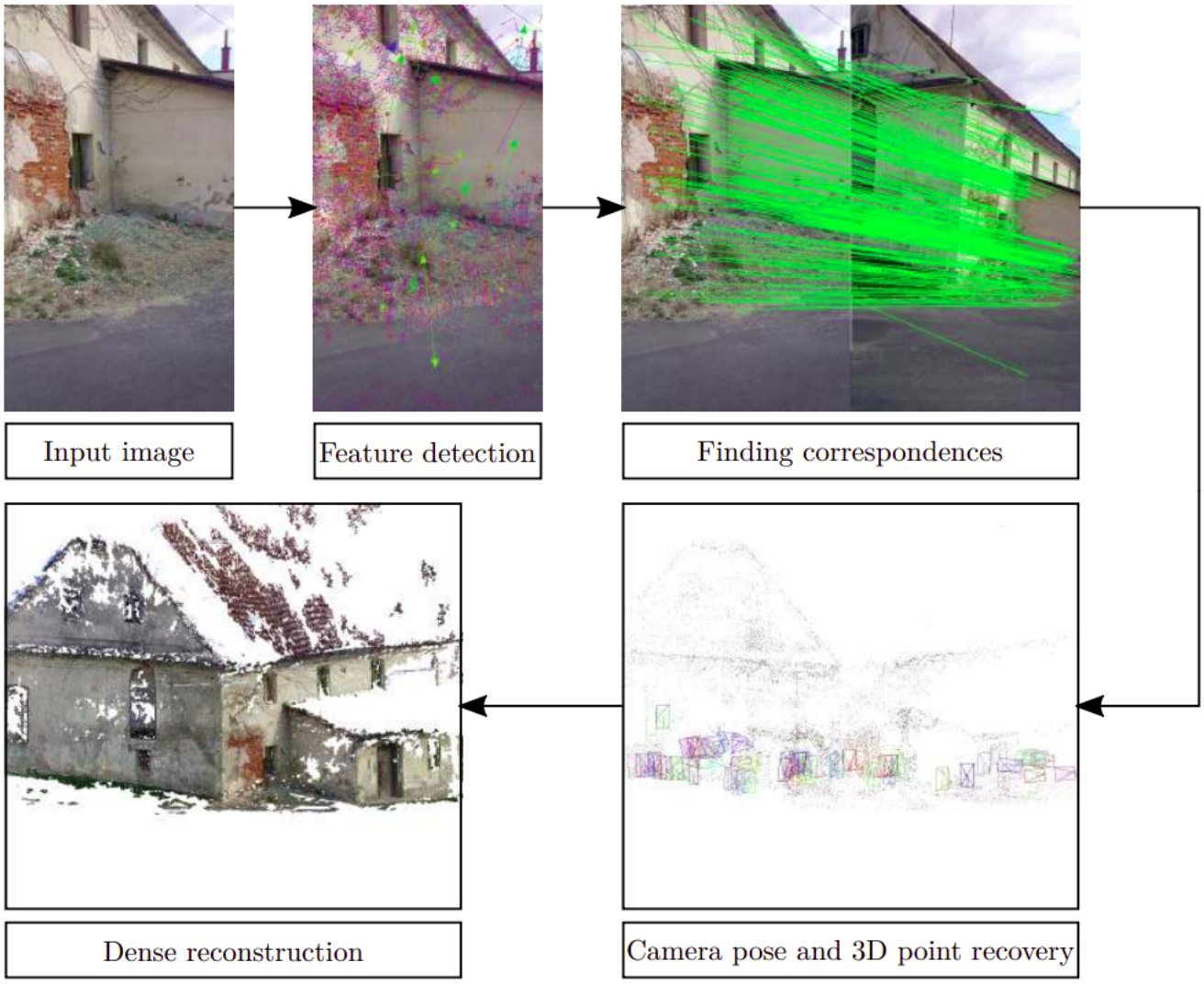

Fig. 3.8.: The reconstruction pipeline: Each image passes through the stages - feature detection, feature matching, camera pose detection (motion), sparse reconstruction (structure) and dense reconstruction (MVS).

implementation of the reconstruction pipeline can be explained through the following four stages :

1. Feature detection The first step of the pipeline is the feature detection step, where distinctive image interest points are recovered, that can be subsequently used in the next step to find relations between different images. There exist various feature detectors and they have been explained in the section 2.

   One of the most prominent detectors is the SIFT algorithm. SIFT is an abbreviation for scale-invariant feature transform, which implies that the SIFT algorithm delivers feature descriptors that are not affected by image transformations like rotation or scaling. SIFT finds features placed at the minima or maxima of a Difference-of-Gaussians (DoG) function that are present at multiple scales. For each feature, a description vector with 128 elements is computed.

2. Finding correspondences, motion and structure After identifying the features, they can be used to find feature correspondences between the images, which are then used to recover the relative camera positions and orientations(motion) and locations of the 3D points (structure). This stage of the pipeline applies a GPS filter where in correspondences are only computed among images present in a 15m region. This filter not only reduces the number of pictures to be processed, but also increases the matching and odometry accuracy when the area to be mapped has similar architectural buildings. Relations between the features in image pairs can be found by using a k-nearest neighbour classification in feature space. The pair-wise matches are optimised using a RANSAC (Random Sample Consensus) loop, where in each iteration computes a fundamental matrix describing the relation between the two views. The original matches are then verified by using the recovered matrix and the outliers which are not geometrically consistent are removed.

   The fundamental matrix F is a 3x3 matrix describing the epipolar geometry of two camera views (Refer figure 3.9. A projection x of a 3D point X in one image

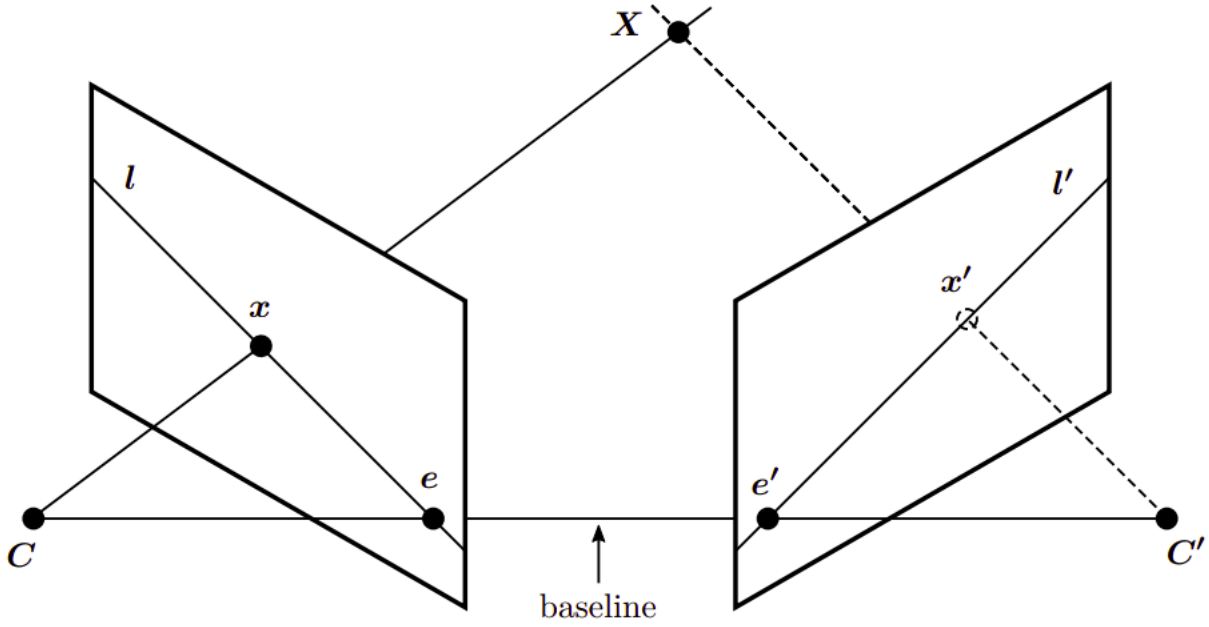

Fig. 3.9.: Epipolar Geometry: The epipoles e and e' are defined by the intersections between the camera baseline and the image planes.

has its corresponding projection x' in the other image on epipolar line l'. The epipolar geometry and therefore the F matrix is independent of scene structure and depends just on the camera's intrinsics and their relative position and orientation. If a projection x has its own corresponding x' in the other view, then the epipolar constraint (refer eqn 3.1) is satisfied.

$$x'^T F x = 0 \tag{3.1}$$

Having determined F, it is subsequently used to reocver the camera's extrinsic and intrinsic parameters, and therefore their projection matrices P and P' by the means of triangulation (explained in 2).

Usually, the process of computing structure and motion out of images follows an incremental approach. At first the cameras and 3D points are calculated using an initial pair of input images. Subsequently, one image after another is added to the

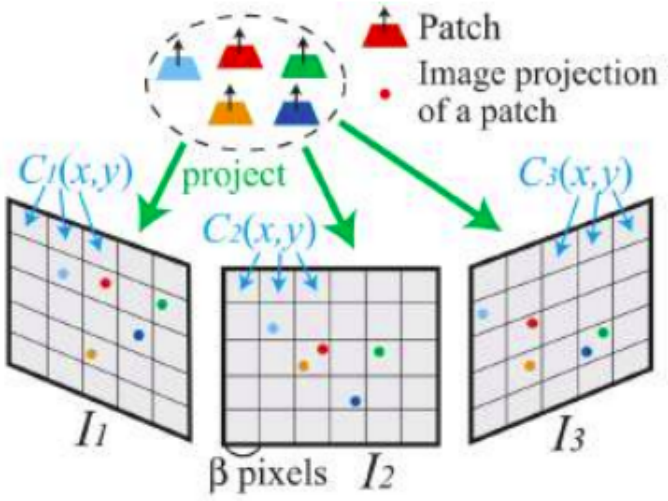

Fig. 3.10.: Patch projection into image cells. The goal of the algorithm is that each cell contains atleast once patch projection.

existing reconstruction until all photos are processed. After each incrementation, an operation called bundle adjustment is performed.

3. Dense reconstruction using MVS Different MVS approaches can be classified into four categories – (1) Vovel based, (2) Polygon mesh based, (3) Multiple depth map based & (4) Patch based. The reconstruction algorithm makes used of a patch-based multi-view stereo (PMVS) algorithm that is able to reconstruct large scenes in an accurate and complete manner. The PMVS algorithm is a three-step algorithm that first generates a patch based scene reconstruction which is subsequently converted into a polygonal mesh and at last refined by a mesh based MVS algorithm. Figure 3.10 shows the patch projection algorithm that is used by the first stage of the algorithm.

   MVS converts a sparse reconstruction into a dense reconstruction and has to perform an operation known as patch expansion for the same. The previously matched features and corresponding sparsely reconstructed 3D points called "seeds" are ordered into a priority queue accoridng to their similarity. This queue is processed sequentailly, such that the best placed seed always comes first. Each of the seeds

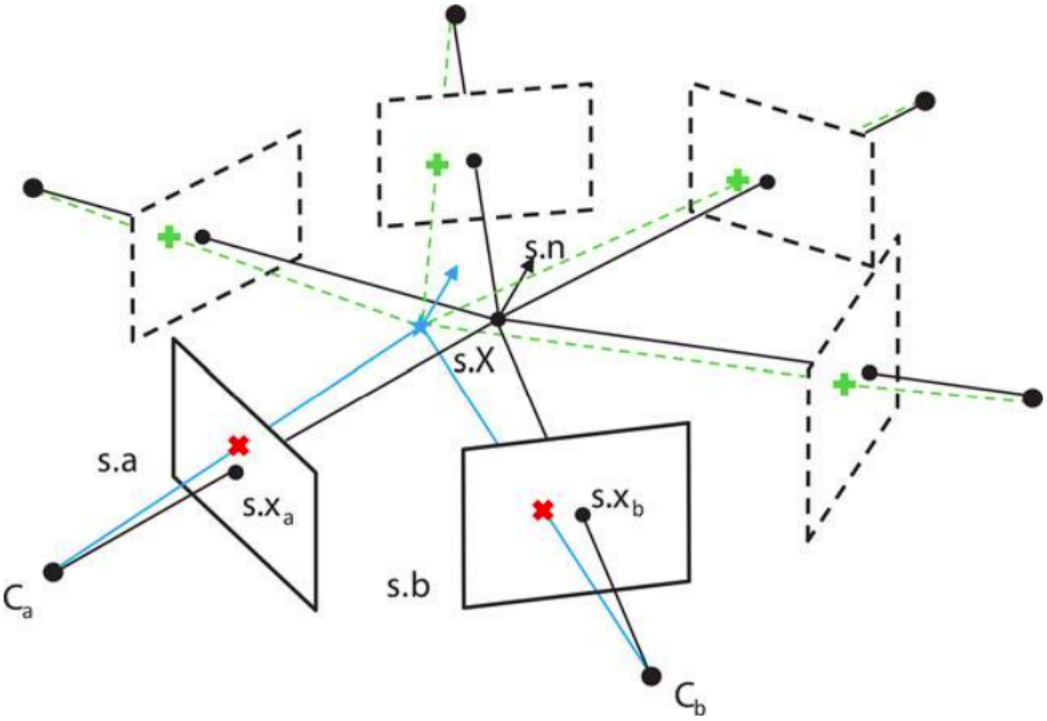

Fig. 3.11.: Patch expansion process

in the queue is expanded. The expansion process is depicted in figure 3.11, where s describes the currently processed seed. To expand it, the method searches for new correspondence pairs in the neighbourhood of the current seed in the corresponding reference views. A new 3D point is reconstructed by the means of triangulation. At last the newly created point is projected into all other images and matches are evaluated.

The RPN inlcudes a face clustering node for recognizing existing faces and creating new identifiers for new faces. The basic block diagram of the face clustering node is depicted in figure 3.12.

Each image is passed through the face clustering node. The clustering algorithm progresses in three stages.

1. Face detection : A simple face detector script first extracts all the faces from the still photograph using the `dlib` face detection algorithm. At the end of this stage,

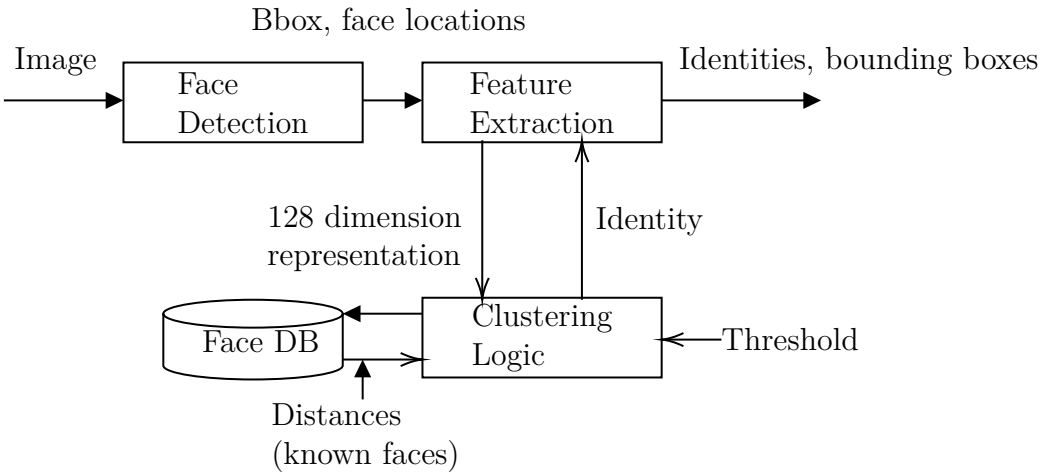

Fig. 3.12.: Block Diagram of the Face Clustering Node

each face within the image has a bounding box associated with it. The bounding box provides the pixel boundaries for each face.

2. Face encoding : By using a dlib feature descriptor, a 128 dimensional representation is created for each face in each image.

3. Cluster Logic : The face encoding provides a 128-dimensional vector for each face. The The clustering logic stores the face encoding in the form of a 128-dimension embedding and associates each point with a class label. The clustering logic first compares the image with all the existing data points within the embedding and computes the distance between the query point and existing point. It then computes which data point is the closest to the query point. If the distance between the query point and the closest point is below a preset threshold, then the new query point is classified according to the closest point. If the distance is greater than the threshold, then a new unique label is assigned for that particular face. At the end of the clustering logic, the identities and the bounding boxes are returned.

The weapon recognition node is a simple object detection module which detects and classifies objects within an image frame. The Yolo V3 algorithm is used for weapon recognition.

The RPN REST API has been implemented using Python (`Flask`) and makes use of the TCP `3000` port. The application binds to the `0.0.0.0:3000` to open access from outside localhost. The REST API exposes four routes :

1. Route : `/mission/new`, method : `POST` : The MMI has to send a POST request to the `/mission/new` endpoint with all the images under the `images` key value. The MMI also has to send the `mission_uuid` along with the images for reference on the RPN.

2. Route : `/mission/info`, method : `GET` : This endpoint requires a `uuid` parameter. It returns the current status of the mission processing progress.

3. Route : `/mission/log`, method : `GET` : This endpoint also requires the `uuid` parameter. It return the processing log of a mission under process. The processing logs are saved even after the reconstruction process is completed.

4. Route : `/mission/files`, method : `GET` : This endpoint requires the `uuid` parameter and returns all the generated files into one compressed archive.

3.3.3 Software implementation on the Mission Management Interface(MMI)

The MMI acts as the interaction point between the UAV, the RPN and the operator. The operator can access the UAV and the RPN APIs only via the Mission Management Interface. Figure 3.13 shows the home page of the Mission Management Interface. The MMI is a web application (Backend + Frontend) and binds to the TCP port `5000`. The MMI introduces operator management, mission management and visualisation tools. The MMI can be broken down into a nexus of modules :

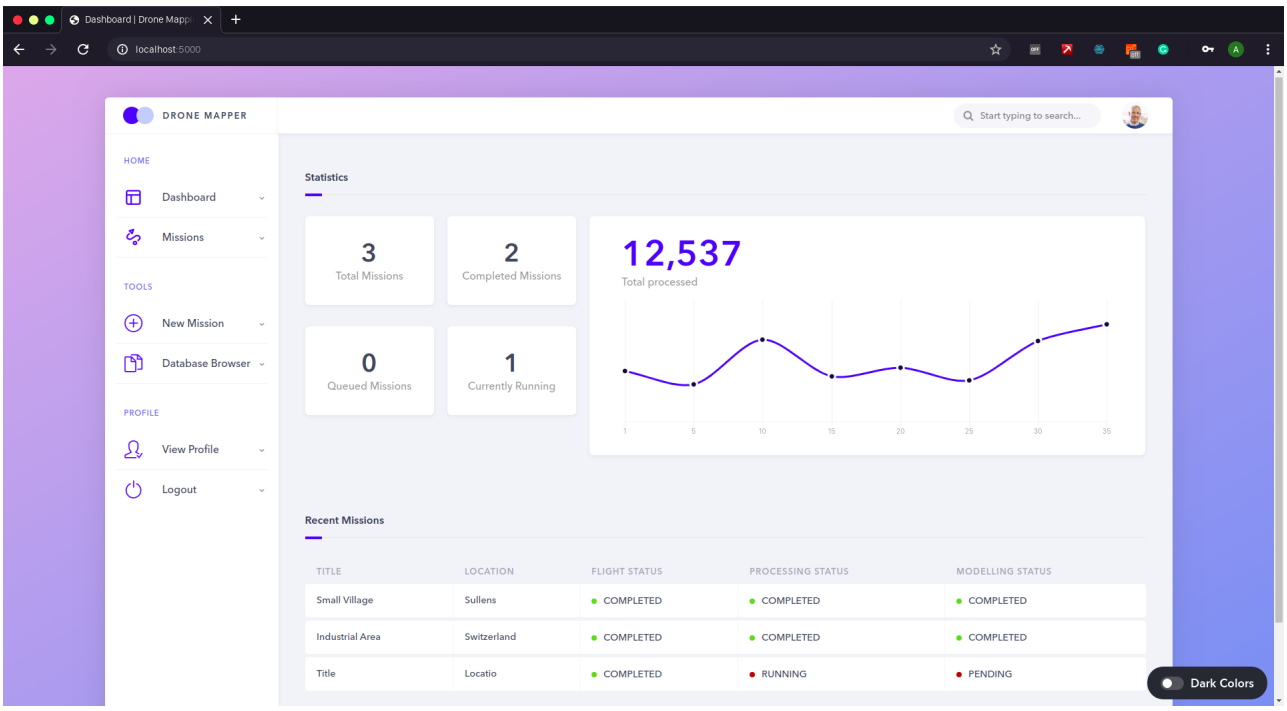

Fig. 3.13.: Screenshot: Mission Management Interface (Homepage)

1. Mission Management Interface

   The mission management interface provides GUI based forms for configuring full / partial missions. Full missions require flying where as partial missions are those in which the image acquisition has been done beforehand.

   Figure 3.14 depicts the home page of the new mission section. The new mission section offers mapping for three main use-cases : (1) Reconnaissance, (2) Architecture, (3) Agriculture. The type of mission is stored with the mission details and is helpful for creating useful statistics and / or segregating the use-cases of the the project. The user can choose a (1) full or a (2) partial mission in any of the types.

   a) Full Mission

      A full mission requires a flight plan and makes use of available drones to fly through the flight plan and capture images. During the setup of a full mission, the operator has to create a polygon around the area to be mapped in the flight planner. Additional details such as Title, Location (for storage) and

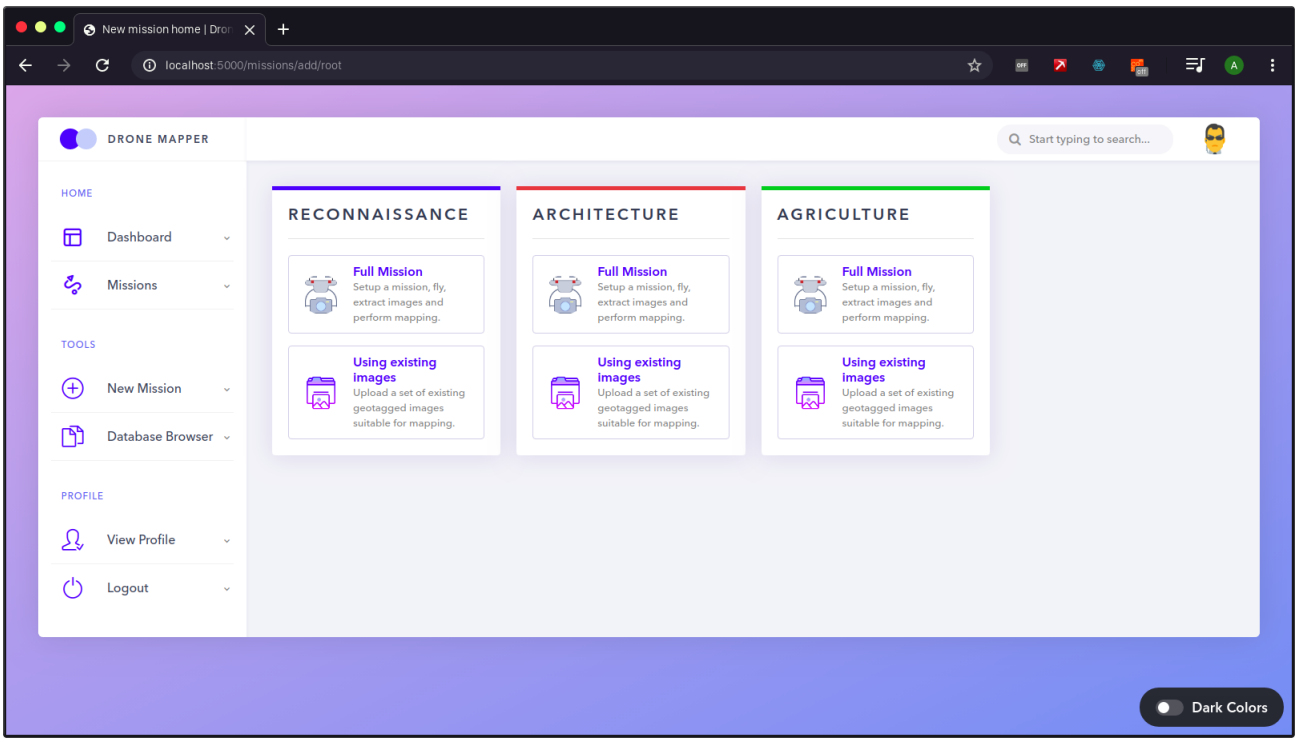

Fig. 3.14.: Screenshot : Create a Mission (Home)

Description are taken in from the operator. In the final step, the operator has to select which drone is to be used for the current mission. The list of drones is populated using the Drone Discovery Protocol as discussed earlier in the UAV software implementation section.

Figure 3.15 shows the flight planner user interface during the setup of a new mission. This is followed by the extra details section where the operator has to enter the mission title, mission location and mission description, as shown in figure 3.16. Finally, the operator has to select the drone in the hardware selection menu as shown in figure 3.17.

2. Flight Planner

   For a full mission, a proper flight plan has to be computed. The operator configures the home position, and draws a polygon around the area to be mapped as shown in figure 3.15. However, simply following the border of the area to be mapped

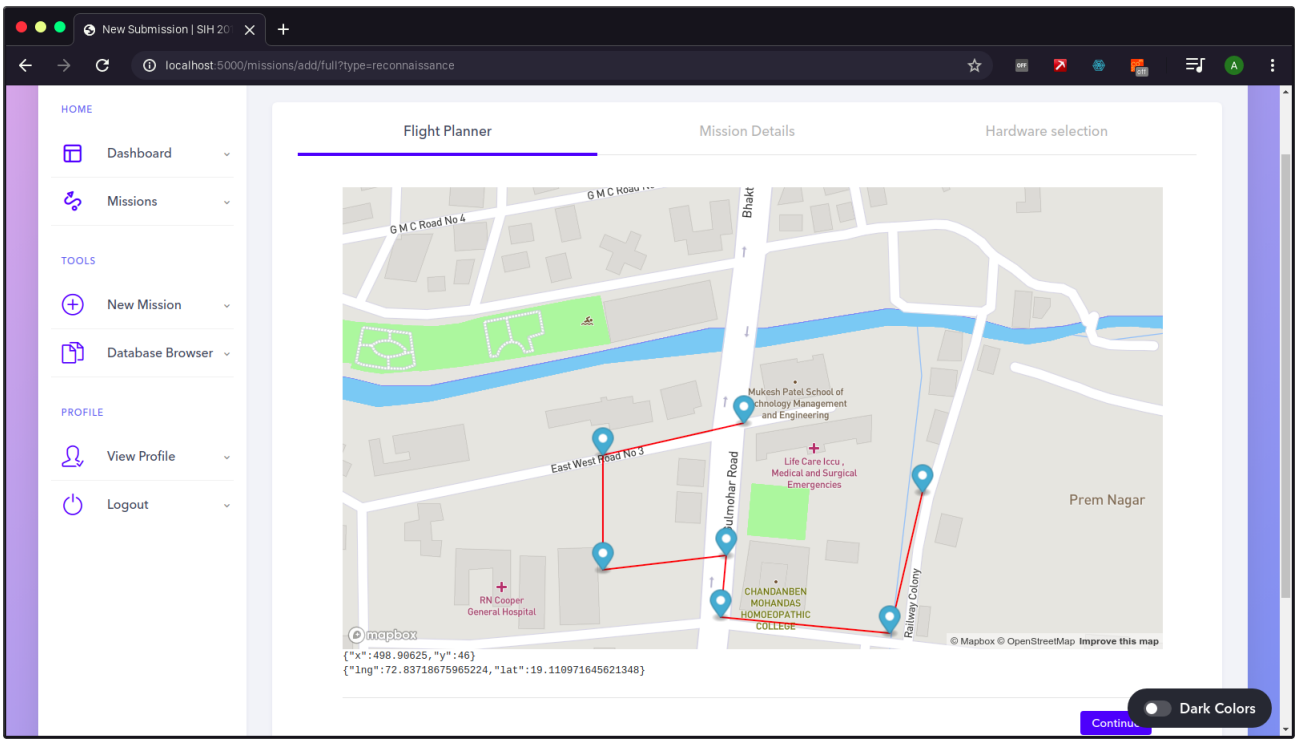

Fig. 3.15.: Screenshot : Setup a full mission (Flight Planner)

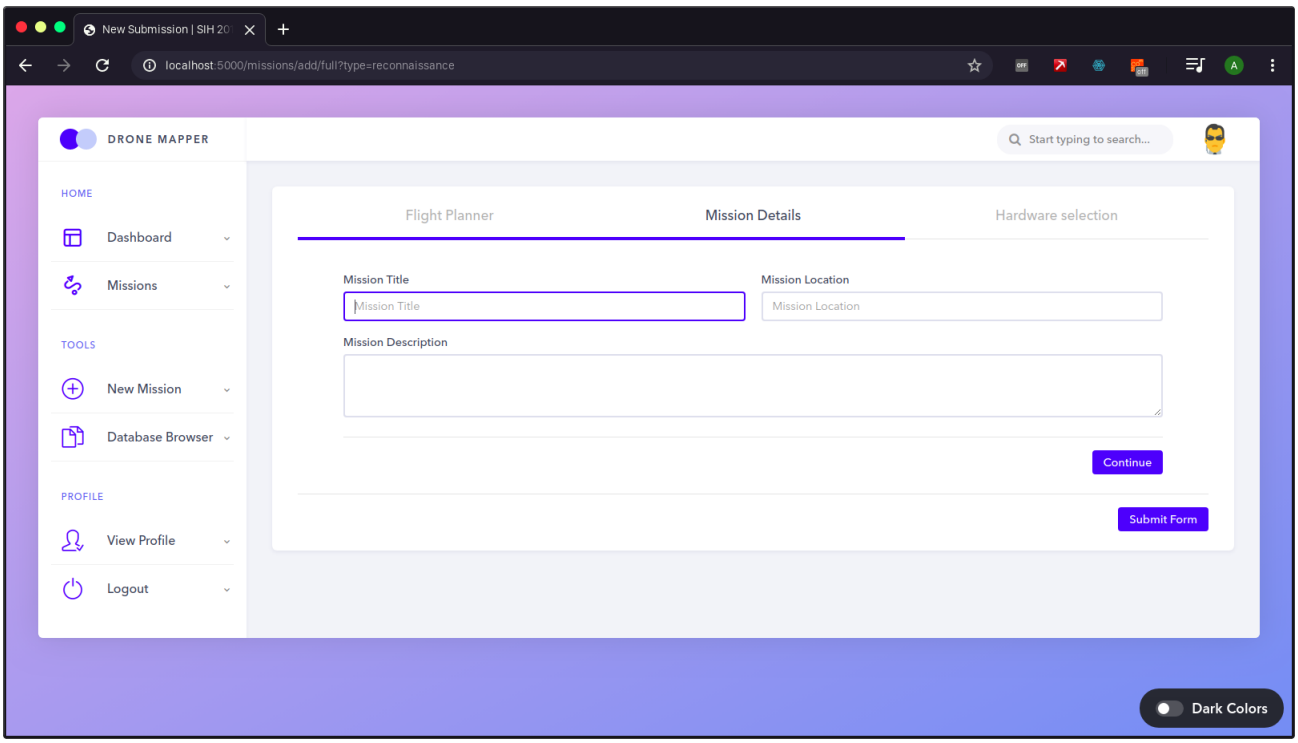

Fig. 3.16.: Screenshot : Setup a full mission (Extra details)

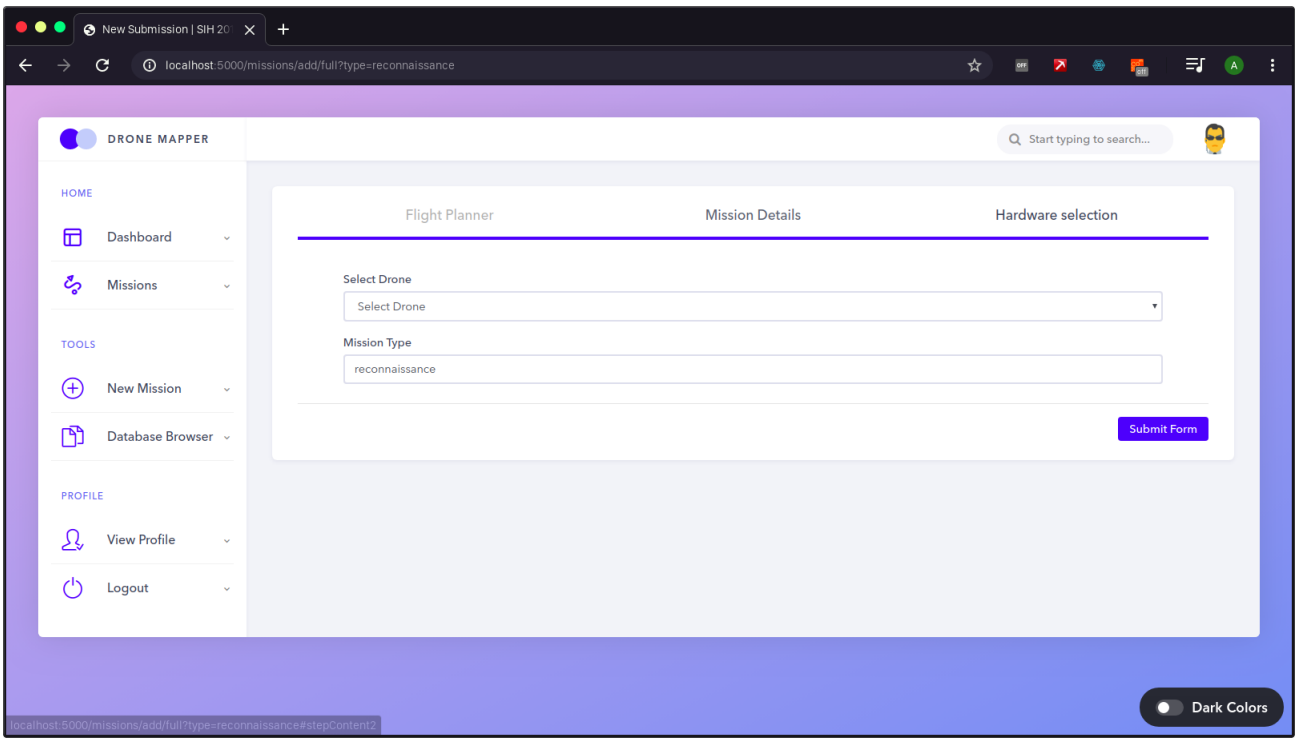

Fig. 3.17.: Screenshot : Setup a full mission (Hardware Selection)

will not provide enough image data for an accurate reconstruction. An optimum flight route has to be computed before the flight can be started. To compute an optimum flight plan, the camera parameters play an important role. The algorithm to compute an optimum path is a stage process.

a) Computing the Ground Sampling Distance (GSD) : The ground sample distance is the distance between center points of each sample taken of the ground. Since we're capturing digital photos, each sample is a pixel. In simpler terms, the GSD tells us how big each pixel is on the ground. Figure 3.18 depicts GSD visually. To compute the GSD for a particular mission, we need to know the (1) Camera sensor resolution (I_w, I_h), (2), Camera Sensor dimensions (S_w, S_h), (3), Camera Focal Length (F_l) & (4) Flight height (F_h).

Consider a setup as shown in figure 3.19. If the camera dimensions are not symmetric (in case of rectangular sensors), two Ground Sampling Distances

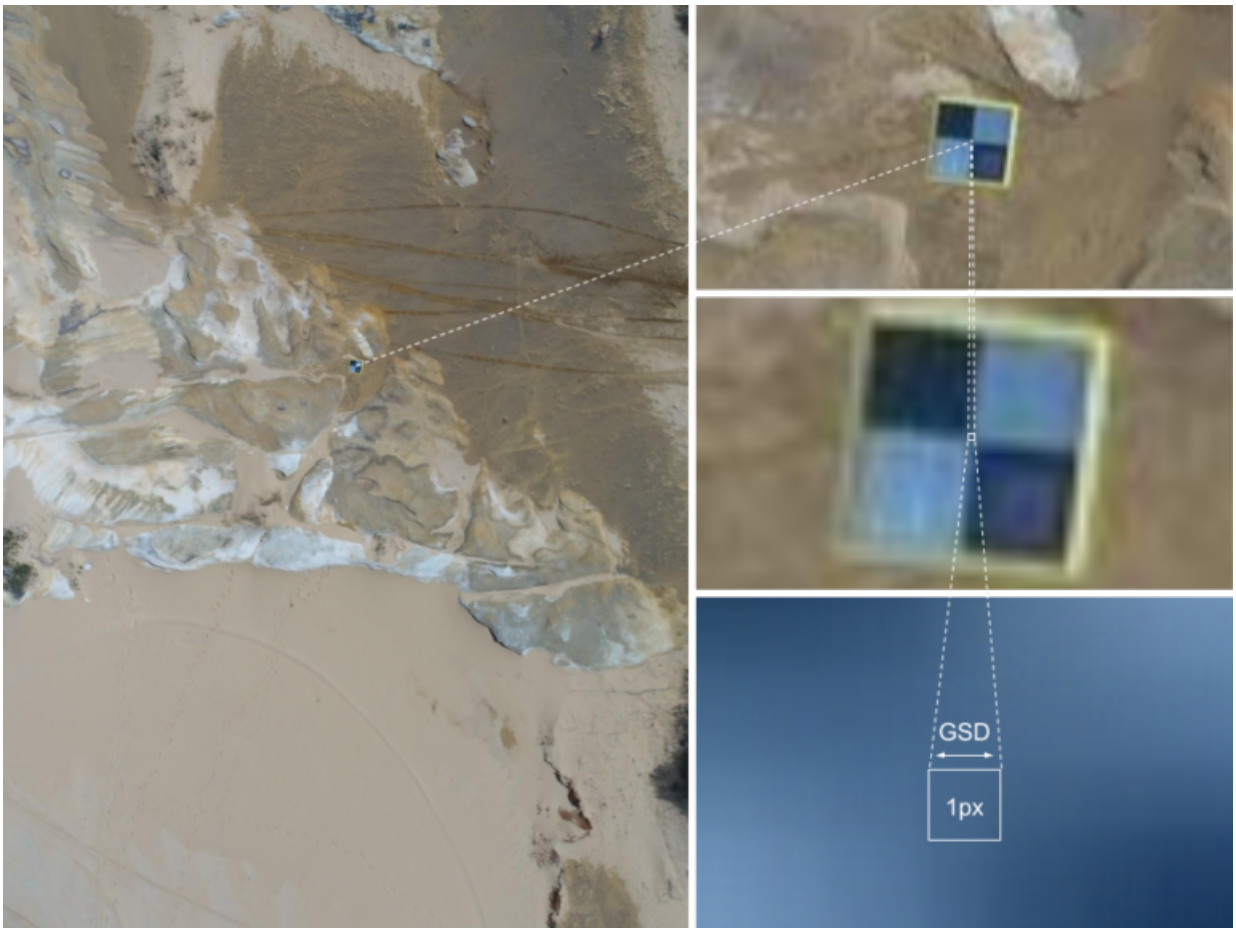

Fig. 3.18.: Understanding the Ground Sampling Distance (GSD)

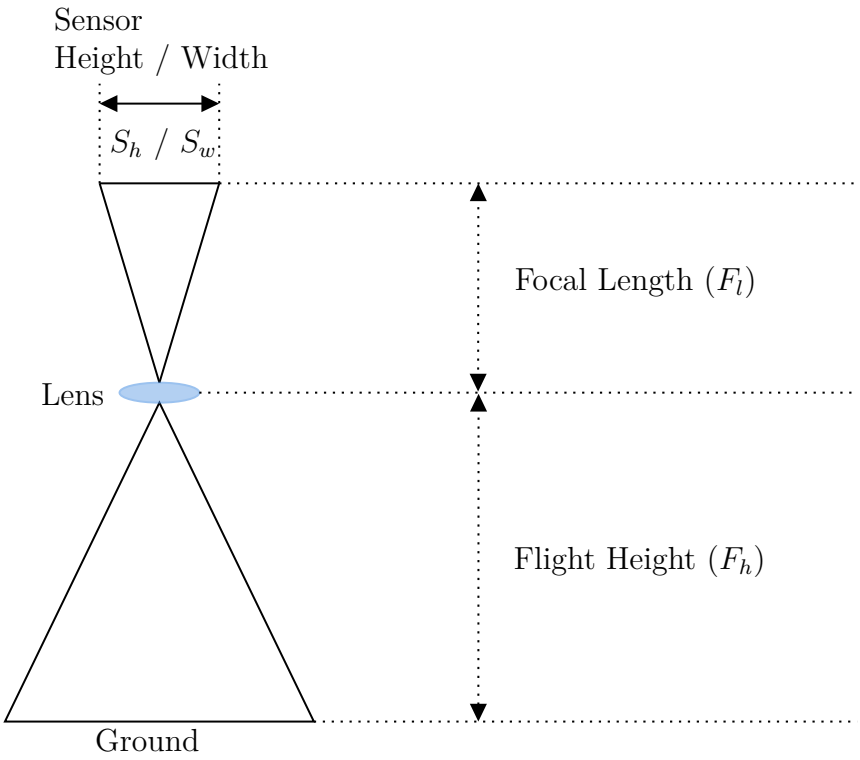

Fig. 3.19.: Setup for calculating the Ground Sampling Distance

can be computed. The worst-case GSD (higher value) is considered for all the further calculations. This creates scope for an error margin. GSD_h can be computed as shown in equation 3.2 and GSD_w can be computed as shown in equation 3.3.

$$GSD_h = \frac{F_h * S_h}{F_l * I_h} \tag{3.2}$$

$$GSD_w = \frac{F_w * S_w}{F_l * I_w} \tag{3.3}$$

Once the GSD is known, we can compute how much actual area is covered by a drone in one particular image. Now we can apply the overlap criterion to generate a route over the map. Figure 3.20 provides a visual representation of the forward and side overlap. Once the GSD and the overlap is known,

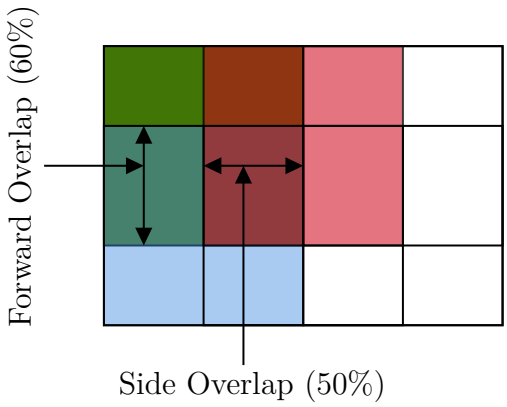

Fig. 3.20.: Understanding Forward Overlap and Side overlap

we can compute the horizontal and vertical distance covered by one image. Assuming that the drone is in the center of the area covered, the forward overlap criterion can be applied to compute the camera trigger distance, and the side overlap criterion can be used to compute the horizontal offset in next vertical lap. Figure 3.21 shows a route generated by this algorithm.

3. Mission Browser

   Mission browser provides an interface to view all missions and their details. On the missions home page, the missions are sorted based on their status. The status of any mission can be "QUEUED", "FLYING", "RUNNING", "CANCELLED, "FAILED", or "COMPLETED".

   Figure 3.22 shows a screenshot of the mission home, where in two missions are in the COMPLETED section and one of the mission is under the RUNNING section.

   Figure 3.23 shows the screenshot of a single mission home. The processing logs, along with a progress bar are placed in the main area along with options to view the 3D model, view the AR model and view the Orthophoto, DSM, DTM on a map interface. On the right hand side, the operator who started the mission can be seen, and a list of faces that were clustered during the mission is shown (if any).

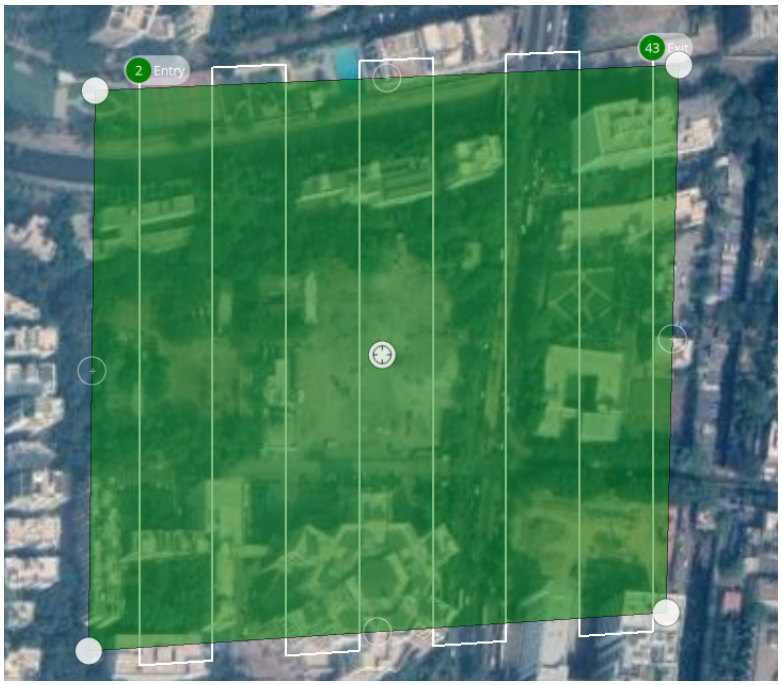

Fig. 3.21.: Route generated by the flight planning algorithm

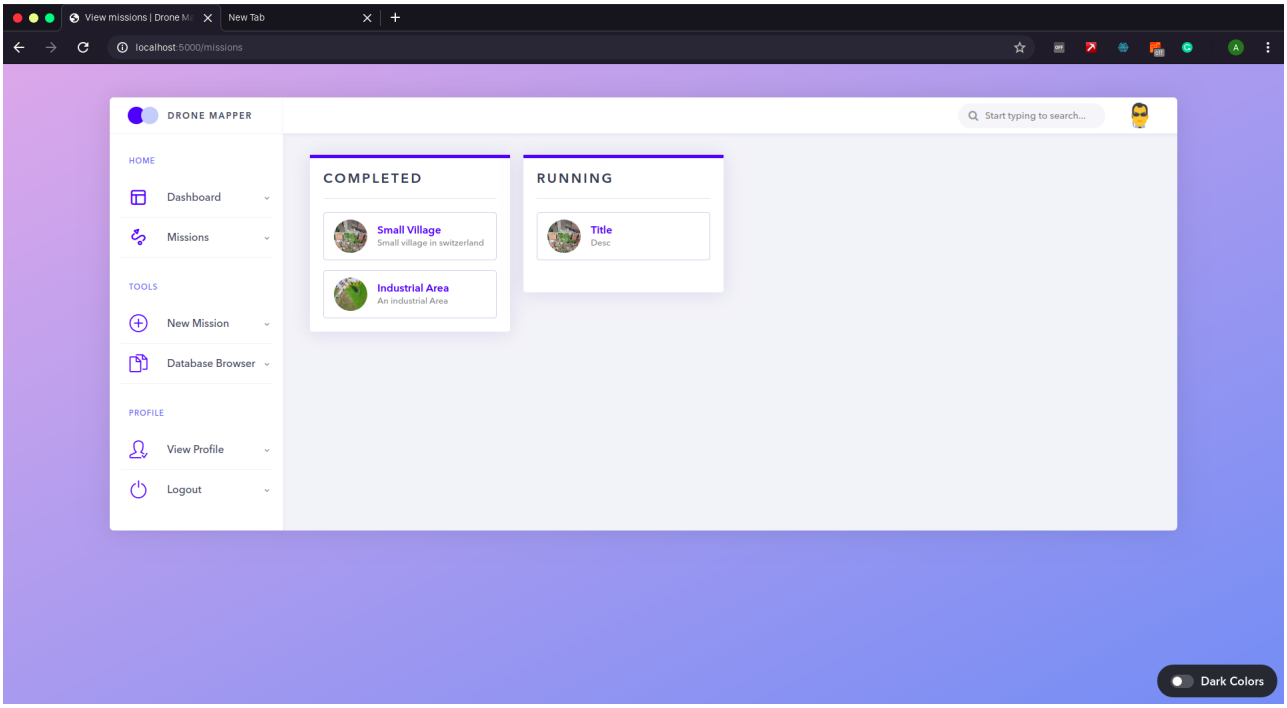

Fig. 3.22.: Screenshot : Mission Browser Home

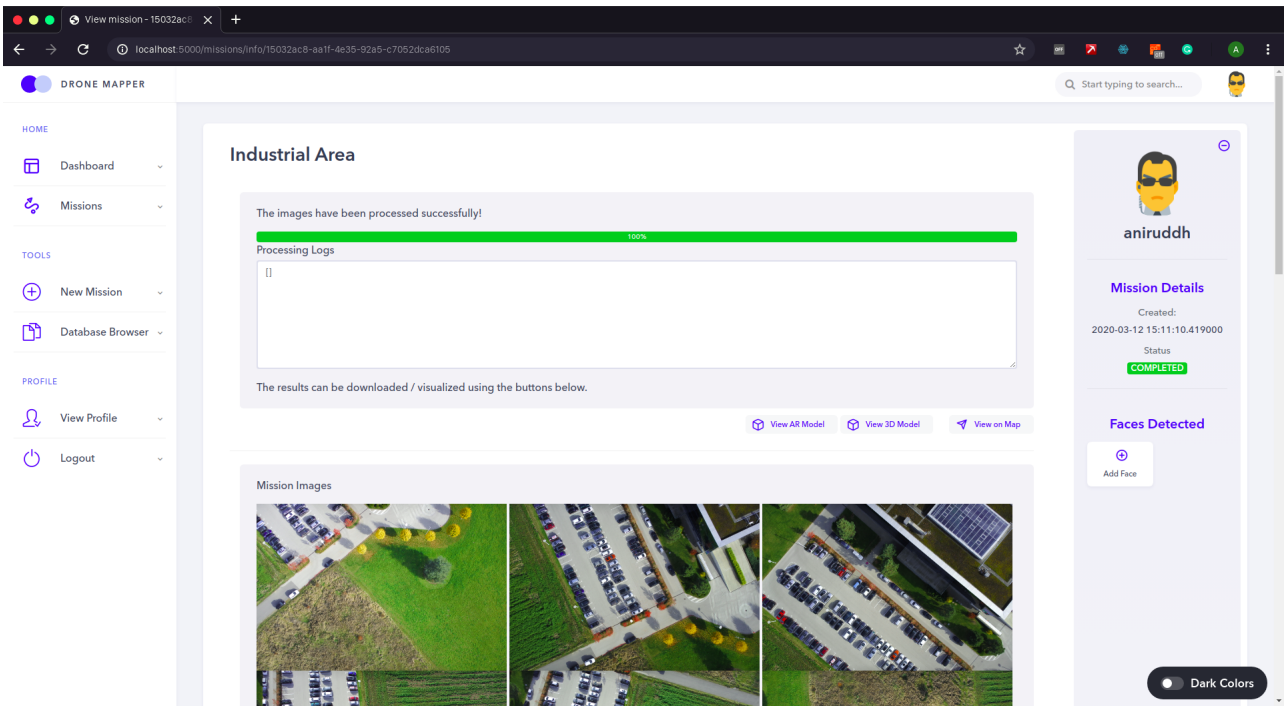

Fig. 3.23.: Screenshot : Single Mission Home

Below the output console, a photo gallery has been implemented which shows all the images that were captured during the mission.

If a mission is under the "FLYING" status, the single mission homepage shows a real-time view of all the flight instruments. They include a (1) ground speed indicator, (2) Pitch and Roll axis indicator, (3) Analog alt-meter adjusted to 10m for one full rotation, & (4) Compass Heading. Below that is a map updated with real-time location information about the drone. The figure 3.24 shows the interface in action.

4. Orthophoto, DEM visualiser A leaflet container with base maptiles loaded using MapBox is used for displaying the Orthophoto, DSM and DTM. The Orthophoto, DTM and DSM are stored as a tile map resource and are loaded using a REST API URI string. The map also offers options for to-scale measurements using the measure tool on the map.

   Figure 3.25 shows the orthophoto viewed on a map in the leaflet container.

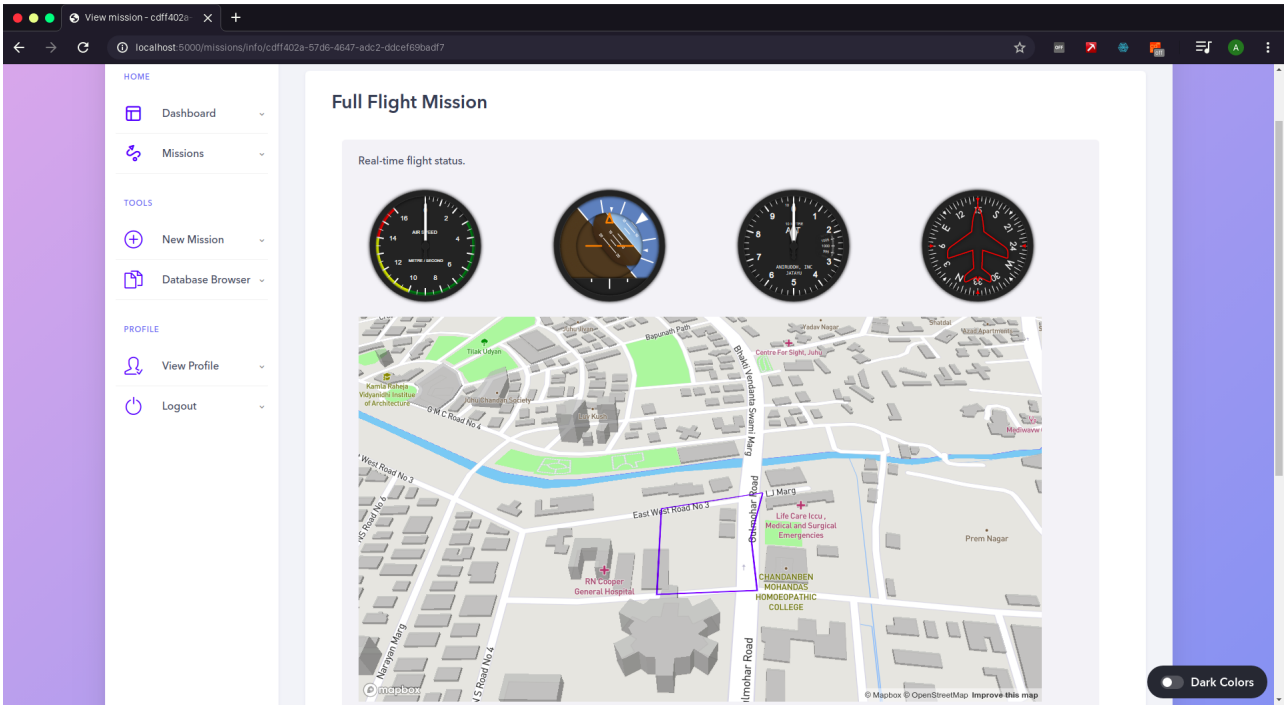

Fig. 3.24.: Screenshot : Mission Home - Flying status

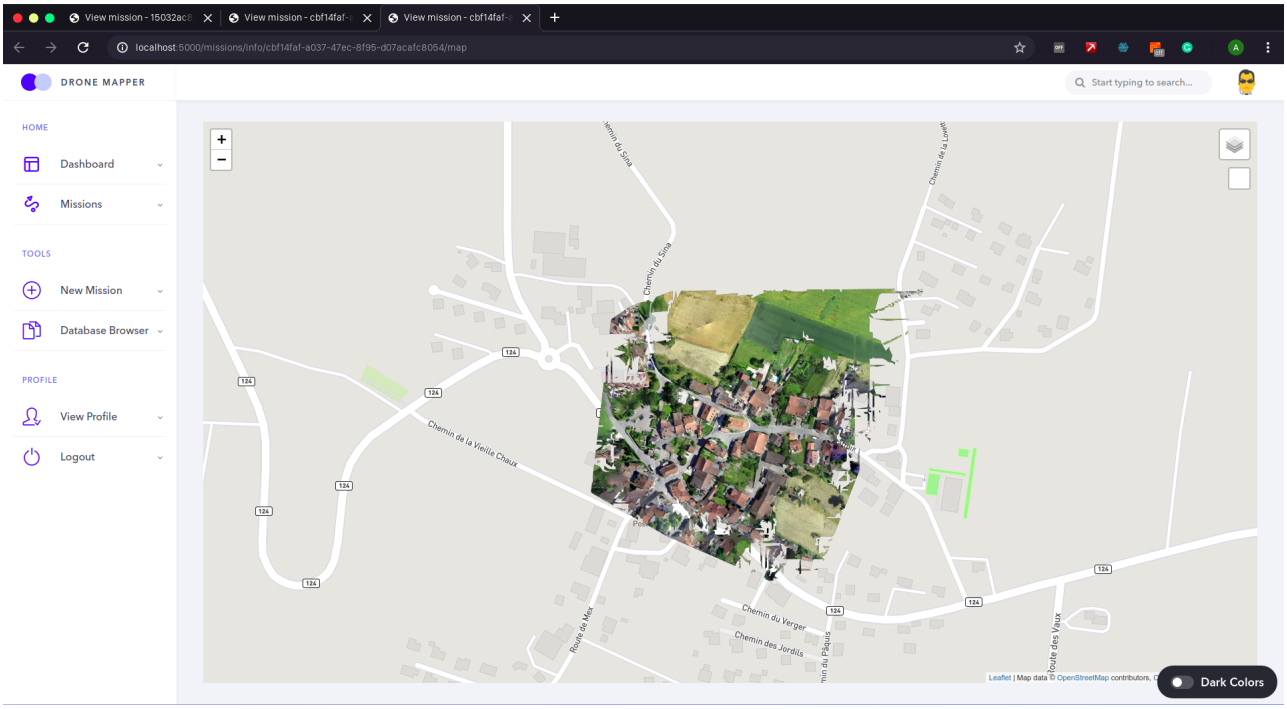

Fig. 3.25.: Screenshot : Orthophoto Viewer

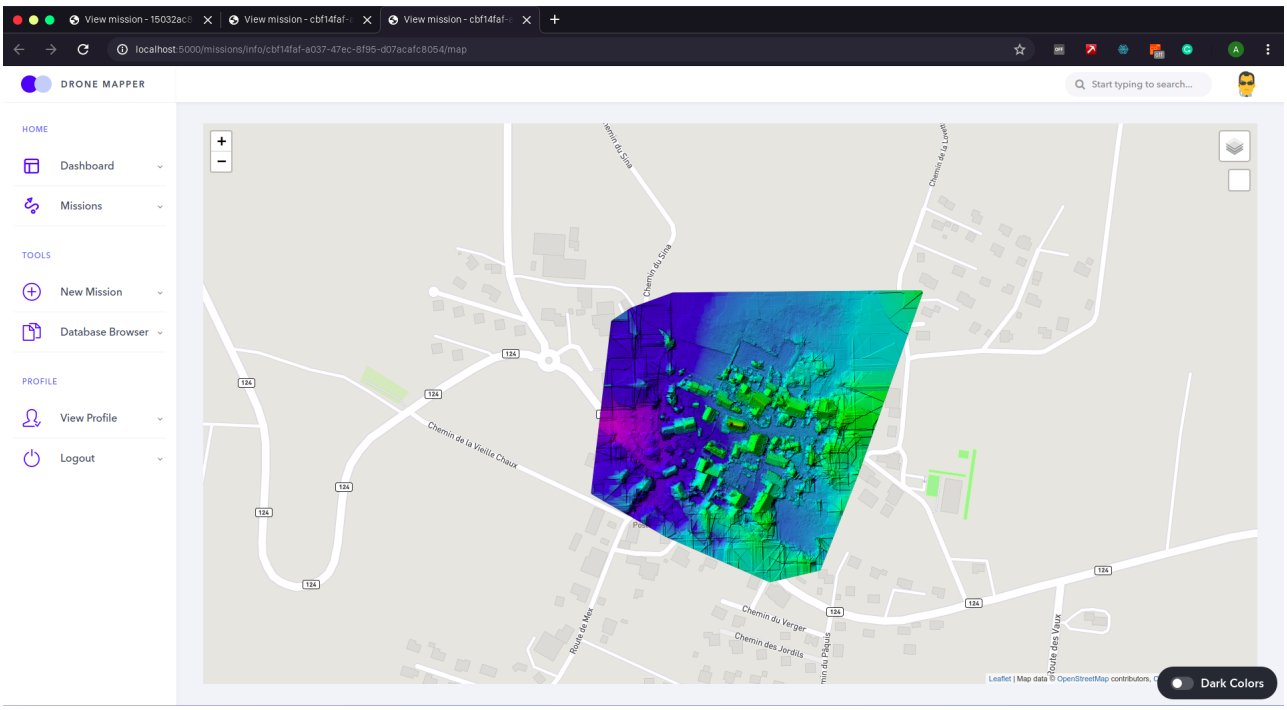

Fig. 3.26.: Screenshot : Digital Surface Model viewer

Figure 3.26 shows the DSM viewed on a map in the leaflet container.

Figure 3.27 shows the DTM viewed on a map in the leaflet container.

Figure 3.28 shows the measurement tool in action on an orthophoto shown on a map container.

5. 3D model visualiser The `model-viewer` JavaScript library is used to render a large textured model right into the browser. The field-of-view and the camera controls are completely adjustable. Figure 3.29 shows the 3D model being rendered in the browser.

6. AR Playground The `model-viewer` JavaScript library is used to render a large textured model right into the browser. The AR model does not show colors at the moment due to large texture sizes and this is a feature on high priority in the future scope.

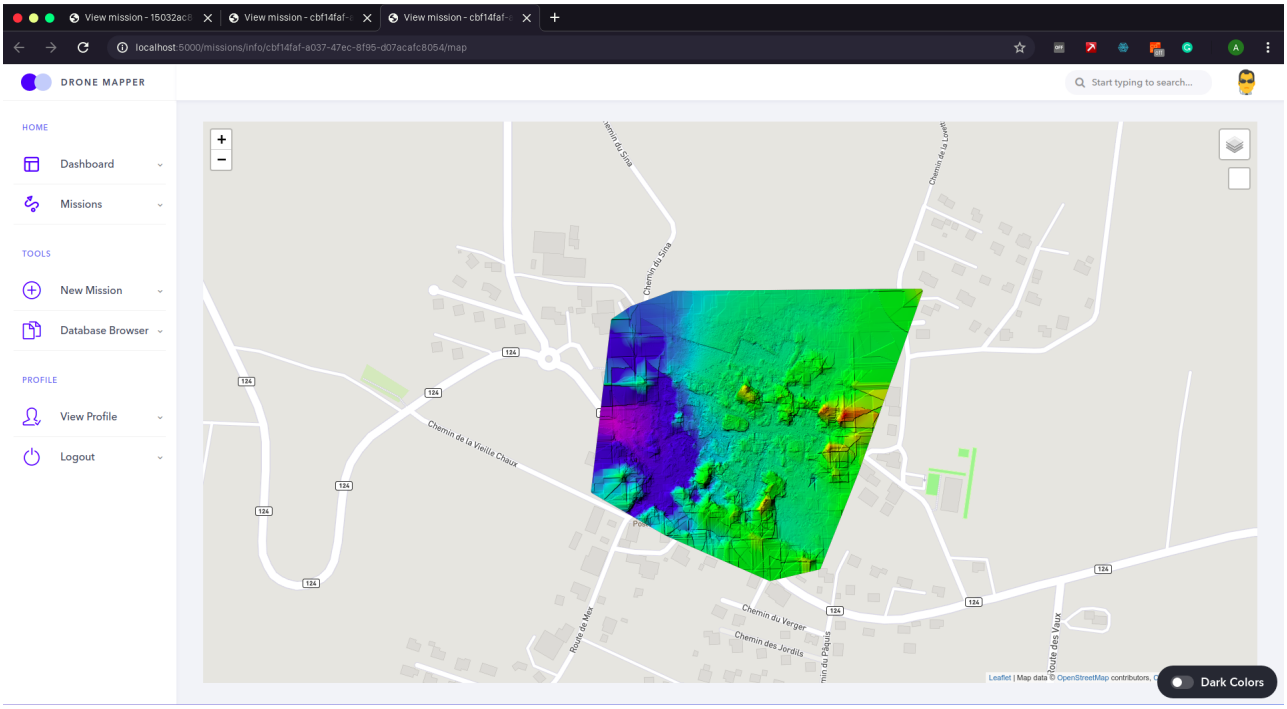

Fig. 3.27.: Screenshot : Digital Terrain Model viewer

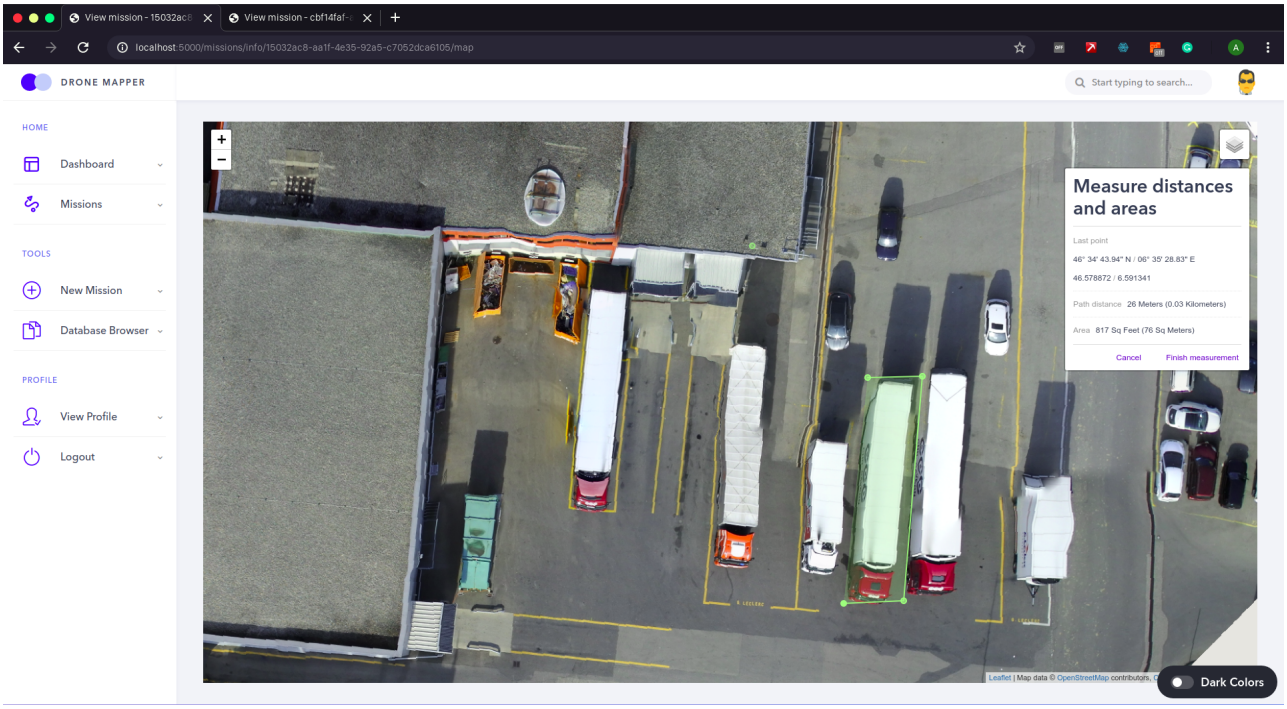

Fig. 3.28.: Screenshot : Measurement tool on map

Fig. 3.29.: Screenshot : 3D model viewer

7. Face and weapons database Each image is passed through the face clustering node and the weapon recognition node. The result of each image in association with the mission is stored into a database. The database scheme for storing the face and weapons results has been depicted in the figure 3.30. The `uuid` acts as the primary key for the `missions` collection and the `mission_uuid` acts as the secondary key for the `weapons` and `faces` collections. Every time an image is passed through the system, a new document corresponding to the results are stored into the corresponding collections. These are retrieved and displayed in the mission browser.

8. User Management (Supports multiple operators)

   The mission management interface allows multiple users with their own set of drones to make use of the processing nodes. The image 3.31 shows the login page of the web-app.

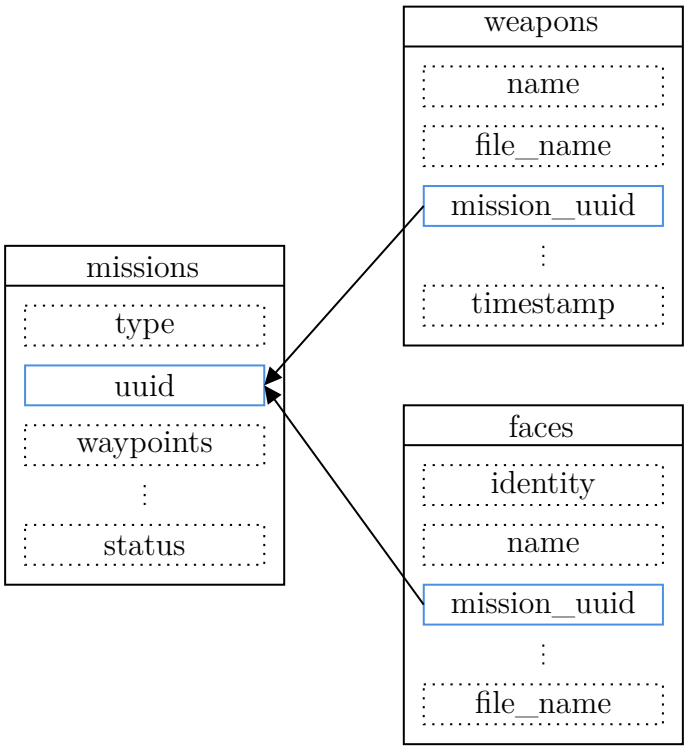

Fig. 3.30.: Face and weapons database

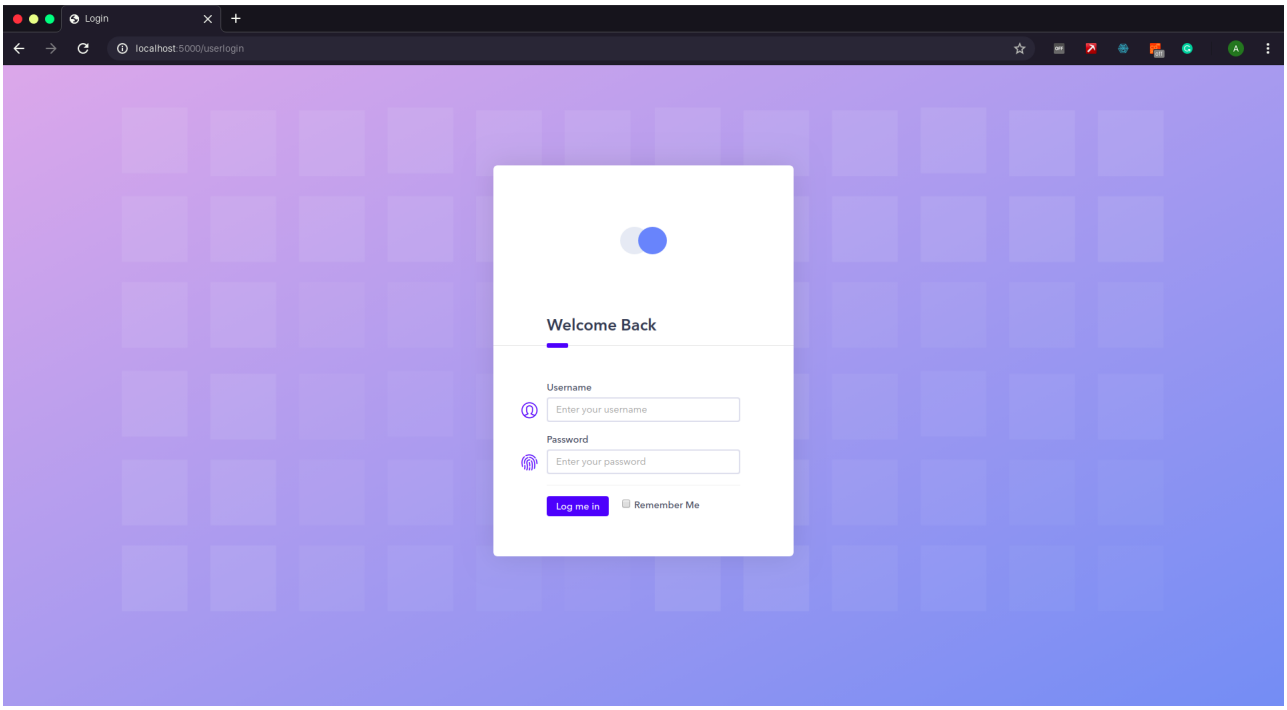

Fig. 3.31.: Screenshot : Login page

4 Results & Analysis

This chapter shows the results of our implementation by demonstrating the features of application. Additionally, we present some bench marking results and performance numbers that can be used to prove the usability of our system.

4.1 System

Even though the system architecture allows for a remote processing node, the same workstation was used for the Mission Management Interface as well as the Remote Processing Node. The development workstation was a system composed of an Intel i3-6200U processor and 8GiB RAM. The workstation was running Ubuntu 18.04 LTS (x86_64). The workstation was also used to access the mission management interface for setting up missions during the implementation work.

4.2 Performance Tests

As discussed in the Chapter 3, the system is able to produce accurate orthophotos, digital surface models, digital terrain models, 3d meshes, face clusters and weapons list. The results for each of the product have been discussed in separate sub-sections.

81

4.2.1 3D Model, Orthophoto, DEMS

The aerial processing node, responsible for generating the Orthophoto, 3D mesh, Digital Surface Model and Digital Terrain Model was tested on two datasets – (1) A small city in Switzerland (Sullens) & (2) A large industrial area in Switzerland (SenseFly office and surroundings). Both the aerial datasets have been made available by SenseFly drone group for educational research and testing [@35]. The following sections discuss the result for each dataset.

1. **Results on Dataset - Sullens (A small city in Switzerland)**
   The sullens aerial dataset consists of 37 high resolution geo-tagged images captured from a flight height of 40m. Table 4.1 lists down general parameters related to the dataset.

Tab. 4.1.: Sullens dataset - General Parameters

GSD	Coverage	Flight height	No. of Views	GCP Available
3.4cm/pixel	0.03 sq.km	40m (131.2ft)	37	no

Table 4.2 shows the results obtained on processing the sullens dataset.

Tab. 4.2.: Sullen Dataset: Processing Results

No. of views	Avg. GSD	Total featues	Median keypoints
37 of 37 (100%)	5.4cm/px	37,414	1,011 / image

Time taken	Area Covered	No. of vertices	Tile Resolution
1hr 23mins	0.032 sq.km	5,761,385	4,096

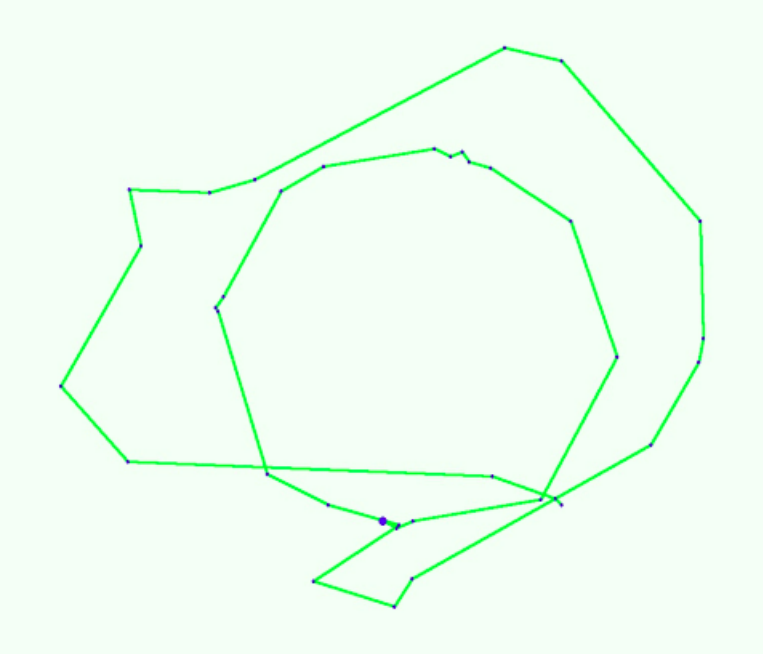

Fig. 4.1.: Sullens Dataset: GPS waypoints

Fig. 4.2.: Sullens Dataset: Orhophoto mosaic

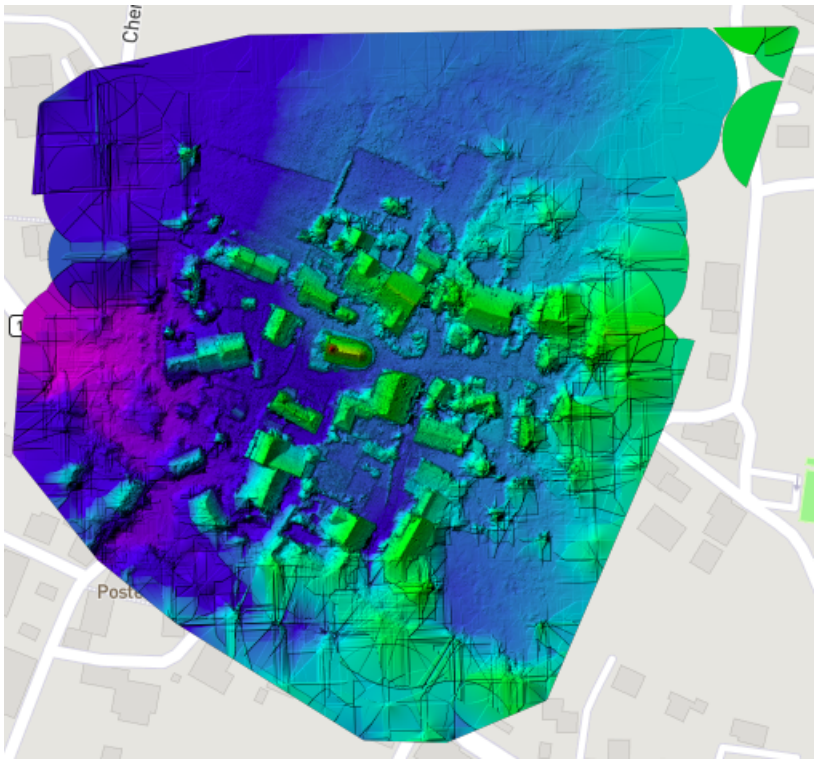

Fig. 4.3.: Sullens Dataset: DSM

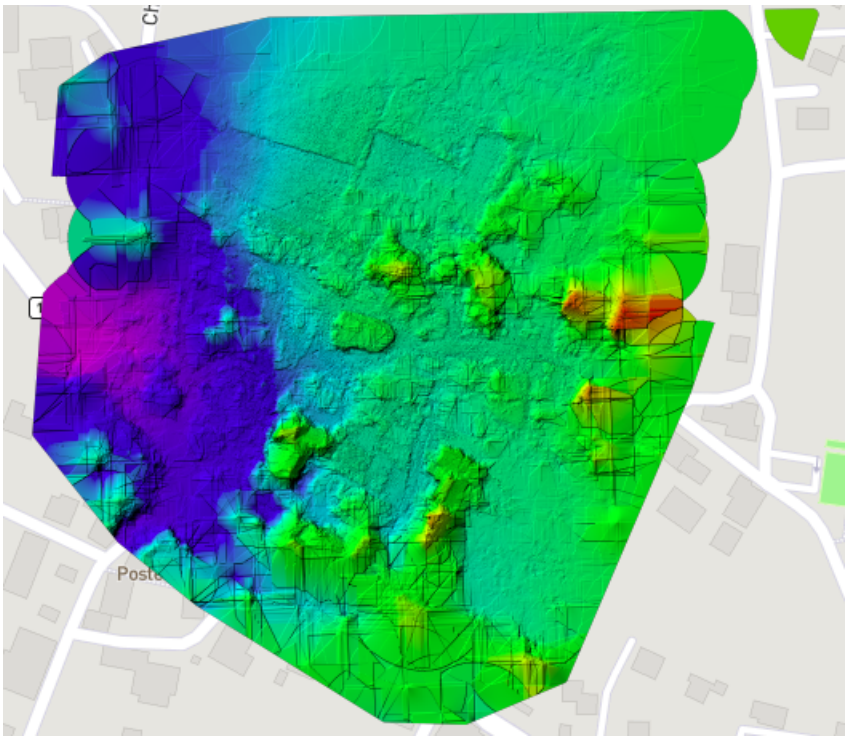

Fig. 4.4.: Sullens Dataset: DTM

Figure 4.1 shows the topview of the geotags extracted from the EXIF tags of the RGB images. The green line follows the geotags of the images in the time starting from the large blue dot. The figure 4.2 shows the Orthophoto from sullens on a map. Figure 4.3 shows the Digital Surface Model from Sullens on a Map. The Figure 4.4 shows the Digital Terrain model from Sullens on a map.

2. **Results on Dataset - Industrial Area (Industrial area in Switzerland)**

The industrial area aerial dataset consists of 113 high resolution geo-tagged images captured from a flight height of 100m. Table 4.3 lists down general parameters related to the dataset.

Tab. 4.3.: Industrial area dataset - General Parameters

GSD	Coverage	Flight height	No. of Views	GCP Available
2.1/pixel	0.21 sq.km	100m (131.2ft)	113	yes

Table 4.4 shows the results obtained on processing the Industrial area dataset.

Tab. 4.4.: Industrial area Dataset: Processing Results

No. of views	Avg. GSD	Total featues	Median keypoints
113 of 113 (100%)	2.1cm/px	140,534	1,011 / image

Time taken	Area Covered	No. of vertices	Tile Resolution
1hr 23mins	0.21 sq.km	7,103,843	4,096

Figure 4.5 shows the topview of the geotags extracted from the EXIF tags of the RGB images. The green line follows the geotags of the images in the time starting from the large blue dot. The figure 4.6 shows the Orthophoto from Industry area

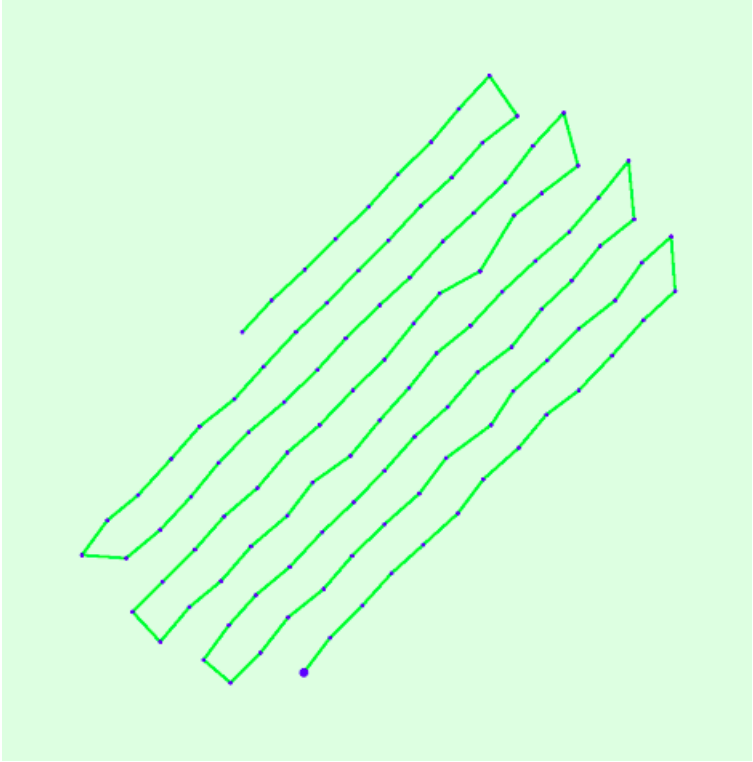

Fig. 4.5.: Industrial area Dataset: GPS waypoints

Fig. 4.6.: Industrial area Dataset: Orhophoto mosaic

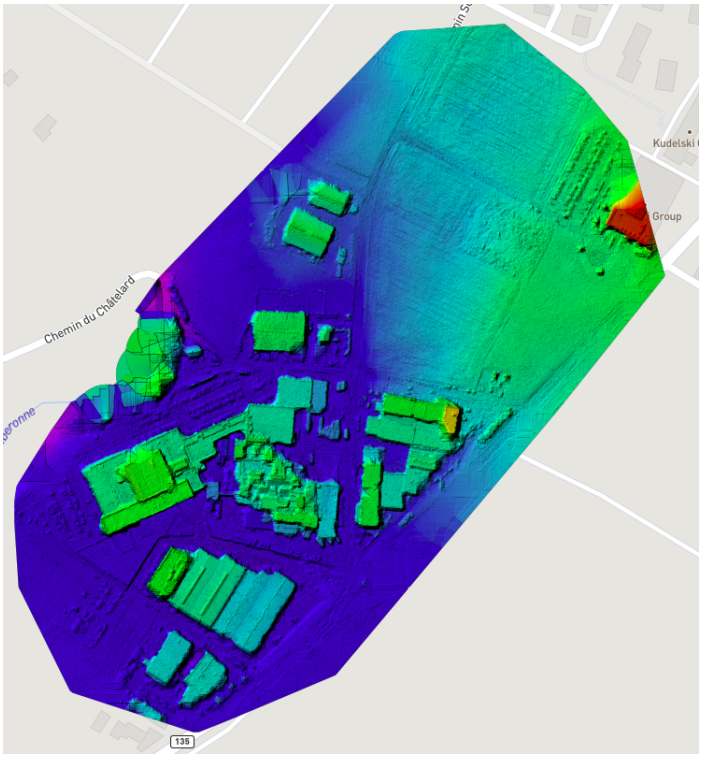

Fig. 4.7.: Industrial area Dataset: DSM

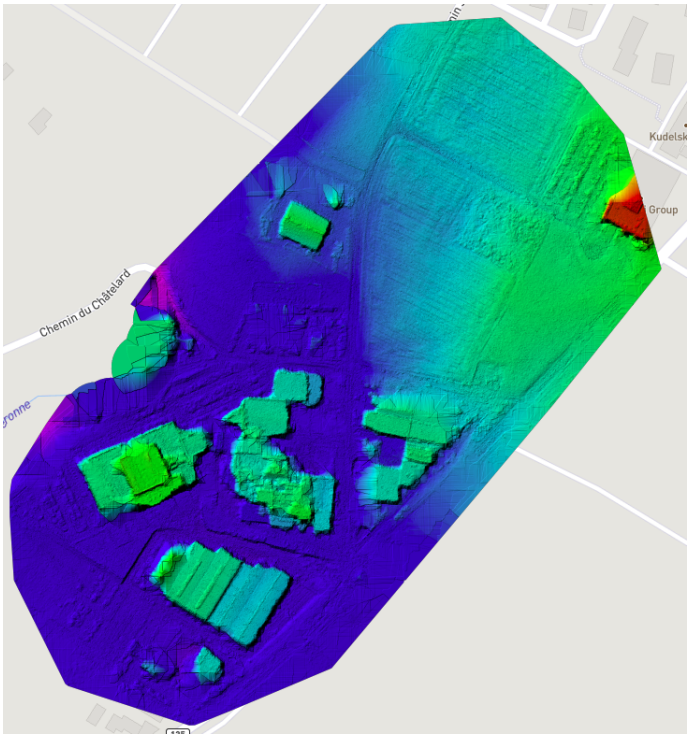

Fig. 4.8.: Industrial area Dataset: DTM

on a map. Figure 4.7 shows the Digital Surface Model from Industry area on a Map. The Figure 4.8 shows the Digital Terrain model from Industry area on a map.

3. **Overall Results**

Combining results from both the datasets, an overall result table can be computed as shown in table 4.5.

Tab. 4.5.: Overall processing result

Dataset	Images	Time Taken	No. of Vertices	No. of faces	GSD
Sullens	37	1hr 23mins	5,761,385	4096	5.4cm/px
Industrial Area	113	3hr 17mins	7,103,843	4096	2.1cm/px

4.2.2 Face Clustering

The face clustering algorithm was tested on the LFW dataset. Details about the LFW dataset have been tabulated in the table 4.6. The table 4.7 shows the clustering and classification accuracy from our algorithm.

Tab. 4.6.: LFW Dataset information

Numer of Images	Number of unique identities	Identities with 2 or more distinct images
13,233	5,749	1,680

Tab. 4.7.: Verification of face recognition algorithm on LFW dataset

No. of real identities	No. of clusters created	Accuracy
5,749	5,735	99.75%

No. of images	Correctly classified	Accuracy
13,233	13,108	99.05%

4.2.3 Weapon Recognition

The weapon recognition data has been collected from the GitHub repository available at [@11]. The dataset provides 10,014 images that have been classified into three types : "Gun", "Long Gun" and "Knife". A subset of 2,000 images was manually annotated with bounding boxes covering "Gun" and "Knife".

The weapon recognition algorithm was trained on 20,000 iterations and was stopped at 0.1016 validation loss. We achieved a mAp of 0.50@0.75 IOU.

5 Advantages, Limitations & Applications

5.1 Advantages

Surveying with a drone offers enormous potential. With a drone, it is possible to carry out topographic surveys of the same quality as the highly accurate measurements collected by traditional methods, but in a fraction of the time. This substantially reduces the cost of a site survey and the workload of specialists in the field. Capturing topographic data with a drone is up to five times faster than with land-based methods and requires less manpower. An aerial mapping drone can take off and fly almost anywhere. One is no longer limited by unreachable areas, unsafe steep slopes or harsh terrain unsuitable for traditional measuring tools. There is no need to close down highways or train tracks. In fact, data can be captured during operation without an organizational overhead. Digital Terrain Models (DTM) and Digital Surface Models (DSM) are useful in 3D modeling for telecommunications, urban planning and aviation. In case of defence purposes, this aerial surveillance and reconstruction can be done for a base area without any need for any personnel to actually visit the site. With AR export, personnel can observe the layout in a more detailed manner and plan their execution. The same technology can also be used to strengthen existing army bases by creating virtual maps and analysing them. Thus the loss of human life is prevented by finding a safer and faster alternative to gather intelligence about the enemy lines.

90

5.2 Limitations

1. The 3-D reconstruction as well as the development of the digital elevation models requires high processing power, and do not make use of a GPU for the reconstruction or recognition processes even if it is made available.

2. For extremely accurate models a high resolution camera is needed that can provide the required clear images. Currently, a $25 PiCamera is used for capturing images.

3. Ranging hardware can be added onto the drone to enable stable altitude hold. Currently, the drone fluctuates in altitude by about 0.5-0.6m while flying.

4. The maximum flight time of the drone is 20 minutes on the current battery. This can be increased by adding a high capacity battery.

5.3 Applications

1. **Reconnaissance**

   Reconnaissance plays a key role in strategic planning and coordination, however attack sites are often dangerous for humans to travel to or inaccessible because of the terrain. Thus by sending an UAV loss of life can be prevented and reconnaissance can be done in a faster and more accurate manner. Face detection and recognition will help to identify the strength of enemy forces and to highlight any key targets present. Weapon Recognition pipeline will help to gauge the firepower of the enemy. By looking at the reconstructed map in AR the soldiers will know what to expect on the field and prepare accordingly.

2. **Architecture**

   Before greenlighting any building as structurally stable it is important to perform a detailed analysis of the building in order to have a proper understanding of the

layout of the buildings and any faults present,if any. It will provide a systematic pipeline for enhanced scanning which involves a step by step extraction of crucial information which can be captured using Unmanned Aerial Vehicles (UAVs) which are running a custom auto-pilot which will be specifically designed to optimise this task. It can be used for reconstructing the structure as a 3-D mesh and comparing it with the plan given for the said structure and also to try and find any structural faults present. Finally, all of this information can be compiled and exported into a virtual reality platform where one can scrutinise the model and make any desired changes on it, thus increasing the feasibility of experimentation.

3. **Agriculture**

   The Digital Surface Model (DSM) and Digital Terrain Models (DTM) are particularly useful in this domain.

   a) VEGETATION MANAGEMENT: Along the predefined route, DSMs can see where and how much vegetation is encroaching.

   b) HYDROLOGIC MODELLING: Hydrologists can use the generated DTM to delineate watersheds, calculate flow accumulation and flow direction.

   c) TERRAIN STABILITY: Areas prone to avalanches are high slope areas with sparse vegetation. This is useful when planning a highway or residential subdivision.

   d) SOIL MAPPING: DTMs assist in mapping soils which is a function of elevation (as well as geology, time and climate).

6 Conclusion & Future Scope

6.1 Conclusion

This project was successfully able to fulfill all the problem statements defined in the section 2.1. The UAV was able to complete a mission by flying autonomously on a route computed by the MMI in order to survey a geological area as defined in the Flight planner section in the MMI. The processing node successfully generated an Orthophoto mosaic, Digital Terrain Model, Digital Surface Model and a 3D mesh of the area surveyed. The face recognition and and the weapon recognition pipelines provided state-of-the art results as shown in the 4.2. Majority of the limitations posed by the system can be solved by using higher quality equipment such as as lighter drone, high capacity battery, a powerful workstation and a high resolution camera.

6.2 Future Scope

This project has a vast scope in various fields such as defence, agriculture and architecture to name a few. This work can be further extended in the following ways.

1. Making use of a GPU to decrease processing time.
2. To create a real-time pipeline for aerial mapping.
3. Increased flight time either by using batteries with higher capacity or by using lighter frames (made out of carbon fiber).

4. Ranging sensors for better flight stability.

5. To build a proper user management system where in multiple operators can sign up and process their images.

6. To build a 3D mapping gallery, where-in all the projects processed by the software can be shown on a single map. Anyone with an account should be able to publish their models onto the open map.

7. To generate interactive and textured Augmented Reality / Virtual Reality models.

Bibliography

[1]M. Adorjan, “Ein kollaboratives structure-from-motion system”, (cit. on pp. 25, 27, 29).

[2]A. Allouch, O. Cheikhrouhou, A. Koubaa, M. Khalgui, and T. Abbes, *Mavsec: Securing the mavlink protocol for ardupilot/px4 unmanned aerial systems*, May 2019 (cit. on p. 13).

[3]B. Amos, B. Ludwiczuk, M. Satyanarayanan, *et al.*, “Openface: A general-purpose face recognition library with mobile applications”, *CMU School of Computer Science*, vol. 6, 2016 (cit. on pp. 32, 33).

[4]M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features”, *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007 (cit. on p. 28).

[5]Z. Chen, B. Devereux, B. Gao, and G. Amable, “Upward-fusion urban dtm generating method using airborne lidar data”, *ISPRS journal of photogrammetry and remote sensing*, vol. 72, pp. 121–130, 2012 (cit. on p. 30).

[6]S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust reconstruction of indoor scenes”, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015 (cit. on p. 23).

[7]J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse depth parametrization for monocular slam”, *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, Oct. 2008 (cit. on pp. 18, 19).

[9]E. S. M. Ebeid, M. Skriver, and J. Jin, “A survey on open-source flight control platforms of unmanned aerial vehicle”, Aug. 2017 (cit. on pp. 11, 12).

[10]D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones", *Nature*, vol. 521, pp. 460–466, 2015 (cit. on pp. 9, 10).

[12]R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, ISBN: 0521540518, 2004 (cit. on p. 27).

[13]*Hyptertext Transfer Protocol – HTTP/1.1*, RFC 2068, Jan. 1997 (cit. on p. 16).

[14]*Hyptertext Transfer Protocol – HTTP/1.1*, RFC 2616, Jun. 1999 (cit. on p. 16).

[15]"Unmanned aircraft systems", International Organization for Standardization, Geneva, CH, Standard, 2014 (cit. on p. 13).

[16]M. Keller, D. Lefloch, M. Lambers, *et al.*, "Real-time 3d reconstruction in dynamic scenes using point-based fusion", in *2013 International Conference on 3D Vision-3DV 2013*, IEEE, 2013, pp. 1–8 (cit. on p. 24).

[17]D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1755–1758, 2009 (cit. on p. 32).

[18]Y. Kortli, M. Jridi, A. A. Falou, and M. Atri, "Face recognition systems: A survey", *Sensors*, vol. 20, no. 2, p. 342, 2020 (cit. on pp. 30, 31).

[19]A. Koubâa, A. Allouch, M. Alajlan, *et al.*, "Micro air vehicle link (mavlink) in a nutshell: A survey", *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019 (cit. on pp. 11, 16).

[20]V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles", *The International Journal of Robotics Research*, vol. 31, pp. 1279–1291, Sep. 2012 (cit. on p. 10).

[21]M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation", *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019 (cit. on pp. 21, 22).

[22]H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles", *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 33–45, Sep. 2012 (cit. on p. 10).

[23]D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004 (cit. on p. 28).

[24]L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6235–6240 (cit. on p. 10).

[25]J. Meza, A. G. Marrugo, G. Ospina, M. Guerrero, and L. A. Romero, "A structure-from-motion pipeline for generating digital elevation models for surface-runoff analysis", *Journal of Physics: Conference Series*, vol. 1247, p. 012 039, Jun. 2019 (cit. on p. 29).

[26]R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system", *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015 (cit. on pp. 19–21).

[27]E. Paiva, J. Soto, J. Salinas, and W. Ipanaqué, "Modeling, simulation and implementation of a modified pid controller for stabilizing a quadcopter", in *2016 IEEE International Conference on Automatica (ICA-ACCA)*, Oct. 2016, pp. 1–6 (cit. on pp. 13, 15).

[28]D. N. Parmar and B. B. Mehta, "Face recognition methods & applications", *arXiv preprint arXiv:1403.0485*, 2014 (cit. on p. 30).

[30]T. J. Pingel, K. C. Clarke, and W. A. McBride, "An improved simple morphological filter for the terrain classification of airborne lidar data", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 77, pp. 21–30, 2013 (cit. on p. 30).

[31]F. Pirotti, "Open software and standards in the realm of laser scanning technology", *Open Geospatial Data, Software and Standards*, vol. 4, no. 1, p. 14, 2019 (cit. on p. 30).

[32]S. K. Pylappan, J. A. Howard, B. A. Moore, and C. Blumenberg, *Smart loading of map tiles*, US Patent 9,453,734, Sep. 2016 (cit. on p. 17).

[33]J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018 (cit. on pp. 34–36, 44).

[36]J. Tang, S. Liu, and J.-L. Gaudiot, "Embedded systems architecture for slam applications", Feb. 2017 (cit. on p. 25).

[37] *Transmission Control Protocol*, RFC 768, 1981 (cit. on p. 16).

[38] *User Datagram Protocol*, RFC 768, Aug. 1980 (cit. on pp. 15, 16).

[39]T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph", Robotics: Science and Systems, 2015 (cit. on pp. 17, 24, 25).

[45]C. Xing, J. Wang, and Y. Xu, "Overlap analysis of the images from unmanned aerial vehicles", Jun. 2010 (cit. on p. 29).

[46]Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing", *arXiv preprint arXiv:1801.09847*, 2018 (cit. on pp. 23, 24).

Webpages

[@8]D. G. of Civil Aviation. (). Civil aviation requiremnt for drones, section 3 - air transport, series x, part i., [Online]. Available: `http://dgca.nic.in/rules/car-ind.htm` (cit. on p. 13).

[@11]GitHub. (). Weapon classification using cnns, [Online]. Available: `https://github.com/ivaibhavkr/Weapon-Detection-And-Classification` (cit. on p. 89).

[@29]W. Pedia. (). Orthophoto, [Online]. Available: `https://en.wikipedia.org/wiki/Orthophoto` (cit. on p. 16).

[@34]D. Robertson and R. Cipolla. (). Structure from motion, [Online]. Available: http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf (cit. on pp. 26, 27, 29).

[@35]SenseFly. (). Sensefly educational drone image dataset, [Online]. Available: https://www.sensefly.com/education/datasets/ (cit. on p. 82).

[@40]Wiki. (). Micro air vehicle., [Online]. Available: https://en.wikipedia.org/wiki/Micro_air_vehicle (cit. on p. 8).

[@41]WikiPedia. (). Digital elevation model, [Online]. Available: https://en.wikipedia.org/wiki/Digital_elevation_model (cit. on p. 16).

[@42]——, (). Geographic information system, [Online]. Available: https://en.wikipedia.org/wiki/Geographic_information_system (cit. on p. 16).

[@43]——, (). Tile map service, [Online]. Available: https://en.wikipedia.org/wiki/Tile_Map_Service (cit. on pp. 17, 42).

[@44]Wikipedia. (). Unmanned aerial vehicle., [Online]. Available: https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle (cit. on p. 8).

A

Appendix: List of Components

1. 920 KV Brushless Multirotor Motor :

   KV(rpm/v): 920.

   Max Power: 370W.

   Max Thrust: 1200 gr ams.

   Weight: 53 grams.

   Shaft Diameter: 4mm.

2. BLDC 30A Electronic Speed Controller :

   Burst Current: 40A for 10 sec

   Constant Current: 30A max 40A< 10s

   BEC: 3A

3. Pixhawk PX4 Autopilot PIX 2.4.8 32 Bit Flight Controller

   Microprocessor:

   32-bit STM32F427 Cortex M4 core with FPU

   168 MHz/256 KB RAM/2 MB Flash

   32-bit STM32F103 failsafe co-processor

   Sensors:

   ST Micro L3GD20 3-axis 16-bit gyroscope

   ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer Invensense MPU 6000 3-axis

   accelerometer/gyroscope

MEAS MS5611 barometer

4. APM Power Module
   Operating Voltage (VDC): 6 to 28
   Max Input Voltage (VDC): 28
   Max Current Sensing(A) : 90

5. Flysky FS-i6 6Ch 2.4GHz Transmitter
   RF Range: 2.408-2.475GHz
   RF Power: <20dbm
   RF Channel: 135
   Power: 6V DC

6. Flysky FS-iA6 Receiver
   RF Range: 2.408-2.475GHz
   RF Channel: 135
   RF Receiver Sensitivity: -105dbm
   Power: 4-6.5V DC

7. NEO 7M GPS with Compass
   Tracking Sensitivity: 3.5-5.5V DC
   Tracking Sensitivity: 161 dbm
   Capture Sensitivity: 148 dbm
   Maximum Altitude: 18000
   Maximum Speed: 515 m/s

8. 433MHz Telemetry Module Pair

   Band: 433MHz

   Antenna connectors: RP-SMA connector

   Output power: 100mW (20dBm), adjustable between 1-20dBm

   Sensitivity: -117dBm sensitivity

   Interface: Standard TTL UART

   Connection status: LED indicators

9. Raspberry Pi Camera Board V2

   Resolution: 8 megapixels

   Sensor Resolution: 3280 x 2464 pixels

   Focal Length: 3.04mm

   Horizontal FOV : 62.2 degrees

   SoC: Broadcom BCM2837

10. Raspberry Pi 3

    CPU: 4× ARM Cortex-A53, 1.2GHz

    GPU: Broadcom VideoCore IV

    RAM: 1GB LPDDR2 (900 MHz)

    Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

    Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

    Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

B

Appendix: List of Papers Presented and Published

None yet.

v