AERIAL ROBOT - GROUND ROBOT COORDINATION FOR SPATIAL MAPPING AND ANALYSIS

by

Ahmet Harun Eker

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2024

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. H. Işıl Bozma, who inspired me with her dedication, diligence, and hard work. I am incredibly grateful for her patient support, faithful guidance, and for providing every possible opportunity.

I would also like to thank my thesis committee members, Prof. Yağmur Denizhan and Prof. Hakan Temeltaş, for their valuable contributions.

I am grateful to my lab colleagues, Arif Yılmaz for his organic lettuces, Yunus As for not leaving me alone at long work nights, S. Batuhan Vatan for his joyful music, Serhat İşcan for his unexpected jokes, and Semanur Kumaş for leaving me half of the couch in the lab.

I would like to express my special thanks to Ertaş Demir for his guidance and support during the mechanical assemblies and technical drawings.

Lastly, I would like to thank my mother, who has brought me to this day in challenging conditions, my sister, my father, my grandfather and grandmother, and my friends, who have always supported me.

This work has been supported in part by TUBITAK grant EEEAG 119E203 and in part by ROYAL CB SBB 2019K12-149250.

ABSTRACT

AERIAL ROBOT - GROUND ROBOT COORDINATION FOR SPATIAL MAPPING AND ANALYSIS

This thesis is concerned with the coordination of an aerial robot (AR) and a mobile ground robot (MGR) for aerial spatial mapping and analysis. Although ARs are favorable platforms for environmental mapping, their operation times are limited due to the current battery technologies. This can be alleviated through having the AR coordinate with an MGR acting as a mobile battery replenishment platform. In this thesis, this problem is studied from three aspects. First, a novel rendezvous scheduling algorithm is presented to determine where and when should the robots meet to replenish the battery of the AR. The resulting algorithms are shown to be real-time applicable in a wide range of scenarios varying with respect to landing type (dockless landing, docked landing), muling (done or not), charging times (fixed, varying), and final location constraints regardless of the mission scale while ensuring optimality for the associated mission. Second, three-dimensional spatial mapping and localization algorithms based on the sensory data as collected by the AR are formulated. Three-dimensional mapping is based on the matching of the incoming frames with the so-far constructed map. For localization, an interpolation-based method and a geometry-based method are considered. Thirdly, a real coordination system is designed. The system consists of a replaceable battery design for ARs, a battery replacement platform (BRP), and a MGR that carries this platform. The hardware and software development of this platform are initiated.

ÖZET

UZAMSAL HARİTALAMA VE ANALİZ İÇİN UÇAN ROBOT - KARA ROBOTU KOORDİNASYONU

Bu tez, uçan robotların (UR) ve gezgin kara robotlarının (GKR) havadan uzamsal haritalama ve analiz için koordinasyonu ile ilgilidir. Her ne kadar hava robotları çevresel haritalama için uygun olsalar da, mevcut batarya teknolojileri sebebiyle operasyon süreleri sınırlıdır. Bu problem, UR'yi, gezgin batarya yenileme platformu olarak görev yapan GKR ile eşgüdümlendirerek aşılabilir. Bu tezde, bu problem üç açıdan ele alınmaktadır. İlk olarak, UR'nin bataryasını yenilemek için robotların nerede ve zaman buluşacaklarını karar verebilen yeni bir randevu planlama algoritması sunulmaktadır. Ortaya çıkan algoritmaların, görev ölçeği gözetilmeksizin ilgili görev için optimaliteyi garanti ederken, iniş türü (kenetlenmesiz iniş, kenetlenmeli iniş), katırlama (yapılıp yapılmamasına göre), şarj süresi (sabit ya da değişken) ve son konum kısıtlarına göre değişen çok sayıdaki senaryoda uygulanabilir olduğu gösterilmiştir. Ardından UR ile toplanan algılayıcı verilerine dayalı üç boyutlu uzamsal haritalama ve konumlama algoritmaları formülize edilmiştir. Üç boyutlu uzamsal haritalama, gelen imgelerin şimdiye kadar oluşturulmuş haritaya izdüşümsel dönüşümüne dayanmaktadır. Konumlama için ise aradeğerleme ve geometri temelli olarak iki farklı metod değerlendirilmiştir. Son olarak, gerçek bir eşgüdüm sistemi tasarlanmıştır. Bu sistem UR'ler için değiştirilebilir bir batarya tasarımı, bir batarya değiştirme platformu (BDP) ve bu platformu taşıyan bir GKR'den oluşmaktadır. Bu tasarıma dayalı olarak bu platformun donanım ve yazılım geliştirmeleri başlatılmıştır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xii
LIST OF SYMBOLS . . . xiii
LIST OF ACRONYMS/ABBREVIATIONS . . . xv
1. INTRODUCTION . . . 1
1.1. Problem Statement . . . 1
1.1.1. Rendezvous Scheduling . . . 1
1.1.2. 3D Mapping & Vision-Based Localization . . . 2
1.1.3. Coordination System . . . 2
1.2. Contributions . . . 3
1.3. Outline . . . 4
2. RENDEZVOUS SCHEDULING . . . 5
2.1. Related Work . . . 6
2.2. Problem Formulation . . . 8
2.3. Optimal Rendezvous Schedule . . . 14
2.3.1. Finding Best Successor Charge Selections . . . 15
2.3.2. Dockless Landing . . . 16
2.3.3. Docked Landing . . . 19
2.3.4. Muling . . . 20
2.4. Complexity Analysis . . . 23
2.5. Results . . . 24
2.6. Discussion . . . 31
3. MAPPING AND LOCALIZATION . . . 32
3.1. Related Work . . . 32
3.1.1. 3D Mapping . . . 32

3.1.2. Localization . . . 34
3.2. 3D Mapping . . . 35
3.3. Visual Localization . . . 39
3.3.1. Interpolation-Based Localization . . . 40
3.3.2. Geometry-Based Localization . . . 41
3.3.2.1. Minimizing 3D Error . . . 42
3.3.2.2. Minimizing 2D Error . . . 43
3.4. Experiments and Results . . . 44
3.4.1. Mapping Experiments . . . 46
3.4.2. Localization Experiments . . . 50
3.4.2.1. Interpolation-Based Localization Experiments . . . 51
3.4.2.2. Geometry-Based Localization Experiments . . . 54
4. COORDINATING ROBOTS SYSTEM . . . 58
4.1. Related Work . . . 58
4.2. Battery Replacement Platform . . . 61
4.2.1. Easily Replaceable Battery Casing . . . 62
4.2.2. Platform Design . . . 66
4.3. Mobile Ground Robot . . . 73
4.3.1. Electronics and Electrical Design . . . 73
4.3.2. Robot Control . . . 75
4.3.3. Software Architecture . . . 78
5. CONCLUSION . . . 80
REFERENCES . . . 83
APPENDIX A: PROOFS OF PROPOSITIONS . . . 94
APPENDIX B: FINDING A RENDEZVOUS SCHEDULE . . . 99
APPENDIX C: MECHANICAL DESIGNS . . . 101

LIST OF FIGURES

Figure 2.1. Sample intermediate charge selection. 9

Figure 2.2. Flight time $\alpha(x', x)$. 11

Figure 2.3. Representation of the problem over a graph. 13

Figure 2.4. Muling from CS_x to $CS_{x''}$ after visiting P_{x_a}. 22

Figure 2.5. Sample Baylands Park scenarios. 26

Figure 2.6. Sample Marmaracık National Park scenarios. 27

Figure 2.7. Sample Ceylanpınar Agricultural Enterprise scenarios. 29

Figure 3.1. Bilinear interpolation. 36

Figure 3.2. AR with the downward field of view sensors. 45

Figure 3.3. Transferred model in simulation environment. 46

Figure 3.4. Resulting map from the short dense loop dataset. 47

Figure 3.5. Sparse datasets: Mean positional errors vs frame index. 49

Figure 3.6. Dense datasets: Mean positional errors vs frame index. 50

Figure 3.7. Dataset image locations for Back-and-Forth patterns. 51

Figure 3.8. Dataset image locations for uniform random distribution. 51

Figure 4.1. *Case_bottom*. 63

Figure 4.2. *Case_middle*. 63

Figure 4.3. *Case_lid*. 64

Figure 4.4. *Case_top*. 65

Figure 4.5. *Case_latchbolt*. 65

Figure 4.6. Assembled battery case: (a) locked and (b) unlocked states. 66

Figure 4.7. BRP top view. 67

Figure 4.8. BRP electronics plate. 71

Figure 4.9. BRP connectors bay. 72

Figure 4.10. MGR hardware. 76

Figure 4.11. MGR motion control variables. 78

Figure 4.12. MGR software architecture. 79

Figure C.1. arm_aruco. 103

Figure C.2. ball_joint_platform. 103

Figure C.3. ball_joint_linear_motor. 104

Figure C.4. case_bottom. 105

Figure C.5. case_latchbolt. 105

Figure C.6. case_lid. 106

Figure C.7. case_middle. 106

Figure C.8. case_top. 107

Figure C.9. ethernet_switch_hp_adapter. 107

Figure C.10. ethernet_switch_tnet_adapter. 108

Figure C.11. fan_adapter. 108

Figure C.12. gps_f9p_adapter. 109

Figure C.13. hokuyo_utm30lx_adapter. 109

Figure C.14. jetson_agx_orin_adapter. 110

Figure C.15. hinge_joint_linear_motor. 111

Figure C.16. mpu6050_adapter. 111

Figure C.17. pixhawk_6x_adapter. 112

Figure C.18. siyi_air_unit_adapter. 112

Figure C.19. siyi_antenna_adapter. 113

Figure C.20. siyi_ip_camera_adapter. 113

Figure C.21. XT60E_M_adapter. 114

LIST OF TABLES

Table 2.1. Coordination simulation results. 25

Table 3.1. Specifications of sensors. 45

Table 3.2. Interpolation-based localization errors 53

Table 3.3. Normalized interpolation-based localization errors 53

Table 3.4. Geometry-based localization experiments 55

Table 3.5. Interpolation-based vs geometry-based localization comparison . . 55

Table 4.1. Properties of various drive types 60

Table 4.2. Descriptions of the BRP components. 69

Table 4.3. Functions of the connectors. 72

Table 4.4. List of MGR electronic components. 74

Table C.1. Parts used in the assembly of MGR. 101

Table C.2. Parts used in the assembly of BRP. 102

LIST OF SYMBOLS

$a \in \mathcal{A}$, $P_a \in R^3$	Waypoint, position of the waypoint
$\mathcal{A}$	Ordered set of waypoints
$b \in \mathcal{B}$, $V_b \in R^3$	Charging location, position of the charging location
CS_x	Charge selection
$\mathcal{C}$	Ordered set of charge selections
$d_h(V_b, V_{b'})$	Shortest distance between charging locations V_b and $V_{b'}$ over $\mathcal{H}(\mathcal{B}, \mathcal{E})$ (m)
$d_s(P_a, P_{a'})$	Route distance between waypoints P_a and $P_{a'}$ (m)
$E_{x'x}$	Recharge time of CS_x (sec)
f_s	Average descriptor distance
G_x	Going to time of CS_x (sec)
$\mathcal{H}(\mathcal{B}, \mathcal{E})$	Graph associated with movement of the MGR
$\mathcal{I}_k$	Intersection set for CS_k
L_1	Regular grid, 0% overlap
L_2	Regular grid, 50% overlap
L_3	Regular grid, 75% overlap
L_4	Uniform random, sparse
L_5	Uniform random, dense
N, M	Number of waypoints, number of possible charge locations
N'	Number of non-initial charge selections in $\mathcal{C}$
R_x	Returning time of CS_x (sec)
$\mathcal{R}(CS_x)$	Set of charge selections in range of CS_x
$T_{x'x}$	Added time of CS_x (sec)
$T_{\downarrow}$, $T_{\uparrow}$	Landing, take-off times (sec)
$T'_{x'x \to x''}$	Time cost of CS_x when muled to $CS_{x''}$
$\lvert U \rvert$	Average size of binary search tree
w	Effective track seperation of MGR
$W_{x'x}$	Waiting time of CS_x (sec)

$x = \begin{bmatrix} x_a & x_b \end{bmatrix}^T$	Waypoint - charging location pair
v	1D linear velocity
v_r	Velocity of the right wheel of the MGR
v_l	Velocity of the left wheel of the MGR
$\alpha(x', x)$	Flight time between $CS_{x'}$ and CS_x (sec)
γ_a	Look-ahead distance for controller of the MGR
γ_A, γ_G	Average speeds of AR and MGR (m/sec)
γ_p, γ_d	Gains of proportional and derivative controllers
η	Step size of gradient descent method
η_A	Maximum flight time of AR (sec)
λ_A	Slope of the linear recharge function
ψ_{err}	Heading error
ψ_{rbt}	Heading of the robot
ω	1D angular velocity

LIST OF ACRONYMS/ABBREVIATIONS

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
ADDIM	Average Descriptor Distance Inverse Match
ADDIMSR	Average Descriptor Distance Inverse Match Square Root
AR	Aerial Robot
BP	Baylands Park
BRP	Battery Replacement Platform
CAE	Ceylanpınar Agricultural Enterprise
CC/CV	Constant Current-Constant Voltage
CPU	Central Processing Unit
DC	Direct Current
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
ERBC	Easily Replaceable Battery Casing
FLANN	Fast Library for Approximate Nearest Neighbors
FOV	Field Of View
Fps	Frame Per Second
GNSS	Global Navigation Satellite System
ID	Identification
IM	Inverse Match
IMS	Inverse Match Square
IMU	Inertial Measurement Unit
IMSR	Inverse Match Square Root
LIDAR	Light Detection and Ranging
MGR	Mobile Ground Robot
MNP	Marmaracık National Park
NP	Nondeterministic Polynomial

ORB	Orientated Fast and Robust Brief
PCL	Point Cloud Library
PD	Proportional derivative
PLA	Polylactic Acid
PnP	Perspective-n-Point
PWM	Pulse Width Modulation
RADAR	Radio Detection And Ranging
RANSAC	Random Sample Consensus
RC	Radio Control
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue and Depth
ROS	Robot Operating System
RTK	Real-Time Kinematic
SIFT	Scale Invariant Feature Transform
SkNNI	Spherical k-Nearest Neighbors Interpolation
SQPnP	Sequential Quadratic PnP
SURF	Speeded-up robust features
SVD	Singular Value Decomposition
UGV	Unmanned Ground Vehicle
USA	United States of America

1. INTRODUCTION

Aerial robots (AR) are being increasingly used in domains such as surveillance, logistics, inspection, agriculture, and terrain mapping and analysis [1, 2]. In many of these applications, the AR is required to fly over a large region for aerial terrain mapping and analysis. However, possible area coverage will be limited since mission times are restricted due to current battery technologies. This can be alleviated by having the AR coordinate with a mobile ground robot (MGR) acting as a mobile battery replenishment platform. However, this is a challenging problem as it requires the coordination of the two robots while the AR is doing its task.

1.1. Problem Statement

This thesis is focused on the coordination problem of an AR with an MGR for three-dimensional terrain mapping and analysis. This problem is addressed in three parts:

- Rendezvous scheduling problem between a single AR and a single MGR for charging coordination.
- 3D mapping of the environment and localization of the AR using onboard sensory data
- Design and development of a real coordinating robots' system

1.1.1. Rendezvous Scheduling

Most terrain mapping tasks require the AR to fly through a finite set of in-air waypoints [3–5]. If the mission is large-scale, it may not be possible for the AR to complete it in a single sortie - as it may not have enough battery capacity. This is addressed by having the AR recharge as necessary. Equivalently, recharging is also referred to as 'refueling' in the literature. Recharging can be achieved via either deploying station-

ary charge stations in the operation area or coordinating with MGR acting as a mobile charge station. In contrast to stationary charge stations, mobile on-site power delivery can be more economical or feasible - in particular for large-scale operations [6]. As such, rendezvous scheduling - namely the selection of rendezvous locations and times - for charging coordination becomes critical. Previous work on the coordination of ARs and MGRs for recharging typically aims to find the route sequence for the AR and the associated rendezvous schedule together. As the resulting problem is NP-Hard, there are no scalable exact algorithms to solve the problem [7]. Furthermore, due to the difficulty of the problem, the proposed approaches are not large-scale applicable.

1.1.2. 3D Mapping & Vision-Based Localization

ARs are being increasingly used for mapping tasks thanks to their high agility and maneuverability. Typically, the AR is endowed with an onboard downward-facing RGB camera and/or a Light Detection and Ranging (LIDAR) system. The incoming sensory data can then be used to reconstruct a map of the ground surface. This requires the alignment and fusion of observations obtained at different locations. As this is a computationally challenging problem, the proposed approaches are often slow for real-time applications. Most image stitching/mosaicking methods rely on inter-image homography computation based on a planar scene assumption. These typically provide two-dimensional maps. The map can be three-dimensional if sensory data contains depth information so that the maps can be further augmented with this knowledge. Apart from encoding terrain information, these maps can also be used for vision-based localization. This becomes particularly important in Global Navigation Satellite System (GNSS) denied environments.

1.1.3. Coordination System

The AR and MGR coordinate to provide the AR with a charged battery. Two alternative approaches are possible for the depleted battery problem: Recharging the battery or swapping the battery to a charged battery. As battery recharging is time-

consuming, downtime of the AR increases. Hence, battery swapping tends to be preferred. Such systems are also known as battery replacement systems. In this case, downtime is related to the swapping time. Of course, this requires the battery to be removable from the AR. Battery removal can be done either manually by a human operator or autonomously. In both cases, a swapping station or equivalently battery replacement platform (BRP) needs to be available for the AR to land on. The BRP can be either stationary or mobile. In the former case, it is not possible for the AR to fly long distances without ensuring that it has enough power to return to the BRP. Hence, more recent work has considered mobile platforms through using MGRs. In this case, as the AR is flying for its mission, the MGR can navigate to a nearby location so that the AR can land on it as its battery becomes depleted. Of course, such a system is much harder to realize as all three components need to be designed and developed so that autonomous battery replacement is possible.

1.2. Contributions

The contributions of the thesis can be summarized as follows:

- A novel approach for rendezvous scheduling for recharging is proposed to determine when and where the two robots should meet for recharging. The core of our approach has been presented in [8].
- Various methods for 3D mapping by and localization of an AR are investigated and their applicability to real scenarios are discussed.
- A complete system with an AR with a swappable battery, a battery-replacement platform, and an MGR capable of carrying this platform is designed. The hardware and software development of this system is initiated.

1.3. Outline

The outline of this thesis is as follows:

- In Chapter 2, the novel rendezvous scheduling approach is presented.
- In Chapter 3, three-dimensonal mapping and localization algorithms are developed.
- Chapter 4 presents the design details of a real AR-MGR coordination system.
- Finally, work done in this thesis is summarized in Chapter 5 along with a discussion of future work.

In addition, details pertaining to various parts of this work are presented in the accompanying appendix.

- Appendix A contains proofs of propositions of the rendezvous scheduling algorithm.
- The usage of the program that implements the rendezvous scheduling algorithm is explained in Appendix B.
- Appendix C contains sketches of designed parts for coordinating robots.

2. RENDEZVOUS SCHEDULING

This chapter is focused on the rendezvous scheduling problem between a single aerial robot (AR) and a single mobile ground robot (MGR) for charging coordination. In the proposed approach, the problem is decomposed into two stages. In the first stage, the route of the AR is determined by using one of the plethora of available algorithms. Of course, if the mission constraints dictate a specific route, this can also be used. Our focus is on the second stage as to determine when and where the two vehicles should meet for recharging. The contributions of this approach can be summarized as:

- The scheduling problem is formulated as a discrete optimal control problem in which the objective is to minimize the total mission time while adhering to flight time constraints between consecutive recharges.
- This problem is then solved using dynamic programming with backward induction that exploits problem-specific properties to reduce computational complexity.
- The effectiveness of the formulation and the solution is demonstrated on large-scale field data.

The proposed approach is shown to have the following advantages:

(i) Flexibility: It can be used in various charging scenarios varying with respect to charging setup, muling, charging times, and final location constraints. Two alternatives are possible for charging setup: i) Dockless landing - namely, the AR can land prior to the arrival of the MGR; ii) Docked landing - namely, the AR has to hover in the air and wait for the MGR's arrival. Muling - transportation of the AR by MGR - can be done or not. Charging times can be constant (battery replacement) or charge-level dependent (battery recharging). Finally, the two vehicles can be required to end up at specific charging locations at mission completion.

(ii) Optimality: As the problem is decomposed into two relatively less complex prob-

lems, each becomes more tractable, and the optimal rendezvous schedule for a given route can be found in realizable time, even for large-scale missions.

(iii) Scalability: It can be used in large-scale missions and in real time.

To our knowledge, this is the first study that can find the optimal solution to the problem with a long route (i.e., a large number of aerial waypoints and charging locations) in realizable time for varying realistic scenarios. Interestingly, the proposed approach turns out to have performance close to the optimistic lower bound of the solution when the route sequence and rendezvous scheduling are solved together (as is typically done), as demonstrated in our simulations.

The outline of the chapter is as follows: Related literature is discussed in Section 2.1. The problem is formulated in Section 2.2. The proposed approach is explained in Section 2.3. The computational complexity of the proposed approach is analyzed in Section 2.4. Simulation results from an extensive study, including a comparative study, are presented in Section 2.5. In Section 2.6, the issues that may arise in on-robot applications of the proposed approach are discussed. Appendix A contains the proofs of all propositions.

2.1. Related Work

The rendezvous scheduling problem has been extensively studied - considering either electric or fuel-constrained vehicles [9–12]. Typical applications are tasks requiring delivery [10], [13], area coverage [14–16], and persistent monitoring [17–20]. The proposed work can be categorized into two groups depending on whether charging is assumed to be always available or not.

In the former, as charge stations are assumed to be stationary, the focus has been to ensure that recharging is done as needed - such as driving the AR to a fixed charging station as its energy level becomes critical [14], [21]. Work reported in [13], [22–26] addresses the optimal placement of the charging depots as to maximize the service area

with the minimum number of charging depots. Formulations based on mixed-integer linear programming have been introduced - considering one AR [27] and multiple ARs [6], [15], [17], [22], [28, 29]. More recently, deep reinforcement learning has also been utilized for AR coverage path planning under power constraints [16].

The second group considers mobile charging stations - as the availability of charging at all stations at all times may not be feasible due to limited resources. In such cases, it suffices for the charging to be available as needed. This can be achieved through the deployment of MGRs acting as mobile charging stations. The proposed approaches vary depending on their assumptions regarding the problem specification: i) Single [30, 31] or multiple ARs [23], [32] are employed; ii) Single [30], [33, 34] or multiple MGRs [32] are employed; iii) Route planning required [32] or not [35]; iv) Charging locations being on AR routes [33] or at different locations [14]; v) Potential charging locations being fixed [14], [36] or need to be determined [5], [30], [33]; vi) Muling (transporting of ARs by MGRs) done or not [31] or not [30]; vii) Type and capacity of charging such as servicing a single AR or multiple ARs at a time [37]; viii) Task repetitiveness such as one-time monitoring or persistent monitoring [38].

In this thesis, we consider one-time monitoring with one AR and one recharging MGR - also referred to as fuel-constrained unmanned aerial vehicle routing problem with a mobile refueling station. In most work, the routing and scheduling problems are solved together [7], [9], [30, 31], [39–41]. However, scalability to a high number of waypoints is limited due to high computational complexity. Alternatively, two-stage approaches solve routing and scheduling problems separately. Interestingly, there has been relatively less work that focuses only on the schedule generation aspect of the problem. In [5], once the routes of the AR and MGR are determined, the rendezvous scheduling with stochastic energy consumption of the AR is formulated as a Chance Constrained Markov Decision Process. However, the algorithm scales exponentially with the planning horizon and the number of robots. In [35], while the proposed approach is an efficient and hence scalable algorithm, optimality is not guaranteed. In this thesis, we also present a two-stage approach. However, differing from related

work, the resulting rendezvous schedule is shown to be optimal while being large-scale applicable. A preliminary version of this approach has been presented in [8]. Here, we extend on that presentation by expanding the covered charging scenarios, revising the mathematical formulation, giving the proofs of all the propositions, analyzing the complexity of the associated algorithms, and conducting a more extensive evaluation of the approach using real terrain data from three different areas.

2.2. Problem Formulation

Consider a AR that is capable of landing at a charging location, recharging, and then taking off to continue with its mission.

Assumptions: We make the following assumptions, which are typical in the literature.

- The AR has a constant maximum flight time and moves with a known average speed [7], [31], [41–43].
- The MGR does not have energy limitation and it moves with a known average speed [7], [31], [35], [44].
- The MGR navigates on a road network and the potential rendezvous locations are finitely many with locations known a prior [42]. This is a realistic assumption since topographic or environmental constraints typically either do not allow the AR to land at arbitrary locations or the MGR to reach such arbitrary locations [30].
- The AR can land immediately as it reaches the recharge site even if the MGR has not arrived yet (dockless landing), or is required to hover in the air until the MGR arrives at that location (docked landing) [7], [44]. The latter case may occur if the landing area is not physically suitable for the AR to wait on the ground or the AR can not land in the absence of the MGR - i.e. the AR lands on a dock carried by the MGR.
- The charging times are either constant (battery replacement) [32], [40] or linear functions of the flight time (battery recharge) [31], [38].

- If MGR is capable of muling the AR, muling can also be considered [31], [38] or not [7], [43].
- At the end of the mission, the two vehicles can meet at any of the potential rendezvous locations to minimize the completion time or at a specific location.

Problem Formulation: Suppose the AR's route is given by the ordered set $\mathcal{A} = \{1, 2, \ldots, N\}$ of waypoints - where each waypoint $a \in \mathcal{A}$ is associated with an in-air position $P_a \in \mathbb{R}^3$ that it needs to fly through consecutively depending on its order. Let its average speed be $\gamma_A > 0$ m/sec and assume it to have limited energy so that its maximum flight time between consecutive recharges is η_A seconds. Let the average speed of the MGR be $\gamma_G \geq 0$ m/sec. The AR coordinates with the MGR to recharge as needed at mutually agreed locations and times. These locations are referred to as charging locations. The movement of the MGR is associated with a road network, represented by the graph H$(\mathcal{B}, \mathcal{E})$ with nodes as indexed by the set $\mathcal{B} = \{1, \ldots, M\}$ and edges $\mathcal{E}$. Each node $b \in \mathcal{B}$ is associated with a position $V_b \in \mathbb{R}^3$ that designates a potential charging location and edges indicate the availability of direct routes between the corresponding charging locations. Finally, let $x \in \mathcal{A}' \times \mathcal{B}$, where $\mathcal{A}' = \mathcal{A} \cup \{0\}$, denote the waypoint $x_a \in \mathcal{A}'$ - charging location $x_b \in \mathcal{B}$ pair - namely $x = \begin{bmatrix} x_a & x_b \end{bmatrix}^T$. At the mission onset, the AR and MGR are assumed to be initially co-located at the charging location - as denoted by x^0 such that $P_{x_a^0} = V_{x_b^0}$ for some $x_b^0 \in \mathcal{B}$.

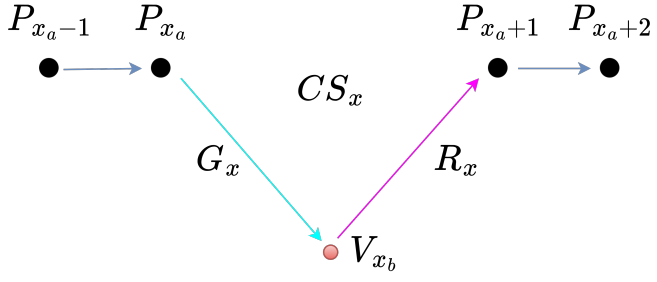


Figure 2.1. A sample intermediate charge selection CS_x associated with waypoint P_{x_a} and charging location V_{x_b}.

The AR-MGR coordination is achieved through the rendezvous schedule. The rendezvous schedule is defined by the ordered set $\mathcal{C} = \{CS_{x^i} \mid i = 0, 1, 2, \ldots, N' \leq N\}$ of N' charge selections. A charge selection represents the meeting of the two vehicles for recharge and is defined by a waypoint-charging location pair. Depending on the index of the associated waypoint, three types of charge selections are defined: initial charge selection, intermediate charge selections, and final charge selections. An intermediate charge selection CS_x denotes that after visiting the waypoint at P_{x_a}, the AR goes to location V_{x_b} for recharging and then returns to P_{x_a+1} to continue with its route in $\mathcal{A}$ as shown in Figure 2.1. Each intermediate charge selection CS_x is associated with an added time $T_{x'x}$ spent for that recharge. Its value depends both on the current charge selection as well as the previous charge selection $CS_{x'}$ as

$$T_{x'x} = G_x + W_{x'x} + T_{\downarrow} + E_{x'x} + T_{\uparrow} + R_x. \tag{2.1}$$

The terms G_x and R_x refer to the time required to reach the charging location from P_{x_a} and the time required to reach the next waypoint from the charging location respectively as

$$G_x = \frac{d(P_{x_a}, V_{x_b})}{\gamma_A}, \qquad R_x = \frac{d(V_{x_b}, P_{x_a+1})}{\gamma_A} \tag{2.2}$$

where $d(p, q)$ is the distance between points p and q. The terms $T_{\downarrow}$ and $T_{\uparrow}$ refer to the times required for landing and taking off respectively and are assumed to be constant. Both $W_{x'x}$ and $E_{x'x}$ values depend on the previous charge selection. The term $W_{x'x}$ represents the time that AR has to wait until the MGR reaches V_{x_b} from the previous charging location and can be calculated as

$$W_{x'x} = \max\left\{0, \frac{d_h(V_{x'_b}, V_{x_b})}{\gamma_G} - \alpha(x', x)\right\}. \tag{2.3}$$

Here, $d_h(V_{x'_b}, V_{x_b})$ is the shortest path distance that is associated with going from $V_{x'_b}$ to V_{x_b} over the edges in the graph $H(\mathcal{B}, \mathcal{E})$. For all $x'_b, x_b \in \mathcal{B}$, these distances can be computed using Floyd–Warshall or Johnson's algorithm. The term $\alpha(x', x)$ corresponds to the flight time since the previous charge selection as shown in Figure 2.2. It is equal to

$$\alpha(x', x) = T_{\uparrow} + R_{x'} + \frac{d_s(P_{x'_a+1}, P_{x_a})}{\gamma_A} + G_x + T_{\downarrow}. \tag{2.4}$$

Here, $d_s(P_{x'_a+1}, P_{x_a})$ denotes the route distance between waypoints associated with charge selections $CS_{x'}$ and CS_x,

$$d_s(P_{x'_a}, P_{x_a}) = \begin{cases} \sum_{l=x'_a}^{x_a-1} d(P_l, P_{l+1}) & \text{if } x'_a \leq x_a \\ d_s(P_{x_a}, P_{x'_a}) & \text{Otherwise.} \end{cases} \tag{2.5}$$

Note that $W_{x'x} = 0$ means the MGR has arrived at the charging location before the AR. For $W_{x'x} > 0$, the AR may wait either on the ground or in the air. The former is referred to as 'Dockless Landing' while the latter is referred to as 'Docked Landing'. In the latter case, waiting causes extra energy consumption and the MGR is required to reach the charging location before the battery of the AR drains.

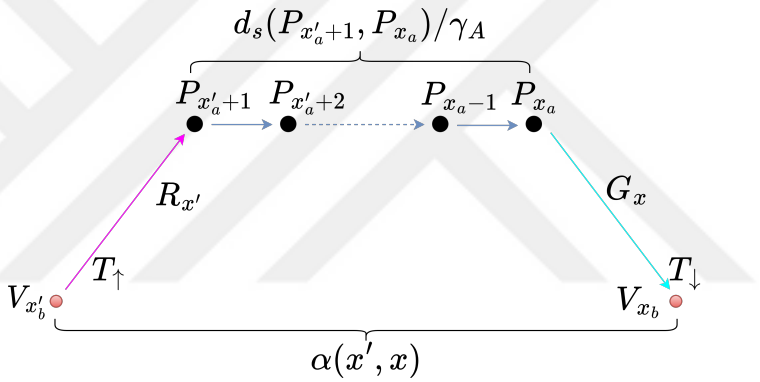


Figure 2.2. Flight time $\alpha(x', x)$.

The value $E_{x'x}$ is the time spent for recharging at V_{x_b}. It can be either constant, $E_{x'x} = E$ (i.e. battery replacement), or determined by a linear function of flight time. In the latter case, at each charge selection, the AR is recharged just enough to renew the energy that is spent to reach that charge selection from the previous charge selection; hence, the recharge time of a charge selection depends only on the preceding charge selection. In the case of Dockless Landing,

$$E_{x'x} = \alpha(x', x)\lambda_A \tag{2.6}$$

where λ_A is the slope of the linear recharge function. In the case of Docked Landing,

$$E_{x'x} = \Big(\alpha(x', x) + W_{x'x}\Big)\lambda_A. \tag{2.7}$$

As the AR and MGR are assumed to be initially co-located at the mission onset, the mission start is represented by the initial charge selection CS_{x^0} associated with a fixed

time requirement T_{x^0},

$$T_{x^0} = T_{\uparrow} + R_{x^0} \text{ where } R_{x^0} = \frac{d(V_{x_b^0}, P_1)}{\gamma_A}. \tag{2.8}$$

The final charge selections $CS_{x^{N'}}$ is such that the AR visits the waypoint $x_a^{N'} = N$ prior to the landing. In this case, the associated added time $T_{x'x^{N'}}$ is

$$T_{x'x^{N'}} = G_{x^{N'}} + W_{x'x^{N'}} + T_{\downarrow} + E_{x'x^{N'}}. \tag{2.9}$$

Once a rendezvous schedule $\mathcal{C}$ is determined, starting the mission from x^0, the mission time $J(x^0, \mathcal{C})$ is defined by the sum of flight times between waypoints associated with consecutive charge selections and the charging times,

$$J(x^0, \mathcal{C}) = T_{x^0} + \sum_{i=1}^{N'} \Big(\frac{d_s(P_{x_a^{i-1}+1}, P_{x_a^i})}{\gamma_A} + T_{x^{i-1}x^i} \Big). \tag{2.10}$$

The term $d_s(P_{x_a^{i-1}+1}, P_{x_a^i})/\gamma_A$ represents the time to cover the route between locations $P_{x_a^{i-1}+1}$ and $P_{x_a^i}$ associated with waypoints $x_a^{i-1}+1, x_a^i \in \mathcal{A}$. The third term $T_{x^{i-1}x^i}$ denotes the added time at charge selection associated with CS_{x^i}, after having recharged at $CS_{x^{i-1}}$.

Finally, the optimal rendezvous scheduling is defined as finding the ordered set $\mathcal{C}$ of charge selections that minimize $J(x^0, \mathcal{C})$ subject to AR path constraints $x_a^{i-1} < x_a^i$, $\forall i \in \{1, 2, \ldots, N'\}$ and in-air time constraints regarding maximum flight times between recharges $\forall i \in \{1, 2, \ldots, N'\}$. In the case of Dockless Landing these constraints are as

$$\alpha(x^{i-1}, x^i) \leq \eta_A. \tag{2.11}$$

With Docked Landing,

$$\alpha(x^{i-1}, x^i) + W_{x^{i-1}, x^i} \leq \eta_A. \tag{2.12}$$

The following proposition shows that the mission time $J(x^0, \mathcal{C})$ can be equivalently expressed.

Proposition 2.1.

$$J(x^0, \mathcal{C}) = \frac{d_s(P_1, P_N)}{\gamma_A} + T_{x^0} + \sum_{i=1}^{N'} T'_{x^{i-1}x^i} \text{ where} \tag{2.13}$$

$$T'_{x'x} = T_{x'x} - \frac{d(P_{x_a}, P_{x_a+1})}{\gamma_A}, \quad T'_{x'x^{N'}} = T_{x'x^{N'}}. \tag{2.14}$$

As the first two terms are constant for a given ordered set of waypoints, the optimal solution can be found by finding the ordered set $\mathcal{C}$ that results in minimum $\sum_{i=1}^{N'} T'_{x^{i-1}x^i}$.

Depending on the mission, the vehicles may be required to be at some subset of the potential charging locations at the end. Three different cases can be considered [42]: i) Radiating for the scenario when any of the charging locations can be the final charging location; ii) Circulating if vehicles are required to be at the initial charging location at the mission end; and iii) Segment for the scenario when the final charging location is required to be in a set of specific charging locations. These cases can be covered in our formulation by using the term desired final charge selection for the resulting final charge selections of the related cases.

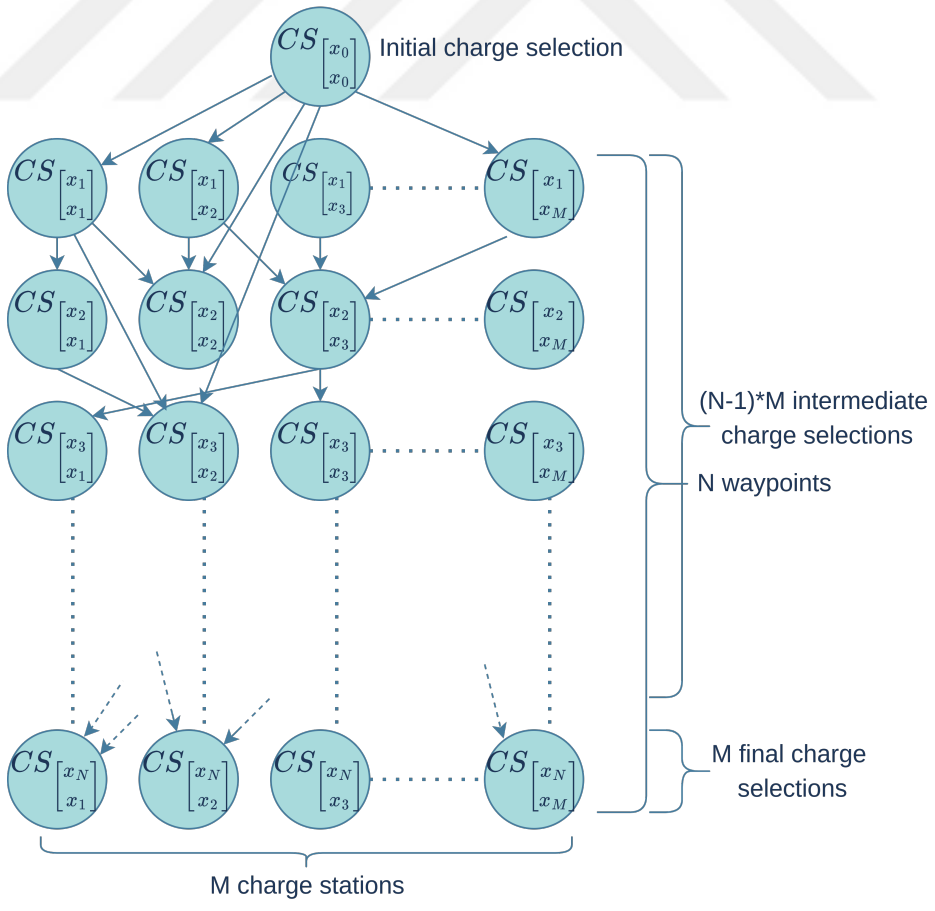


Figure 2.3. Representation of the problem over a graph.

Optimal rendezvous scheduling can possibly be posed as a search problem over a directed graph that consists of NM+1 charge selections as nodes and edges as defined by the constraints of the problem as shown in Figure 2.3. The optimal solution corresponds to the shortest path from the initial charge selection to any of the desired final selections. Unfortunately, the construction of this graph and the search for the optimal path on it are both computationally expensive - since the time cost of a charge selection depends on the preceding charge selection and the number of edges increases with increased η_A. In this thesis, we propose a dynamic programming based solution that does not require the construction of the complete graph. Rather, our solution requires the processing of only the edges that are eligible to be in the optimal solution. In particular, the edges of a node CS_x are constructed in a prioritized manner - making use of the edges of the previously visited node.

2.3. Optimal Rendezvous Schedule

For each charge selection CS_x, let $J^*(x, \mathcal{C}_x)$ denote the smallest possible total cost of the remaining charge selections from CS_x up to mission completion as

$$J^*(x, \mathcal{C}_x) = \begin{cases} 0 & \text{if } x_a = N \\ \min_{CS_{x'} \in \mathcal{R}(CS_x)} T'_{xx'} + J^*(x', \mathcal{C}_{x'}) & \text{Otherwise.} \end{cases} \tag{2.15}$$

As reaching the final waypoint means the end of a mission, $CS_{x^{N'}}$, $J^*(x^{N'}, \emptyset) = 0$ for the desired final charge selections. For the remaining charge selections, the minimum cost is selected from the range $\mathcal{R}(CS_x)$ of the charge selection CS_x. The range $\mathcal{R}(CS_x)$ contains all the charge selections that can be reached from P_{x_a+1} up to $P_{x'_a}$ and still meet with MGR at $V_{x'_b}$ without violating the constraints of the problem. For the 'Dockless Landing' case, it is defined as

$$\mathcal{R}(CS_x) = \{CS_{x'} \mid x'_a > x_a,\, \alpha(x, x') \leq \eta_A\}\,. \tag{2.16}$$

For the 'Docked Landing' case, it is defined as

$$\mathcal{R}(CS_x) = \{CS_{x'} \mid x'_a > x_a,\, \alpha(x, x') + W_{xx'} \leq \eta_A\}\,. \tag{2.17}$$

Thus, given a charge selection CS_x, its best successor is $CS_{x^*} \in \arg\min_{CS_{x'} \in \mathcal{R}(CS_x)} T'_{xx'} + J^*(x', \mathcal{C}_{x'})$. Hence, given an initial x^0, the solution to the problem of finding the optimal

rendezvous schedule corresponds to the sequence of best successor charge selections. The backward induction is realized by starting with the final charge selections and searching backward for the best successors until finding the best successor selection of the initial charge selection.

2.3.1. Finding Best Successor Charge Selections

The best successor charge selections can be determined efficiently considering two index sets - $\mathcal{O}_{\leftarrow}$ and $\mathcal{O}_{\rightarrow}$ - associated with charge selections. The index set $\mathcal{O}_{\leftarrow} = \{l\}_{l=1}^{NM}$ as defined $v(l) \in \mathcal{A} \times \mathcal{B}$ orders intermediate and final charge selections $CS_{v(l)}$ in an increasing manner with respect to the distance of corresponding rendezvous location to the mission's starting location, such that $\forall l, l' \in \mathcal{O}_{\leftarrow}$ with $l < l'$,

$$G_{v(l)} + \frac{d_s(P_1, P_{v(l)_a})}{\gamma_A} \leq G_{v(l')} + \frac{d_s(P_1, P_{v(l')_a})}{\gamma_A}. \tag{2.18}$$

For each non-final charge selection CS_x, define the largest index $p(x) \in \mathcal{O}_{\leftarrow}$ as

$$p(x) = \sup \left\{ l \in \mathcal{O}_{\leftarrow} \middle| R_x + \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} + G_{v(l)} + \frac{d_s(P_1, P_{v(l)_a})}{\gamma_A} \leq \eta'_A \right\} \tag{2.19}$$

where $\eta'_A = \eta_A - T_{\uparrow} - T_{\downarrow} + \frac{d_s(P_1, P_N)}{\gamma_A}$. The following proposition limits candidate successor charge selections in $\mathcal{O}_{\leftarrow}$.

Proposition 2.2. *For each non-final charge selection CS_x, if $l > p(x)$, then $CS_{v(l)} \notin \mathcal{R}(CS_x)$.*

The index set $\mathcal{O}_{\rightarrow} = \{k\}_{k=1}^{(N-1)M+1}$ designates the initial and intermediate charge selections $CS_{z(k)}$, $z(k) \in \begin{bmatrix} x_a^0 & x_b^0 \end{bmatrix}^T \cup \Big(\mathcal{A} - \{N\}\Big) \times \mathcal{B}$ ordered in a decreasing manner with respect to the distance of corresponding rendezvous location to the mission's end location. Thus, $\forall k, k' \in \mathcal{O}_{\rightarrow}$ with $k < k'$,

$$R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A} \geq R_{z(k')} + \frac{d_s(P_{z(k')_a+1}, P_N)}{\gamma_A}. \tag{2.20}$$

Proposition 2.3 holds for the set $\mathcal{O}_{\rightarrow}$.

Proposition 2.3. *$\forall k, k' \in \mathcal{O}_{\rightarrow}$, if $k < k'$, then $p(z(k)) \leq p(z(k'))$.*

2.3.2. Dockless Landing

In this case, the AR can land immediately as it reaches a recharge site. The MGR may have already arrived or the AR may wait on the ground for the MGR to reach this location. Proposition 2.4 can be shown to hold.

Proposition 2.4. *Consider $k \in \mathcal{O}_{\rightarrow}$ and the associated charge selection $CS_{z(k)}$. If the best successor $CS_{z(k)^*}$ of the charge selection $CS_{z(k)}$ is not a final charge selection, $CS_{z(k^*)}$, then $k^* > k$.*

Propositions 2.2 and 2.4 imply that for each $k \in \mathcal{O}_{\rightarrow}$, the associated charge selection $CS_{z(k)}$ has a best successor $CS_{z(k)^*}$ that is in the intersection set $\mathcal{I}_k$ - namely $CS_{z(k)^*} \in \mathcal{I}_k$ where

$$\mathcal{I}_k = \big\{CS_{v(l)} | l = 1, \dots, p(z(k))\big\} \cap \Big(\big\{CS_{z(k')} | k' = k+1, \dots (N-1) \times M + 1\big\} \cup \{CS_x | x = \{N\} \times \mathcal{B}\}\Big). \tag{2.21}$$

Best successor charge selections can be determined consecutively by traversing down $\mathcal{O}_{\rightarrow}$ starting with $k = (N-1)M + 1$. Furthermore, as $k = (N-1)M + 1, \dots, 1$, it is possible to iteratively construct $\mathcal{I}_{k-1}$ from $\mathcal{I}_k$ using Proposition 2.3 and definition of the $\mathcal{I}_k$ as

$$\mathcal{I}_{k-1} = \begin{cases} \text{If } CS_{z(k)} \in \{CS_{v(l)} | l = 1, \dots, p(z(k)\} : \\ \mathcal{I}_k \cup \{CS_{z(k)}\} - \{CS_{v(l)} | l = p(z(k-1)) + 1, \dots, p(z(k))\} \\ \text{Otherwise :} \\ \mathcal{I}_k - \{CS_{v(l)} | l = p(z(k-1)) + 1, \dots, p(z(k))\}. \end{cases} \tag{2.22}$$

Thus, for each charge selection $CS_{z(k)}$, finding the best successor $CS_{z(k)^*}$ amounts to a search of $\mathcal{I}_k$. The efficiency of this search can be increased by representing $\mathcal{I}_k$ with an evolving binary search tree U_k so that to find the $CS_{z(k)^*}$, the whole set $\mathcal{I}_k$ is not searched. In this tree, each node n is associated with a charge selection $CS_{f(n)}$ with key value $\hat{h}(n) = \hat{T}'_{f(n)} + J^*(CS_{f(n)}, \mathcal{C}_{f(n)})$ - namely a tentative cost that does not depend on the preceding charge selection. This way, U_{k-1} can be retrieved from U_k without changing the key values of the existing nodes. Note that while searching for the best successor of $CS_{z(k)}$, the actual value of $\hat{h}(n)$ is $h(n) = T'_{z(k)f(n)} + J^*(CS_{f(n)}, \mathcal{C}_{f(n)})$,

which requires computation of $W_{z(k)f(n)}$ and $E_{z(k)f(n)}$, and depends both on $CS_{f(n)}$ as well as $CS_{z(k)}$. The tentative value is obtained by considering $T'_{z(k)f(n)}$ with a tentative waiting time $\hat{W}_{f(n)} = 0$ that does not overestimate the actual waiting time $W_{z(k)f(n)}$, and in case of linear charge function, a tentative charge time, $\hat{E}_{f(n)} = (\frac{d_s(P_1,P_{f(n)_a})}{\gamma_A} + G_{f(n)} + T_{\downarrow} + T_{\uparrow}) * \lambda_A$. Other terms in $\hat{T}'_{f(n)}$ are used as their original values as they do not depend on the preceding charge selection, $CS_{z(k)}$. The tentative recharge time overestimates the actual value $E_{z(k)f(n)}$ by $\Omega_{z(k)} = (\frac{d_s(P_1,P_{z(k)_a+1})}{\gamma_A} - R_{z(k)}) * \lambda_A$. However, as the overestimation amount depends only on the respective $CS_{z(k)}$, it is the same for all the elements in U_k. Thus, the value $\hat{h}(n) - \Omega_{z(k)} \leq h(n)$ can be considered as an admissible heuristic while searching for the best successor of $CS_{z(k)}$,

$$CS_{z(k)^*} = CS_{f(n^*)} \text{ where } n^* \in \underset{\substack{n \in U_k \\ CS_{f(n)} \in \mathcal{R}(CS_{z(k)})}}{\arg\min} h(n). \tag{2.23}$$

The search for the best successor charge selection of each charge selection is done through an in-order traversal of U_k as presented in Algorithm 1. Here, $N_I(n)$ denotes the next node of n during the in-order traversal of U_k that satisfies range conditions - as some nodes of U_k can possibly be outside the range of the charge selection that is being visited due to AR path constraints. The traversal continues until finding a node n with $h(n) < \hat{h}(n') - \Omega_{z(k)}$. Algorithm 1 is guaranteed to find the best successor via comparing $h(n)$ of a node with $\hat{h}(n') - \Omega_{z(k)}$ of the succeeding nodes during the in-order traversal of the tree. The optimal rendezvous schedule then can be found by iteratively evolving the U_k and searching for the best successor charge selection of all the charge selections as given in Algorithm 2.

Algorithm 1 Finding $z(k)^*$ in U_k

Input: $U_k, z(k)$
Output: $z(k)^*$
$n \leftarrow N_I(0)$, $n' \leftarrow N_I(n)$
while $h(n) > \hat{h}(n') - \Omega_{z(k)}$ **do**
    if $h(n) > h(n')$ **then** ▷ n' is a better successor
        $n \leftarrow n'$
    end if
    $n' \leftarrow N_I(n')$
end while
$z(k)^* \leftarrow f(n)$

Algorithm 2 Finding $\mathcal{C}$ - Dockless landing

For all intermediate and final CS_x, compute $\hat{T}'_x$; $\forall CS_{x^{N'}}$, set $J^*(x^{N'}, \emptyset) = 0$
$k \leftarrow (N-1)M + 1$, $l \leftarrow NM$, initialize U_k with desired final charge selections
while $k \geq 1$ **do** ▷ consider $CS_{z(k)} \in O_{\rightarrow}$ with index k
    while $l > p(z(k))$ **do**
        $U_k \leftarrow U_k - \{CS_{v(l)}\}$, $l \leftarrow l - 1$
    end while
    Using Algorithm 1, find $z(k)^*$ in U_k.
    if $z(k)^* \neq NIL$ **then** ▷ $CS_{z(k)}$ has a best successor
        $J^*(z(k), \mathcal{C}_{z(k)}) \leftarrow J^*(z(k)^*, \mathcal{C}_{z(k)^*}) + T'_{z(k)z(k)^*}$
        if $CS_{z(k)} \in \{CS_{v(l')} | l' = 1, \ldots, l\}$ **then**
            $U_k \leftarrow U_k \cup \{CS_{z(k)}\}$
        end if
    end if
    $U_{k-1} \leftarrow U_k$
    $k \leftarrow k - 1$
end while
Starting from CS_{x^0}, track the best successors until reaching a final charge selection to get the $\mathcal{C}$

2.3.3. Docked Landing

In this scenario, the AR can possibly hover in the air as it needs to wait for the MGR to arrive at the rendezvous location before landing. It differs from the previous 'Dockless Landing' case in various aspects: First, as the AR spends energy while waiting for the MGR, the definition of the range is different, and Proposition 2.4 is not valid. As the computations for the best successor charge selection of a charge selection must be done prior to the computations of a charge selection, charge selections can not be traversed with the decreasing order of k as in the 'Dockless Landing' scenario. As the best successor of a charge selection must be associated with a larger aerial waypoint index, the computations of the best successors are done in decreasing order with respect to the associated waypoint indices. Computation of the best successors for the charge selections with the same waypoint can be done in arbitrary order. In this case, the best successor of a charge selection $CS_{u(a,b)}$, where $u(a,b) = x$ with $x_a = a$ and $x_b = b$, $CS_{u(a,b)^*}$ is in the intersection set $\mathcal{I}_a^b$ - namely - $CS_{u(a,b)^*} \in \mathcal{I}_a^b$ where

$$\mathcal{I}_a^b = \left\{CS_{v(l)} | l = 1, \dots p(u(a,b))\right\} \cap \left\{CS_{x'} \mid x'_a = a+1, \dots, N; x'_b = 1, \dots, M\right\}. \quad (2.24)$$

With this computation order, it is not possible to construct a single intersection set iteratively from the previously computed charge selection. Rather, one needs to evolve M intersection sets $\mathcal{I}_a^b$, $b = 1, \dots, M$ - with each one associated with the charge selections with the same charging location, to ensure that the best successor of a charge selection $CS_{u(a,b)}$ is in the associated intersection set I_a^b.

Among all non-final charge selections associated with the same charging location, each with a smaller waypoint index will also have a corresponding $p(x)$ index that is smaller than or equal to that of the other as shown by Proposition 2.5.

Proposition 2.5. $\forall x, x' \in \begin{bmatrix} x_a^0 & x_b^0 \end{bmatrix}^T \cup \left(\mathcal{A} - \{N\}\right) \times \mathcal{B}$, *if* $x_b = x'_b$ *and* $x_a < x'_a$, *then* $p(x) \leq p(x')$.

Thus, as $a = N, \ldots, 1$, it is possible to iteratively construct the intersection set $\mathcal{I}^b_{a-1}$ of $CS_{u(a-1,b)}$ from intersection set $\mathcal{I}^b_a$ of $CS_{u(a,b)}$ as

$$\begin{aligned}\mathcal{I}^b_{a-1} =& \mathcal{I}^b_a \cup \Big\{CS_{x'} \mid x'_a = a; x'_b = 1, \ldots, M; CS_{x'} \in \{CS_{v(l)} | l = 1, \ldots, p(u(a,b))\}\Big\} \\ & - \big\{CS_{v(l)} | l = p(u(a-1,b)), \ldots, p(u(a,b))\big\}. \end{aligned} \tag{2.25}$$

Similar to the former case, the search in $\mathcal{I}^b_a$ can be done efficiently via evolving a binary search tree U^b_a. In each tree, tentative $\hat{T}'_{f(n)}$ values can be computed the same as that of the Dockless Landing case. Algorithm 1 can be used to find $CS_{u(a,b)^*}$ in U^b_a. The optimal rendezvous scheduling is then determined by evolving M binary search trees as shown in Algorithm 3 and searching for the best successor selections in them using Algorithm 1.

2.3.4. Muling

Muling is considered for the Docked Landing scenario, as the AR must be docked to the MGR during land transportation. It can be viewed as jumping from one intermediate charge selection node to another node at the same row in the graph shown in Figure 2.3, with additional land transportation costs. In this case, as the AR does not visit any waypoints during muling, aerial point indices of the two charge selection nodes are the same. We name such charge selections as muled charge selections, consisting of a source CS_x and a destination $CS_{x''}$ charge selection as shown in Figure 2.4, with $CS_{x'}$ being the previous charge selection of CS_x. The additional time cost brought to the mission by a muled charge selection, $T'_{x'x \to x''}$ is

$$T'_{x'x \to x''} = G_x + W_{x'x} + T_\downarrow + E_{x'x} + \frac{d_h(V_{x_b}, V_{x''_b})}{\gamma_G} + T_\uparrow + R_{x''} - \frac{d(P_{x_a}, P_{x_a+1})}{\gamma_A}. \tag{2.26}$$

Algorithm 3 Finding $\mathcal{C}$ - Docked landing

For all intermediate and final CS_x, compute $\hat{T}'_x$; $\forall CS_{x^{N'}}$, set $J^*(x^{N'}, \emptyset) = 0$
$a \leftarrow N-1$, $\forall b \in \mathcal{B}$ initialize U_a^b with desired final charge selections and set $l^b \leftarrow NM$
▷ for each M intersection set, associated l^b to be swiped
while $a \geq 1$ **do**
    for $\forall b \in \mathcal{B}$ **do**
        while $l^b > p(u(a,b))$ **do**
            $U_a^b \leftarrow U_a^b - \{CS_{v(l^b)}\}$, $l^b \leftarrow l^b - 1$
        end while
        Using Algorithm 1, find $u(a,b)^*$ in U_a^b
        if $u(a,b)^* \neq NIL$ **then**
            $J^*(u(a,b), \mathcal{C}_{u(a,b)}) \leftarrow J^*(u(a,b)^*, \mathcal{C}_{u(a,b)^*}) + T'_{u(a,b)u(a,b)^*}$
        end if
    end for
    If muling is considered, apply Algorithm 4.
    for $\forall b \in \mathcal{B}$ **do**
        for $\forall m \in \mathcal{B}$ **do**
            if $CS_{u(a,b)} \in \{CS_{v(l')} | l' = 1, \dots l^m\}$ and $u(a,b)^* \neq NIL$ **then**
                $U_a^m \leftarrow U_a^m \cup \{CS_{u(a,b)}\}$
            end if
        end for
        $U_{a-1}^b \leftarrow U_a^b$
    end for
    $a \leftarrow a - 1$
end while
Find the best successor of CS_{x^0} in the associated tree $U_0^{b_0}$
Starting from CS_{x^0}, track the best successors until reaching a final charge selection to get the $\mathcal{C}$

Without muling, after recharging at $CS_{x'}$, recharging at CS_x would yield a total remaining cost of $T'_{x'x} + J^*(x, \mathcal{C}_x)$. With muling from CS_x to $CS_{x''}$, the total remaining cost would be $T'_{x'x\to x''} + J^*(x'', \mathcal{C}_{x''})$. Thus, if $T'_{x'x\to x''} + J^*(x'', \mathcal{C}_{x''}) < T'_{x'x} + J^*(x, \mathcal{C}_x)$ or equivalently, $J^*(x'', \mathcal{C}_{x''}) + \frac{d_h(V_{x_b}, V_{x''_b})}{\gamma_G} + R_{x''} - R_x < J^*(x, \mathcal{C}_x)$, muling reduces the remaining cost for CS_x to $J^*(x'', \mathcal{C}_{x''}) + \frac{d_h(V_{x_b}, V_{x''_b})}{\gamma_G} + R_{x''} - R_x$. Note that this value does not depend on $CS_{x'}$. Hence, a muling gain for CS_x can be sought without considering its previous charge selection, $CS_{x'}$, but only the muling destination charge selections with the same aerial index. This can be done by using Algorithm 4 in the Algorithm 3 as shown.

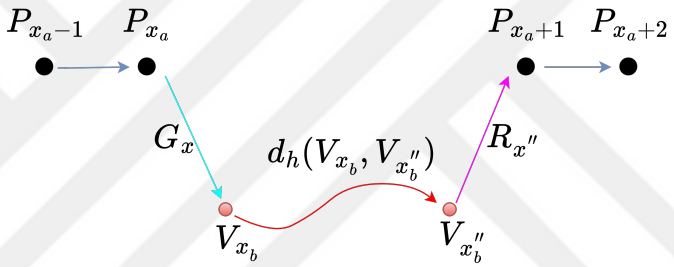


Figure 2.4. Muling from CS_x to $CS_{x''}$ after visiting P_{x_a}.

Algorithm 4 Seeking muling gains

At a given a, after calculating $J^*(x, \mathcal{C}_x) \forall x_b \in \mathcal{B}$
for $\forall x$ s.t. $x_a = a, x_b \in \mathcal{B}$ **do**
    for $\forall x''$ s.t. $x''_a = a, x''_b \in \mathcal{B}$ **do**
        if $J^*(x'', \mathcal{C}_{x''}) + \frac{d_h(V_{x_b}, V_{x''_b})}{\gamma_G} + R_{x''} - R_x < J^*(x, \mathcal{C}_x)$ **then**
            $J^*(x, \mathcal{C}_x) \leftarrow J^*(x'', \mathcal{C}_{x''}) + \frac{d_h(V_{x_b}, V_{x''_b})}{\gamma_G} + R_{x''} - R_x$
        end if
    end for
end for

2.4. Complexity Analysis

The computational complexity of computing $\forall b, b' \in \mathcal{B}$, $d_h(V_b, V_{b'})$ using Floyd-Warshall algorithm is $\mathcal{O}(M^3)$ while Johnson's algorithm can also be used for sparse graphs with a complexity of $\mathcal{O}(M^2 log(M) + M\mathcal{E})$ [45]. The construction of ordered sets $O_{\leftarrow}$ and $O_{\rightarrow}$ requires the sorting of two NM-sized arrays. The computational complexity of each is $\mathcal{O}(NMlog(NM))$. After calculating $R_x + d_s(P_{x_a+1}, P_N)/\gamma_A$ and $G_{x'} + d_s(P_1, P_{x'_a})/\gamma_A$ values for all selections with $\mathcal{O}(NM)$ complexity, $\alpha(x, x')$ for a pair of charge selections can be computed from the summation $R_x + d_s(P_{x_a+1}, P_N)/\gamma_A + G_{x'} + d_s(P_1, P_{x'_a})/\gamma_A - \eta'_A$ with $\mathcal{O}(1)$ complexity. Thus, $W_{xx'}$ and $E_{xx'}$ can also be computed with $\mathcal{O}(1)$ complexity.

In the Dockless Landing scenario, each charge selection is inserted into and removed from U_k at most once. Since U_k is a binary search tree, this operation has an average time complexity of $\mathcal{O}(log(|U|))$, where $|U|$ is the average size of the U_k and $|U| \leq NM$. Therefore, evolving U_k for all of the charge selections in $O_{\rightarrow}$ has an average computation complexity of $\mathcal{O}(NMlog(|U|))$ and a worst-case complexity of $\mathcal{O}(NM|U|)$. For each charge selection, running Algorithm 1 can have a computation complexity between $\mathcal{O}(log(|U|))$ (finding the element with minimum key value) and $\mathcal{O}(|U|)$ (completely traversing U). Therefore, running the algorithm for all charge selections can have a time complexity between $\mathcal{O}(NMlog(|U|))$ and $\mathcal{O}(NM|U|)$. Overall, with Dockless Landing, the scheduling algorithm has a time complexity between $\mathcal{O}(NMlog(NM))$ and $\mathcal{O}(NM|U|)$.

With Docked Landing, each charge selection is inserted into and removed from M different binary search trees U_a^b, at most once. Therefore, evolving U_a^b has an average time complexity of $\mathcal{O}(NM^2log(|U|))$ and a worst-case time complexity of $\mathcal{O}(NM^2|U|)$. Furthermore, running Algorithm 1 for all non-final charge selections can have a time complexity between $\mathcal{O}(NMlog(|U|))$ and $\mathcal{O}(NM|U|)$. Overall, with Docked Landing, the algorithm has a computational complexity between $\mathcal{O}(NM^2log(|U|))$ and $\mathcal{O}(NM^2|U|)$.

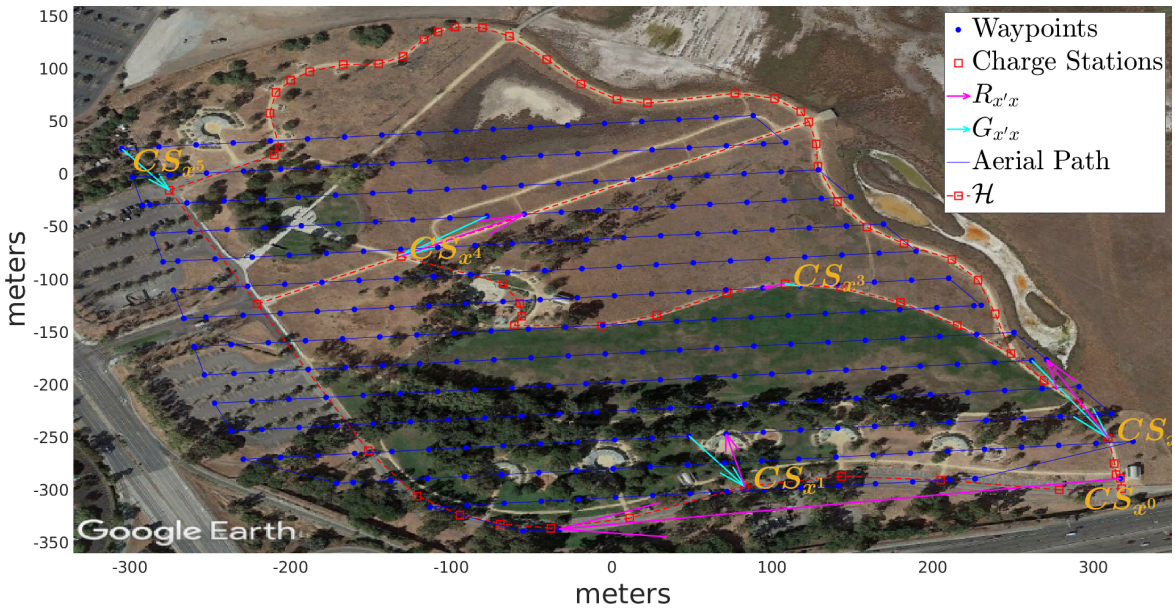
When muling is considered, Algorithm 4 is run for each $a \in A$, N times. It has a computational complexity of $\mathcal{O}(M^2)$. Thus, it brings a computational complexity of $\mathcal{O}(NM^2)$ to the Docked Landing scenario. As the Docked Landing scenario without muling has a larger computational complexity, considering muling does not increase it.

2.5. Results

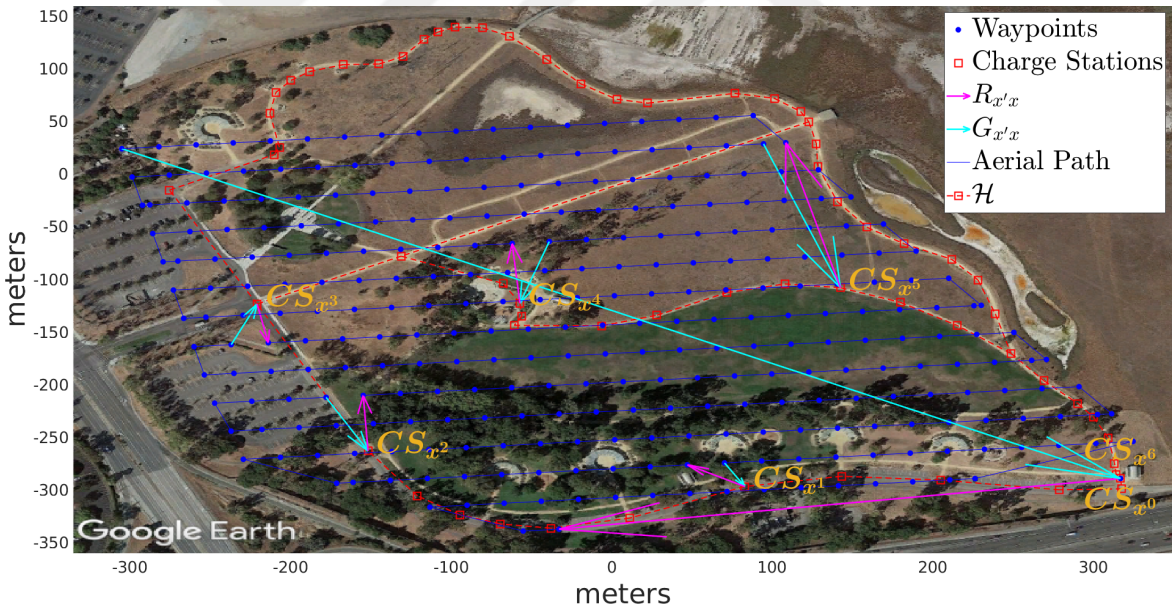
This section presents simulation results for coverage missions using real data from three different areas: Baylands Park, USA; Marmaracık National Park, Turkey; and Ceylanpınar Agricultural Enterprise, Turkey. We consider a range of scenarios varying w.r.t. number of waypoints N, number of potential charging locations M, maximum flight time η_A between consecutive recharges as well as landing, muling, and end locations. The results are shown in Table 2.1. For the first two areas, waypoints are set to ensure area coverage by Back-and-Forth patterns with parallel line segments [46]. The road networks and potential charging locations are extracted manually, considering the geography of the regions. The number of potential charging locations is increased by upsampling the road networks. D- and D+ represent dockless and docked landing scenarios, M-, M+, and M± represent no muling, muling, and either case, R- and R+ represent circulating or radiating end locations. We evaluate both performance and computational complexity. For performance study, we compute $T_C = T_{x^0} + \sum_{i=1}^{N'} T'_{x^i x^{i+1}}$ - namely total time of the charge selections. To understand the effect of charging on total mission time, we also compute $T_W = \frac{d_s(P_1,P_N)}{\gamma_A}$ - namely the waypoint flight time without any recharging. Note that $J^*(x^0, \mathcal{C}) = T_W + T_C$. To evaluate computational complexity, we consider both memory and processing time requirements. We also conduct a comparative study with a baseline approach in which both the route and charging schedule are determined simultaneously. The algorithms are implemented with C++ and the simulations are done on an Intel Core i9 13900K Central Processing Unit (CPU).

Table 2.1. Simulation results. Baylands Park, USA (BP); Marmaracık National Park, Turkey (MNP); and Ceylanpınar Agricultural Enterprise, Turkey (CAE). D- and D+ represent dockless and docked landing scenarios, M-, M+, and M± represent no muling, muling and either case, R- and R+ represent circulating or radiating end locations.

#	Scenarios & Parameters												Performance Measures				Required Resources	
	Area	Type	N	M	γ_A m/sec	γ_G m/sec	η_A min	$T_\downarrow$ min	$T_\uparrow$ min	E min	λ_A	T_W min	N'	T_C min	$J^*(x^0,\mathcal{C})$ min	$J^{**}(x^0,\mathcal{C})$ min	Computation Time sec	$\lvert U\rvert$
1	BP	D-M-R+	301	63	10	1.1	5	1	1	1	-	11.78	5	16.30	28.08	23.78	0.134	3138.2
2		D-M-R-	301	63	10	1.1	5	1	1	1	-	11.78	6	20.67	32.45	23.78	0.172	3138.2
3		D-M-R+	1204	63	10	1.1	10	1	1	1	-	22.68	3	9.69	32.37	31.68	0.086	20220.9
4		D+M-R+	1204	63	10	1.1	10	1	1	1	-	22.68	3	9.69	32.37	31.68	3.010	20172.7
5		D-M-R+	1204	127	10	1.1	10	1	1	1	-	22.68	3	9.65	32.33	31.68	0.267	41037.3
6		D-M-R+	1204	127	10	0.55	10	1	1	1	-	22.68	3	11.03	33.71	31.68	7.606	41037.3
7		D-M-R+	1204	127	10	0.55	10	1	1	-	0.5	22.68	3	21.41	44.09	43.02	7.917	41037.3
8	MNP	D+M± R+	256	21	15	2.2	20	1	1	1	-	30.85	2	7.15	38.00	36.85	M-:0.0170, M+:0.0180	2032.3
9		D+M±R+	2048	21	15	2.2	20	1	1	1	-	83.38	5	16.85	100.23	98.38	M-:0.321, M+:0.328	7303.7
10		D+M±R+	2048	42	15	2.2	20	1	1	1	-	83.38	5	16.52	99.90	98.38	M-:1.899, M+:1.943	14755.8
11		D+M±R+	2048	85	15	2.2	20	1	1	1	-	83.38	5	16.13	99.51	98.38	M-:14.18, M+:13.93	29906.8
12		D+M±R+	4096	85	15	2.2	20	1	1	1	-	117.06	7	22.26	139.32	138.06	M-:39.83, M+:39.58	43845.9
13		D-M-R+	4096	85	15	2.2	20	1	1	1	-	117.06	7	22.26	139.32	138.06	0.365	43899.0
14		D+M-R+	4096	85	15	2.2	20	1	1	-	0.5	117.06	7	81.20	198.26	196.59	37.55	43845.9
15	CAE	D+M-R+	10	39	10	11	10	1	0.5	1	-	21.99	3	10.24	32.22	29.49	0.00134	14.5
16		D+M+R+	10	39	10	11	10	1	0.5	1	-	21.99	3	9.86	31.85	-	0.00168	16.1
17		D+M-R+	10	39	5	2.2	10	1	0.5	1	-	43.97	Not possible.			58.97	0.00088	1.1
18		D+M+R+	10	39	5	2.2	10	1	0.5	1	-	43.97	8	57.80	101.78	-	0.00128	2.4



(a) D-M-R+



(b) D-M-R-

Figure 2.5. Baylands Park (BP): (a) radiating and (b) circulating scenarios solutions with N=301 and M=63.

The first area covered is an approximately 0.167 km^2 area in Baylands Park, USA (BP). Here, we consider $N \in \{301, 1204\}$. The resulting route lengths are $\{7.068, 13.609\}$ kilometers, respectively. The number of potential charging locations $M \in \{63, 207\}$. As expected, the solution for the radiating scenario has better performance

than the circulating scenario, as can be seen in simulations 1 and 2. If the final waypoint is far from the initial charging location, the circulating scenario results in a large return for the final charge selection as seen in Figure 2.5. Additionally, as the MGR has to be at initial charge selection at the mission end, it can not follow the AR through the mission. This indicates that if a circulating scenario is being considered, the importance of the ordering of the waypoints increases. In the radiating scenario, we observed that the MGR navigates in accordance with the waypoint order of the AR to reduce the time spent to reach a charging location. Increasing the scattering density of the charge stations slightly improved the mission performance in simulations 3 and 5. Dockless and docked landing scenarios in simulations 3 and 4 have the same solution. As can be seen in simulations 6 and 7, finding the solutions with constant and linear recharge functions takes nearly the same time.

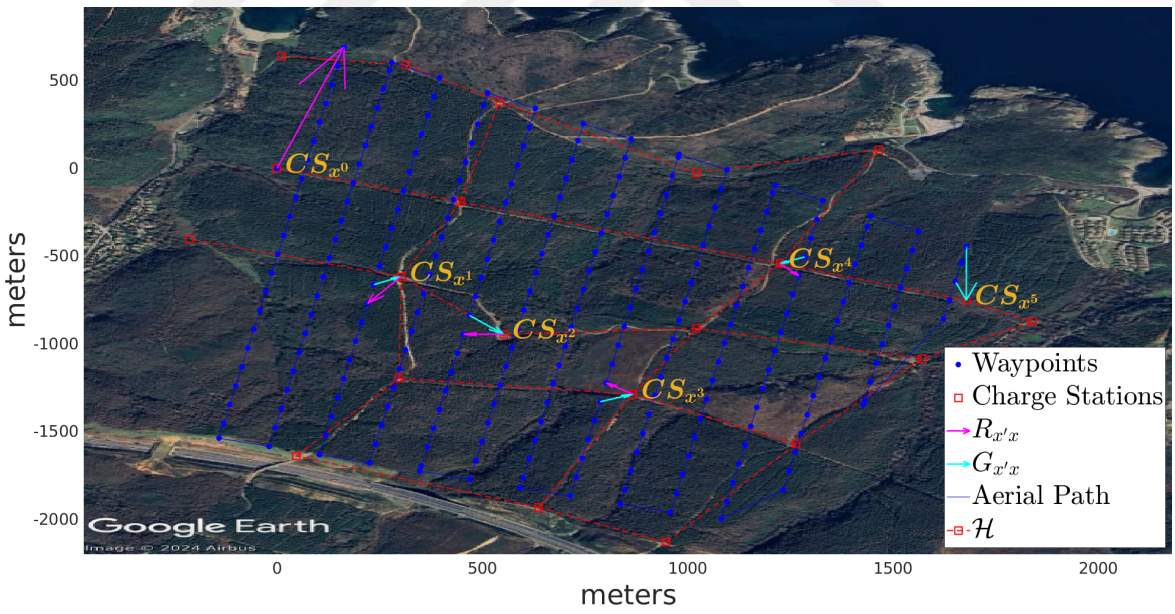


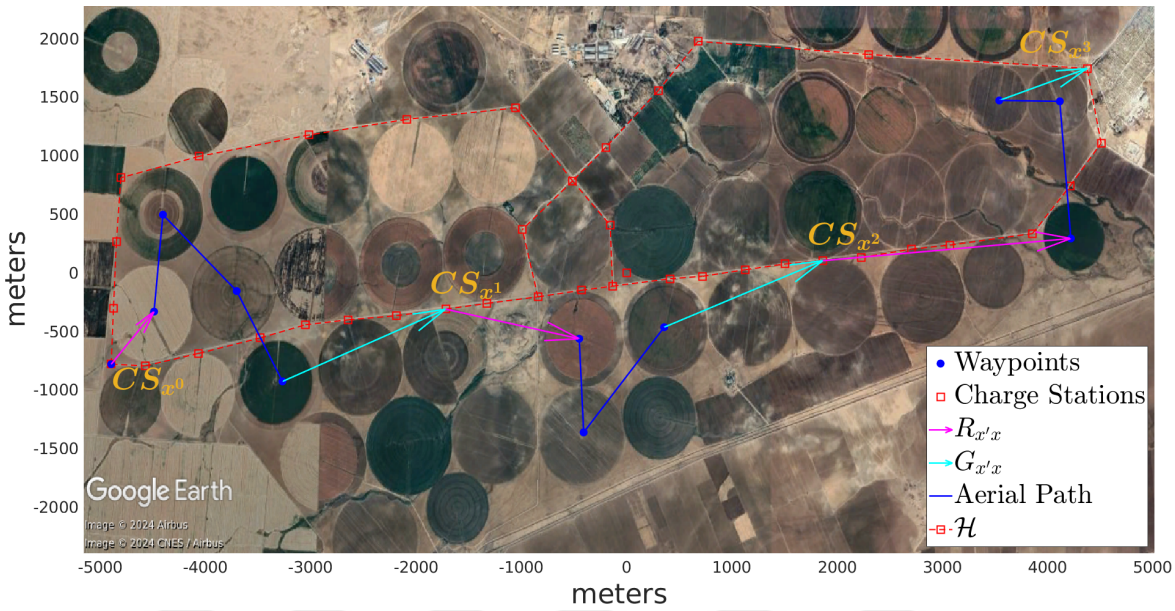
Figure 2.6. Marmaracık National Park (MNP): D-M-R+ and D+M-R+ with N=256 and M=21 have the same solutions.

The second mission result uses data from Marmaracık National Park, Turkey (MNP), covering approximately 3.07 km^2 area. Here, we consider N = {256, 2048, 4096} with the resulting route lengths {27.764, 75.046, 1053.51} kilometers, respectively. The number of potential charging locations $M \in \{21, 42, 85\}$. If the scattering

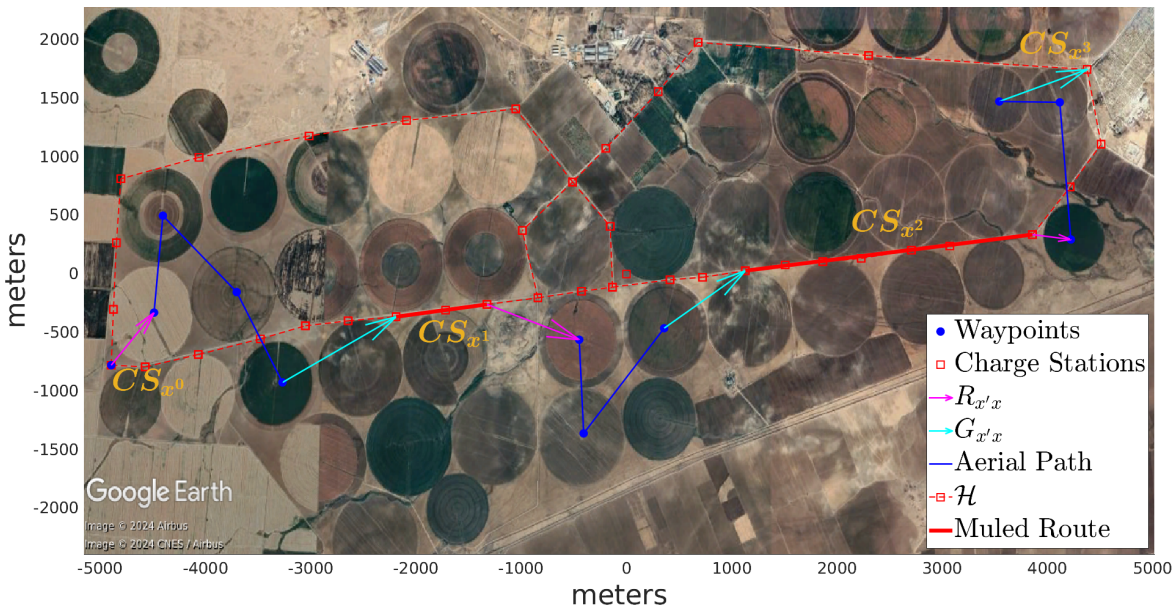
density of the charge stations over the mission area is increased, this results only in a slight reduction of the mission time $J(x^0, \mathcal{C})$ - as seen in simulations 9-11. However, this observation cannot be generalized as the optimal solution also depends on the area coverage of the charging locations. Again, solutions to docked and dockless landing scenarios are observed to be the same in simulations 12 and 13. In fact, most of the time, solutions to docked and dockless landing scenarios are observed to be the same for typical coverage missions. Figure 2.6 demonstrates such a situation. This can be attributed to the fact that if the consecutive waypoints are close to each other, rather than waiting for the MGR, the AR visits more waypoints in the meantime, causing a zero waiting time. If all the charge selections in the optimal solution have zero waiting time for the dockless landing scenario, then is also the optimal solution for the docked landing scenario since this solution violates no constraint of the docked landing scenario, and the docked landing scenario can not have a better solution than the dockless scenario as it is a more constrained version of the problem than dockless scenario. Interestingly, muling does not lead to time gains in simulations of this area. For example, simulations 8-12 turn out to be the same solution regardless of whether muling is done or not. This is attributed to the fact that if the waypoints in the area are dense compared to the potential charging locations, or if the speed of the MGR is very low compared to the speed of the AR, muling does not have much benefit as the AR can traverse the same way much faster itself. In simulations 13 and 14, it is observed that finding the solutions with constant and linear recharge functions takes nearly the same time.

The third set of experiments is done over Ceylanpınar Agricultural Enterprise, Turkey (CAE) Here, 10 waypoints are set as three distant clusters with a path length of 13.192 kilometers. Simulations 15 and 16 consider the same scenario without and with muling. In simulation 16, it is observed that both of the two intermediate charges are carried out with muling. In this case, muling is observed to be useful. This is attributed to MGR being faster than the AR along with charging locations being relatively denser as compared to waypoints so that muling enables the AR to reach the next waypoint faster, as seen in Figure 2.7. In simulations 17 and 18, it is observed

that if the distance between two consecutive waypoints exceeds the flying range of the AR, there is no alternative but to use muling.



(a) D+M-R+



(b) D+M+R+

Figure 2.7. Ceylanpınar Agricultural Enterprise (CAE): (a) muled and (b) not muled solutions with N=10 and M=39.

For the baseline comparison, we compare time performance $J^{**}(x^0, \mathcal{C})$ for the scenarios without muling assuming the route to be optimally determined. Note that this value constitutes a simple optimistic lower bound of the mission time when considering route finding and charge scheduling problems simultaneously. First, the lower bound N'^* on the number of recharges required is computed by considering route length and maximum flight times as $N'^* = \min_{N' \in \mathbb{Z}^+} N'$ s.t. $N'\eta_A \geq T_W + N'(T_\downarrow + T_\uparrow)$. Using the lower bound of the number of recharges, the lower bound of the mission time with constant recharge time is computed as $J^{**}(x^0, \mathcal{C}) = N'^*\big(T_\downarrow + T_\uparrow + E\big) + T_W$. With a linear recharge function, the computation of $J^{**}(x^0, \mathcal{C})$ is also similar. When the total time spent for going to, waiting at, and returning from charges during a mission is low compared to the total mission time, the value $J^*(x^0, \mathcal{C})$ is observed to be close to $J^{**}(x^0, \mathcal{C})$. This ratio is observed to be low in typical coverage tasks in which the potential charging locations envelop the area to be monitored. In fact, in the scenarios considered, we observe the ratio $J^*(x^0, \mathcal{C})/J^{**}(x^0, \mathcal{C})$ to be between 1.009 and 1.365 with 1.060 on average. This indicates that pending the route sequence corresponding to minimum $d_s(P_1, P_N)$ is known, the solution $J(x^0, \mathcal{C})$ of the proposed approach is close to the optimal solution of the problem in which the route order and rendezvous schedule are optimized simultaneously.

The computation time for the rendezvous schedule varies between 0.00128 seconds to 39.58 seconds. Finding the solution for the dockless landing scenario takes significantly less time than that of the docked landing scenario, as seen in simulations 3, 4 and 12, 13. This is expected as the docked landing scenario has higher computational complexity and requires evolving M binary search trees. Considering muling did not have a significant effect on the computation times. The computation times are observed to depend on N, M, average size of the evolving binary trees $|U|$. Along with these, we observed an indirect dependency of computation time on average speed γ_G of the MGR and maximum flight time η_A of the AR. As γ_G or η_A decreases, there is a higher possibility that waiting time $W_{x'x} \neq 0$ for a given charge selection. Since the elements of the evolving binary trees are sorted using a tentative waiting time $\hat{W}_x = 0$, the number of visited nodes in Algorithm 1 increases with decreased quality of this

tentative value. This is observed across simulations 2, 3 and 5, 6. In the former case, while N is quadrupled, computation time decreases from 0.172 sec to 0.086 sec since η_A is doubled. In the latter case, a decrease in γ_G leads to an increased computation time. Computation times are observed to be feasible - even with an average tree size of 43899 nodes. This suggests that finding the best successor of a charge selection requires the traversal of only a small part of this tree.

2.6. Discussion

The work in this chapter makes some assumptions as discussed in Section 2.2. In real-robot applications, some of these assumptions may become invalid. For example, wind conditions in an area may cause an increase in the power consumption of the AR and thus result in a lower maximum flight time. To prevent the issues that may arise from that, a safety margin for the maximum flight time of the AR should be considered, with its level depending on the expected deviations on the maximum flight time of the AR. Another possible situation is ground obstacles preventing the MGR from preserving its average speed, or even using some of the roads associated with the $\mathrm{H}(\mathcal{B}, \mathcal{E})$ due to a blockage. The high speed of the presented method allows for finding a new schedule quickly during an ongoing mission, considering the blockages in the $\mathrm{H}(\mathcal{B}, \mathcal{E})$ or reduced average speed of the MGR to recover the mission. While a delay of the MGR in the Dockless Land scenario may result in an increase in the waiting time, $W_{x'x}$, ultimately increasing the mission time, a delay of the MGR in the Docked Land scenario may cause AR to exceed its maximum flight time while waiting for the MGR, and possibly a mission failure. Depending on the expected level of deviations in the parameters used to find the rendezvous schedule and type of landing, additional safety measures can be considered.

3. MAPPING AND LOCALIZATION

This chapter is focused on the usage of sensory data collected by the AR. In particular, two different usages are considered:

- 3D map generation by an AR based on RGB camera and LIDAR data.
- RGB data-based localization of an AR

The outline of the chapter is as follows: Related literature is discussed in Section 3.1. The proposed 3D mapping method is explained in Section 3.2. Two different visual-based localization methods are presented in Section 3.3. Section 3.4 presents the performance evaluations of proposed 3D mapping and visual localization methods in the simulation environment.

3.1. Related Work

Related work spans two primary areas in robotics: i) 3D Mapping by an AR; ii) RGB-based localization of an AR. This section presents a brief summary of work done in each area.

3.1.1. 3D Mapping

Terrain mapping has been extensively studied in the literature. As the AR flies over a region, it collects sensory data and then uses this data in order to construct the terrain map. The proposed approaches vary with respect to the sensors that are used and how the map is constructed.

Various sensors have been utilized. For example, time-of-flight depth cameras can sense the depth of each pixel by calculating the time of flight of received infrared light transmitted by the sensor itself. The resulting 3D point cloud corresponds to an RGB-

D representation of the environment. However, while operating outdoors, calculating the time-of-flight of the transmitted light by the sensor becomes unreliable due to the increased amount of infrared light from the sun. Alternatively, stereo cameras can be used to create the RGB-D representation of the scene. As the depth range that can be sensed by such cameras is limited by the physical separation and resolution of the sensors, which makes using them with ARs impractical in general scenarios. The 3D structure of the environment can also be created with RGB camera views from different locations using triangulation. However, the scale of the structure created by this technique can not be known. There are many works on constructing the 3D representation of the environment with scale using RGB cameras along with other sensors, such as inertial measurement units (IMU). This way, both the structure and the scale of the environment can be constructed. However, even the state-of-the-art techniques have a scale error of around 1% [47]. This scale error corresponds to large positional inaccuracies while operating in a large environment with an AR. Another approach is to use RGB cameras along with LIDARs.

The construction of the map is typically done by identifying keypoints in each sensory data frame and then relating keypoints from nearby frames. As scene images may be collected from different heights, the method chosen for keypoint extraction should be scale invariant as much as possible so that keypoints in these images to be relatable. Additionally, rotational invariance is also important since the AR may be in various orientations during flight when collecting data. With RGB data, keypoint extraction methods that satisfy these criteria to some extent are Scale Invariant Feature Transform (SIFT) [48], Speeded-up robust features (SURF) [49], and Orientated Fast and Robust Brief (ORB) [50].

Following, the keypoints from different frames are related with each other in order to stitch or mosaic the two data together as to generate a larger map. As is well known, stitching is a fundamental machine vision problem. The basic approach is to create a panoramic image via warping new images onto it one at a time and to align each new image with the images that have already been warped [51, 52]. With multi-

view geometry systems, the respective methods are also known as bundle adjustment methods [53]. If the motion between two frames includes translation, then the relationship between two images is not necessarily a homography. In this case, images are warped to a base view as to minimize misalignment while aiming to keep computational complexity minimal as well [54, 55].

3.1.2. Localization

Position estimation of ARs is typically done using GNSS modules and IMUs. Using a standard GNSS module with a clear view of the sky, the accuracy of the resulting position is around 2 meters. However, GNSS modules rely on signals from satellites, which are prone to jamming and easily get obstructed by physical obstacles such as buildings and trees [56].

An alternative is visual localization using previously taken RGB images of the flight area. In particular, aerial RGB images can be tagged with highly accurate (around 1 cm accuracy) location information using sophisticated positioning systems based on Radio Detection And Ranging (RADAR) or LIDAR sensors or Real-time kinematic (RTK) positioning systems [57]. These images can later be used for positioning ARs with RGB cameras flying in the same area without the use of a sophisticated positioning system. Here, interpolation and geometry-based methods are investigated to achieve this.

In geometry-based methods, a local 3D representation of the environment is constructed from common keypoints of two or more images with known positions. The resulting 3D representation is then used to estimate the position of the current image that views the same area. In order to construct the 3D representation, dataset images are required to have some percentage of overlap. Hence, unlike the interpolation-based estimation, this method can not work with dataset images that have no overlap. Each keypoint in the camera frame is associated with a position in the 3D world that lies on the ray passing through the camera origin and the respective image point. Ideally,

the rays of the matched keypoints of two images should intersect at a point in the 3D world that also corresponds to the actual 3D position of this keypoint. However, due to noisy sensor measurements, limited resolution of the cameras, and erroneous keypoint matches, these rays do not necessarily intersect. Two main approaches are used to estimate the 3D positions of the keypoints with non-intersecting rays: Finding the position with minimum mean squared distance to both rays as to minimize the 3D error, or finding the position that is associated with a projected pixel position that minimizes the mean squared distance to keypoint pixel positions in the images.

3.2. 3D Mapping

The proposed approach consists of the following stages:

(i) RGB-based keypoint extraction in order to relate different frames with each other,
(ii) Depth estimation of keypoints with LIDAR data interpolation,
(iii) Generation of 3D representation of the scene in the sensor coordinate frame,
(iv) Sensor frame to map frame transformation estimation based on keypoint matching,
(v) Registration of new RGB-D representation to the map based on estimated transformation.

In the first stage, keypoints are determined in each incoming RGB frame. Here, SIFT features are used. Note that SURF or ORB features may also be used to reduce the computational load at the cost of reduced accuracy. Next, depth information (i.e., the distance between the corresponding point in the actual world and the camera sensor) is obtained from LIDAR sensor data. Here, LIDAR and the camera are assumed to be at the same position and orientation to ease the data relation from each sensor. In reality, there must be a positional displacement between the two sensors. However, when the scene is far from the sensors, as in the case with ARs, the effect of positional displacement is neglectable. Also, as the sizes of their field of view (FOV) differ, the sensor with the smaller FOV is used to determine a common FOV region. Only the

data within the common FOV region is used. This way, a common image plane where data from two sensors can be represented together is created. Finally, as the resolution of a typical RGB camera is higher than that of a typical LIDAR, pixel points that are associated with a keypoint may not be associated with depth data. This is addressed via interpolating depth data at these points using bilinear interpolation. Consider a keypoint with horizontal and vertical pixel coordinates (p_h, p_v). The interpolated depth $d(p_h, p_v)$ is then defined as

$$d(p_h, p_v) = \frac{1}{(p_{h2}-p_{h1})(p_{v2}-p_{v1})} * \begin{bmatrix} p_{h2}-p_h & p_h-p_{h1} \end{bmatrix} \begin{bmatrix} d(Q_{11}) & d(Q_{12}) \\ d(Q_{21}) & d(Q_{22}) \end{bmatrix} \begin{bmatrix} p_{v2}-p_v \\ p_v-p_{v1} \end{bmatrix} \quad (3.1)$$

where Q_{ij}, $i, j = 1, 2$ represent the horizontal and vertical pixel coordinates closest to (p_h, p_v) with depth data $d(Q_{ij})$ available. The coordinates of these pixels in the image plane will be as seen in Figure 3.1. Let their coordinates be denoted by (p_{h1}, p_{v1}), (p_{h1}, p_{v2}), (p_{h2}, p_{v1}), and (p_{h2}, p_{v2}) respectively.

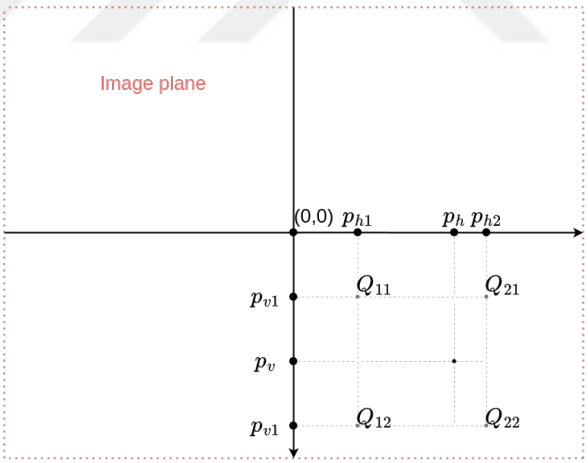


Figure 3.1. Closest pixels used in bilinear interpolation.

Following, the camera matrix is used to find the 3D position $P = [p_x p_y p_z]^T$ of a keypoint in the camera-centered coordinate frame. This matrix relates the pixels in

the image plane and camera-centered coordinate frame as given in Equation 3.2 [58],

$$s * \begin{bmatrix} p_h \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}. \tag{3.2}$$

Here, f is the focal length of the camera, and s is a positive scalar that corresponds to p_z, which is not measured by any of the sensors. Noting the following re-arrangement,

$$\begin{bmatrix} s * p_h \\ s * p_v \\ s \end{bmatrix} = \begin{bmatrix} p_x * f \\ p_y * f \\ p_z \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} s * p_h/f \\ s * p_v/f \\ s \end{bmatrix} \tag{3.3}$$

and

$$p_x^2 + p_y^2 + p_z^2 = (d(p_h, p_v))^2. \tag{3.4}$$

Equivalently,

$$s^2(\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1) = (d(p_h, p_v))^2. \tag{3.5}$$

Since s must be positive,

$$s = d(p_h, p_v) * \sqrt{\frac{1}{\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1}}. \tag{3.6}$$

Substituting s into Equation 3.3,

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d(p_h, p_v) * \sqrt{\frac{1}{\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1}} * p_h/f \\ d(p_h, p_v) * \sqrt{\frac{1}{\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1}} * p_v/f \\ d(p_h, p_v) * \sqrt{\frac{1}{\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1}} \end{bmatrix}. \tag{3.7}$$

Writing in the homogeneous coordinates,

$$P_{k_i} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = d(p_h, p_v) * \sqrt{\frac{1}{\frac{p_h^2}{f^2} + \frac{p_v^2}{f^2} + 1}} * \begin{bmatrix} p_h/f \\ p_v/f \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

where P_{k_i} is the 4×1 homogeneous form of the 3D position of the given pixel positions in the image plane. It should be noted that this relation is valid when the camera operates in compliance with the pinhole model. To ensure this, the camera should be

calibrated, and the image should be rectified accordingly. Using Equation 3.8, locations of all keypoints in a frame are transformed into the camera-centered coordinate frame and stored in a matrix P_k as column vectors.

The next stage is to transfer data from individual frames into a map coordinate frame to generate a common map representation. This transformation can be done as

$$P_{m_k} = A_k^m P_k. \tag{3.9}$$

Here, A_k^m is the 4×4 rigid transformation matrix and P_{m_k} is the representation of P_k in the map coordinate frame as column vectors. The initial frame is assumed to be at the origin to create a common map, i.e., the transformation matrix for the initial frame is identity. Keypoints in the initial frame are transformed into the map frame to create the map P_m. Consider the descriptors of the keypoints in the map P_m and keypoints of the new frame P_k. Keypoint descriptor-based matching is done as follows:

- Descriptors of P_m and P_k are matched using Fast Library for Approximate Nearest Neighbors (FLANN) [59].
- With Lowe's ratio test [48], bad descriptor matches are filtered out.
- The remaining n good matches are used to construct two 4×n dimensional matrices P_{m_r} and P_{k_r} corresponding to the map keypoints and frame keypoints respectively.

The matched keypoints are then used to estimate the the transformation matrix A_k^m. Here, two alternative methods are considered.

The first method is an optimization method. In practical scenarios, due to noisy sensor data, interpolated depth, and matching errors, there is no linear transformation that maps P_{k_r} into P_{m_r} as defined by Equation 3.9. This is addressed through finding the transformation matrix that minimizes the least squares error $||A_k^m P_{k_r} - P_{m_r}||^2$ between transformed points and their matched counterparts in the map frame as

$$A_k^m P_{k_r} = P_{m_r} \tag{3.10}$$

$$A_k^m P_{k_r} P_{k_r}^T = P_{m_r} P_{k_r}^T \tag{3.11}$$

$$A_k^m = P_{m_r} P_{k_r}^T (P_{k_r} P_{k_r}^T)^{-1}. \tag{3.12}$$

In Equation 3.12, $(P_{k_r} P_{k_r}^T)^{-1}$ is assumed to be invertable. This is possible when the $rank((P_{k_r} P_{k_r}^T)^{-1}) = 4$, meaning there are at least four matched keypoints that are non-planar. In practical scenarios, this condition is satisfied when the scene has enough details, e.g., not a featureless area, and there is enough overlap between previously seen areas on the map and the new frame.

The alternative method to estimate the A_k^m is to use Random Sample Consensus (RANSAC) [60]. This method has the capability of filtering out geometrically inconsistent matches and provides another level of filtering in addition to Lowe's ratio test. Thus, robustness against incorrect keypoint matches increases. An implementation from Point Cloud Library (PCL) is used to estimate the transformation matrix from P_{k_r} to P_{m_r} with RANSAC [61].

The final stage is to use the computed A_k^m to update the map via transforming P_k. Thus, the map can be enlarged with previously unseen keypoints. However, multiple occurrences of the same keypoints from different frames cause issues during later matches: Lowe's ratio test checks for the quality of a keypoint match by comparing the distances of the descriptor of a keypoint to the nearest and the second nearest descriptors stored in the map. If the keypoints from different frames corresponding to the same location in the world are stored on the map together, then they will cause very similar distances in the Lowe's test. Thus, only the current frame keypoints that are not matched with the keypoints in the map are added to the map.

3.3. Visual Localization

Two alternative methods are considered for visual localization. Both are based on geo-tagged reference dataset images retained in memory:

- Interpolation-based method: The position is inferred based on similarities be-

tween the image associated with the current position and reference images.

- Geometry-based method: The position is inferred from the geometric relations between the keypoints of the current image and those of the reference images.

3.3.1. Interpolation-Based Localization

If there is an available set of images tagged with position information, the robot can visually localize itself by relating the current RBG image to those images and then inferring its position via interpolating accordingly. As the AR gets closer to the position of a previously taken image, the similarity of the current RGB image will increase accordingly. Thus, the similarity between two images can be used as an estimate of the distance between their respective positions. These estimated distances can be used with interpolation methods to calculate the position of the current RGB frame. For this, similarity is defined based on the visual matching of keypoints. Five different metrics are considered:

(i) Inverse Match (IM): $\frac{1}{n}$
(ii) Inverse Match Square (IMS): $\frac{1}{n^2}$
(iii) Inverse Match Square Root (IMSR): $\frac{1}{n^{1/2}}$
(iv) Average Descriptor Distance Inverse Match (ADDIM): $\frac{f_s}{n}$
(v) Average Descriptor Distance Inverse Match Square Root (ADDIMSR): $\frac{f_s}{n^{1/2}}$

where $|n|$ represents the number of good SIFT descriptor matches between the current image and the reference image, and f_S represents the average Euclidean distance between the matched descriptors of two images in 128-dimensional SIFT descriptor space. The idea behind metrics (i), (ii), and (iii) are similar: As the number of matched keypoints between two images increases, their distance should decrease. Powers of this number are used to see the amount of dependency. Along with the number of matches, metrics (iv) and (v) additionally use a quality measure for these matches: if these are good matches, their descriptors should also be close in SIFT descriptor space. A larger average distance is likely to indicate a larger distance between the two image positions.

In order to estimate the position using these similarity metrics, the Spherical k-Nearest Neighbors Interpolation algorithm (SkNNI) is used [62]. The interpolation value at a location is based on the weighted distances to the nearest data points. Here, the 3D position is considered as the value to be interpolated, and distances are estimated using the similarity metrics defined above. It should be remarked that the SkNNI does not require grid-based data points. This is useful as it enables the positioning algorithm to work with irregular data points. Additionally, it uses distances between the data points and the current RGB image point and does not require the position of the current RGB image point during interpolation.

3.3.2. Geometry-Based Localization

In geometry-based localization, the AR relates the input RGB frame to geotagged frames that are retained in its memory. The robot determines the pair of adjacent frames that are most similar appearance-wise to the current image. It is assumed the positions of these frames are also closest to the unknown position. Following, keypoints from one of the reference frames are matched to those of the other reference frame. Note that each matched pair corresponds to one (albeit unknown) world location. This location can be computed by forming two rays with each passing through one of the reference frame origins and the corresponding keypoint's image location (using Equation 3.2) and finding the 3D position that minimizes the mean squared distance to both rays [63]. Using the 3D positions of these keypoints, a local 3D representation is created.

This 3D representation is then used in localization - given the current RGB frame. Two alternative methods have been proposed for this:

- Finding the transformation matrix that minimizes 3D Error
- Finding the transformation matrix that minimizes 2D Error

3.3.2.1. Minimizing 3D Error. This method consists of two steps. First, the keypoints of the input frame are matched with those of the 3D representation. Next, the matching keypoints are used to estimate the position of the AR. Let P_m denote the homogeneous coordinates of the keypoints in the local 3D representation and P_k denote the homogeneous coordinates of the keypoints of the input frame. Suppose A_k^m denotes the transformation matrix between the respective coordinate frames as

$$A_k^m P_k = P_m. \tag{3.13}$$

P_k can be equivalently expressed as

$$P_k = K_k D_k + \perp \tag{3.14}$$

where K_k is the matrix in the form of 4×n with columns K_{k_i} defined as

$$K_{k_i} = \begin{bmatrix} p_{h_i}/f \\ p_{v_i}/f \\ 1 \\ 0 \end{bmatrix}. \tag{3.15}$$

The matrix D_k is a n×n diagonal matrix with entries $D_{k_{i,i}}$ defined based on the depth of the corresponding keypoint as

$$D_{k_{i,i}} = d(p_{h_i}, p_{v_i}) \sqrt{\frac{1}{\frac{p_{h_i}^2}{f^2} + \frac{p_{v_i}^2}{f^2} + 1}}. \tag{3.16}$$

Finally, the matrix $\perp$ is a 4×n constant matrix defined as

$$\begin{bmatrix} 0_{3\times n} \\ \mathbb{1}_{1\times n} \end{bmatrix}. \tag{3.17}$$

Hence, the relation between the homogeneous coordinates of keypoints in the input frame and the reference frames can be equivalently expressed as

$$A_k^m (K_k D_k + \perp) = P_m \tag{3.18}$$

$$A_k^m (K_k D_k + \perp)(K_k D_k + \perp)^T = P_m (K_k D_k + \perp)^T \tag{3.19}$$

$$A_k^m = P_m (K_k D_k + \perp)^T ((K_k D_k + \perp)(K_k D_k + \perp)^T)^{-1}. \tag{3.20}$$

While both A_k^m and D_k are unknown in Equation 3.20, A_k^m is known to be a rigid transformation matrix, and the D_k matrix must be a positive definite diagonal

matrix. Here, $((K_k D_k + \perp)(K_k D_k + \perp)^T)$ is assumed to be invertible, which is the case when $rank((K_k D_k + \perp)) = 4$. This condition is satisfied when there are at least four non-planar keypoints in P_k.

In practice, it may not be possible to compute these two matrices exactly due to incorrect keypoint matches and the limited resolution of the sensors. As such, they need to be determined through iteratively solving an optimization problem,

$$min_{D_{k_{i,i}}} \sum_{i=1}^{n} ||A_k^m (K_{k_i} D_{k_{i,i}} + \perp_i) - P_{m_i}||^2 \tag{3.21}$$

with the update rule based on the gradient descent method as

$$D_{k_{i,i}}(t+1) = D_{k_{i,i}}(t) - \eta \Bigg(K_{k_i}^T A_k^m(t) \Big(A_k^m(t)(K_{k_i} D_{k_{i,i}}(t) + \perp_i) - P_{m_i} \Big) \Bigg) \tag{3.22}$$

where η is the step size of the update rule. An initial depth matrix $D_k(0)$ is constructed. For this, the diagonal entries are randomly set to be around the sample image height. This enables the corresponding $A_k^m(0)$ to be determined. Since only diagonal entries of the depth matrix are updated, diagonality is preserved. Using Equation 3.20 to update A_k^m may result in A_k^m losing its rigidity and preventing the depth matrix from converging its actual value. Rather, $A_k^m(t+1)$ is determined by finding the rigid transformation from $P_k(t+1)$ to P_m. Then, the update continues until a predetermined iteration number is reached.

3.3.2.2. Minimizing 2D Error. In this method, the problem is solved through finding the transformation matrix that minimizes the distances between pixel positions of the projections of known 3D points with their associated positions in the input image. Let P_m be the homogeneous 3D coordinates of the known points in the map, H_k be the camera matrix for the input image and U_k be the homogeneous pixel coordinates of the matched points in the input image. Then, the relation between known 3D points and their projections to the sample image can be written as

$$U_k = H_k P_m. \tag{3.23}$$

H_k can be decomposed into intrinsic and extrinsic camera matrices as

$$H_k = I_k E_k. \tag{3.24}$$

Recall that intrinsic and extrinsic matrices can be written as

$$I_k = \begin{bmatrix} F_{3x3} & 0_{3x3} \end{bmatrix}, E_k = \begin{bmatrix} R_{3x3} & T_{3x3} \\ 0_{3x3} & 1_{3x3} \end{bmatrix} \quad \text{where } F_{3x3} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

Thus,

$$H_k = \begin{bmatrix} F_{3x3}R_{3x3} & F_{3x3}T_{3x1} \end{bmatrix}. \tag{3.26}$$

Here, R_{3x3} is the rotation matrix of the sample image frame, and T_{3x1} is the translation matrix of the sample image frame. Note that $rank(F_{3x3})$ is 3, and it is invertible. The intrinsic camera matrix of a calibrated camera is known. Hence, the matrices R_{3x3} and T_{3x1} can be reconstructed from H_k. Then, A_k^m can be computed as

$$A_k^m = \begin{bmatrix} R^{-1} & -R^{-1}T \\ 0_{1\times3} & 1 \end{bmatrix}. \tag{3.27}$$

Again, there is no linear transformation H_k that maps the keypoints in 3D representation, P_m, to their matched counterparts in the image frame, U_k. The estimation of H_k using U_k and P_m is considered as the Perspective-n-Point (PnP) problem [64]. Various solutions have been proposed:

- Iterative PnP: The solution is based on minimizing $\sum_{i=1}^{n} ||U_{k_i} - H_{k_i}P_{m_i}||^2$ considering the constraints on R_{3x3} based on Levenberg-Marquardt method [65].
- Sequential Quadratic PnP (SQPnp): This method guarantees the optimal solution to the problem and offers a relatively fast approach [66].
- PnP with RANSAC: This method applies the RANSAC method to the problem and is able to filter out incorrect matches.

3.4. Experiments and Results

The performance of the proposed approaches has been tested in a Gazebo simulation environment. A 3DR Iris Quadcopter is used as the AR as shown in Figure 3.2. It is equipped with a Velodyne VLP-16 LIDAR and an RGB camera with technical specifications as given in Table 3.1. Both sensors are placed to have a downward FOV. In order for RGB data-based keypoints to be meaningful, the simulation environment

should be as photorealistic as possible. This implies that there should not be synthetically created objects that are copied over the simulation environment, such as trees or artificial grass, as these will cause detecting keypoints with very similar descriptors at different places. This is ensured using a 3D model of Northumberlandia Park that has been created using aerial imagery [67]. This model is transferred into the Gazebo simulator after rotating, scaling, and translating the model according to the coordinate axes of the simulator. The resulting environment can be seen in Figure 3.3.



Figure 3.2. AR with the downward field of view sensors, flying in the simulation environment.

Table 3.1. Specifications of sensors.

Sensor	Horizontal res. (Pixels)	Vertical res. (Pixels)	Horizontal FOV (Radians)	Vertical FOV (Radians)	Measurement range (Meters)
RGB camera	1280	720	0.889	0.524	∞
VLP-16 LIDAR	239	16	0.889	0.524	100

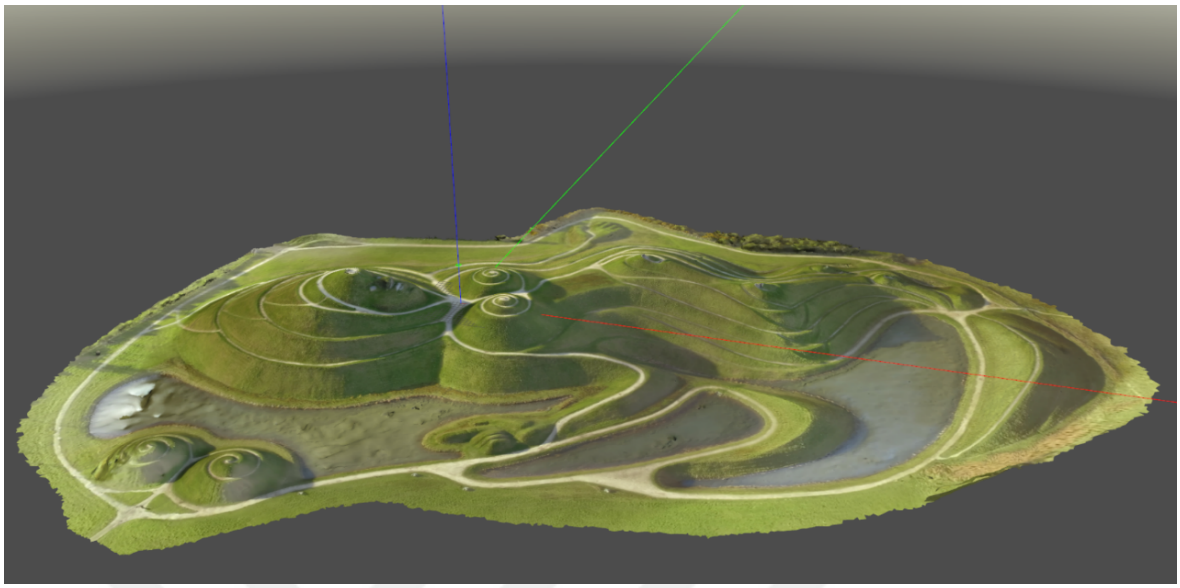


Figure 3.3. Transferred model in simulation environment.

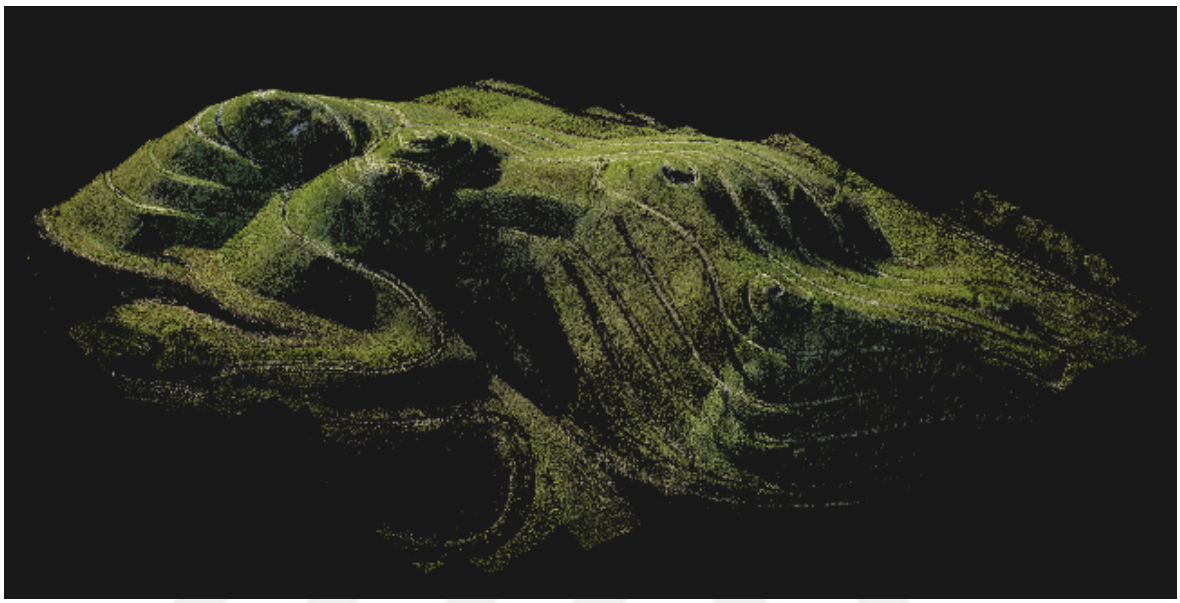
3.4.1. Mapping Experiments

The AR collects data over this terrain by flying over it using Back-and-Forth patterns with parallel line segments [46]. The data points and line interval are determined considering two factors:

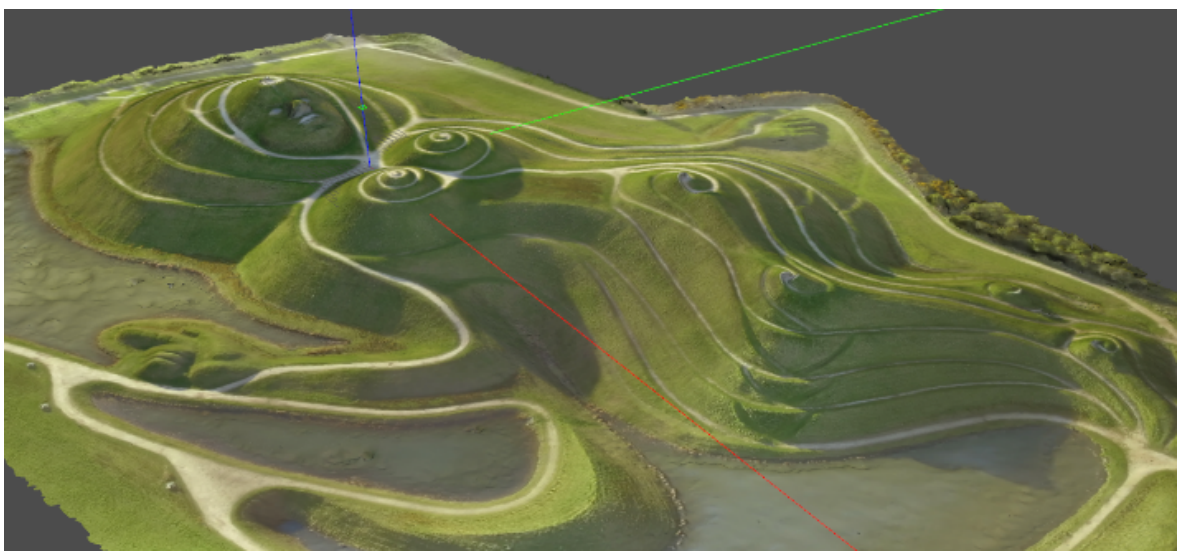
(i) Overlap ratio between consecutive images: The overlap ratio is the ratio of the area that is seen in both two consecutive images to the total area seen in an image. The area seen in an image is calculated considering the FOV of the sensors and the flight height of the AR. The data from the flight patterns with 50% overlaps are considered sparse, while from those with 75% overlaps are considered as dense. Sparse datasets consist of 52 images, while dense datasets consist of 184 images.

(ii) Loop length of the flight: Two alternative loop lengths are considered: Long and short. In a long-loop, the direction of parallel lines in the flight pattern are selected as to maximize the length of these lines. In a short-loop, the direction is selected as to minimize the length of the lines.

Four datasets; short sparse loop, long sparse loop, short dense loop, and long dense loop are created with these patterns.



(a) Resulting map.



(b) Simulation view.

Figure 3.4. (a) Resulting map from the short dense loop dataset with RANSAC and (b) simulation view from similar perspectives.

In the experiments, the position of the first image is assumed to be at the origin. Thus, the transformation matrix for the first image is taken to be the identity matrix.

The following images are added to the map in order according to the flight pattern. In Figure 3.4, the resulting map from the short dense loop dataset and the simulator view of the same area are shown. Here, the RANSAC method is used in Step 4 of the proposed method to estimate the transformation matrix.

The actual positions of the sensors in the simulator are used as a reference to evaluate the method's performance. The positional displacements in transformation matrices for each frame added to the map are compared with their actual positions. As the origin of the created maps, which is the position of the first image in the map, is not the same as the origin of the simulator, the rigid transformation to minimize the least mean squared error is calculated for each map using a Singular Value Decomposition (SVD)-based transformation estimator from PCL [68]. Maps are transformed into the coordinate frame of the simulator using estimated transformation matrices before evaluations.

The evolution of mean positional errors for sparse datasets is shown in Figure 3.5. Transformation estimations are based either on least squares or RANSAC. It is observed that the positional error for the least squares-based method in the long sparse loop dataset diverges and is thus not shown in the figure. The second worst performance is observed with the short sparse loop dataset, again with the least squares-based method. The RANSAC-based method performs better than the least squares method in both scenarios. This is expected as RANSAC-based transformation estimation is able to filter out some of the incorrect matches. Positional errors with the long-loop datasets are observed to be worse than the short-loop ones. When an image with some overlap with a previously viewed area is gathered, a low-quality transformation matrix may occur due to accumulated errors in the transformation matrices between consecutive views of the area. As the consecutive views of the same area prolong, the accumulated error also increases. This explains the relatively poorer performance of both methods with the long-loop datasets in comparison to those resulting from the short-loop datasets.

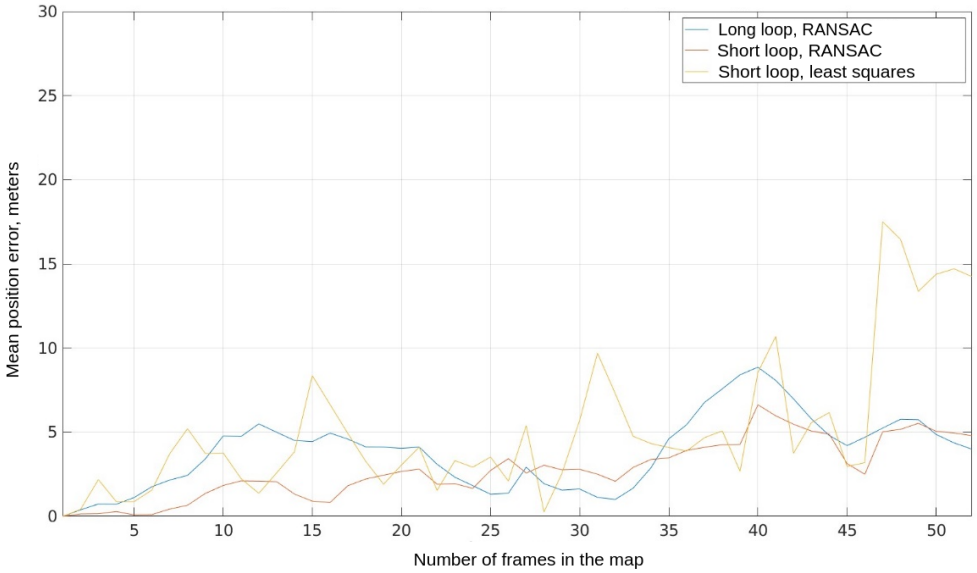


Figure 3.5. Sparse datasets: Mean positional errors vs frame index.

The performance of the proposed methods with the dense datasets is seen in Figure 3.6. Here, the performance of the least squares method with the long-loop dataset diverges as in the sparse datasets and is not shown in the figure. As in the sparse datasets, the RANSAC performs better than the least squares. Both methods perform better in general with short-loops in dense datasets. Interestingly, the performances of both methods are worse with the dense datasets as compared to the sparse datasets at dataset completion. This suggests that, even though both dense and sparse datasets cover the same area, the proposed methods are not able to use additional data from the increased density in the datasets to reduce positional error. It is observed that the current performance of the proposed methods needs to be improved considerably if the map quality needs to be high. One way to improve the mapping performance may be to update the maps through fine-tuning the interpolated depth from LIDAR considering the later images.

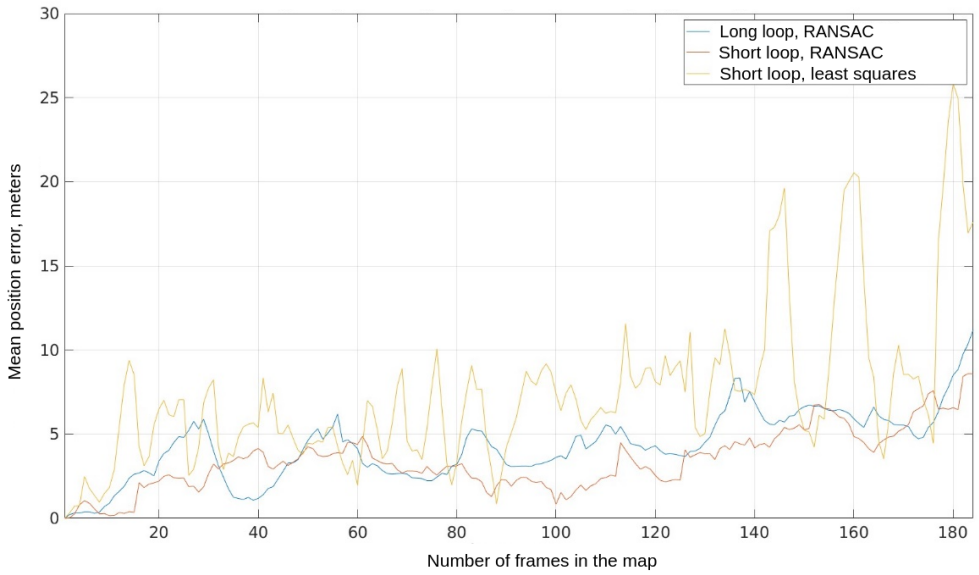


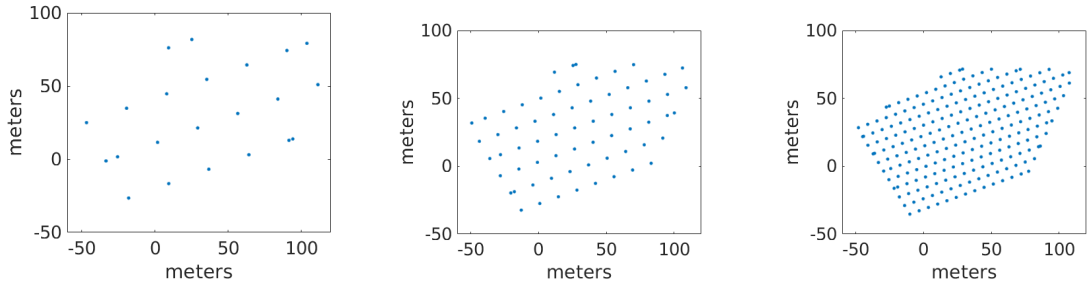
Figure 3.6. Dense datasets: Mean positional errors vs frame index.

3.4.2. Localization Experiments

Localization experiments are also done in the Gazebo simulation environment with Northumberlandia World by creating reference datasets. The datasets are generated based on generating a 2D polygon that encloses the area of interest in the simulation environment. Image positions in this polygon are generated at 40m height, using two alternative methods: Back-and-Forth patterns and uniformly random points inside the polygon.

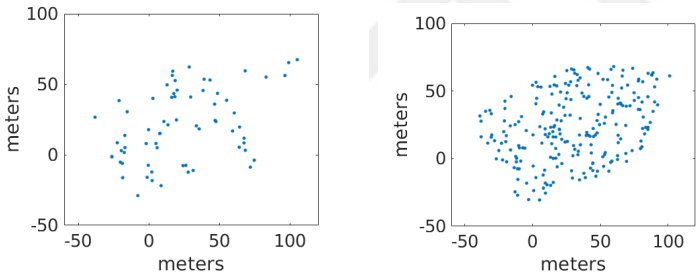
In the Back-and-Forth pattern approach, image point locations are determined considering the overlap of consecutive images. In particular, three different overlap percentages are considered: 0%, 50%, and 75%, with corresponding datasets being L_1, L_2, and L_3, respectively. Note that this method yields data points on a regular grid. The respective datasets consist of 22, 67, and 229 images. Image point locations of these datasets can be seen in Figure 3.7. With the second approach, two datasets - sparse and dense, named L_4 and L_5, respectively - are generated with 67 and 229 points, considering the cardinality of the datasets resulting in the first approach. This method results in respective image positions being on an irregular grid as seen in Figure 3.8.

One hundred sample image points to be used in the tests are generated randomly inside the polygon. Images are taken at these positions with the camera looking downward.



(a) L_1 (b) L_2 (c) L_3

Figure 3.7. Dataset image locations for Back-and-Forth patterns: (a) L_1, (b) L_2, and (c) L_3 datasets.



(a) L_4 (b) L_5

Figure 3.8. Dataset image locations for uniform random distribution: (a) L_4 and (b) L_5 datasets.

3.4.2.1. Interpolation-Based Localization Experiments. Positional errors of interpolated sample images are shown in Table 3.2. Here, four nearest neighbors - determined using the position of the sample image - are used to estimate the position of a sample image. In addition to the five similarity metrics, two additional metrics are used to evaluate the performance of the proposed metrics: Euclidean distance and equal distance. Theoretically, Euclidean distance should result in the best possible interpolated position as it accurately represents an image position's distances to its neighbors, which is the aim of keypoint-based metrics. Equal distance assumes a unit distance between the current RGB image position and all neighbor dataset images and provides

no information regarding the similarity between the current image and dataset images, which should result in the worst performance.

As expected, the Euclidean distance metric resulted in the minimum error in all datasets. Interestingly, the equal distance metric was not the worst-performing one in some datasets: IM performed worse than the equal distance in the L_1, L_3, and L_5 datasets. In other ones, the performance is similar to those of equal distance. The performance of IMS is very similar to IM except for the L_1 dataset. This suggests that these metrics are not a reliable similarity measure. As IMSR is observed to perform better than IM and IMS in all datasets, one can conclude that the relation between the similarity of the two images and the number of matched keypoints in them is more of an inverse square root one than it is an inverse or inverse square one. Excluding the Euclidean distance, ADDIMSR performs the best in all datasets with ADDIM being the second. Both of these metrics estimate the actual distance between two images based not only on the number of matched keypoints, but also on their matching quality.

These results demonstrate that as the density of the dataset increases, average interpolation errors decrease. There may be two significant factors behind this result: as the density increases, the distance between the correct position of the sample point and the data points used in the interpolation decreases. As SkNNI interpolates the position using a normalized linear combination of the position of data points, the estimated point must be in between the convex hull created by the dataset points used during the interpolation, which is getting smaller with increased density. Hence, this performance increase might be misleading. The other factor may be that the algorithm is performing better due to increased overlaps with dataset images with increased density. To remove the effect from varied convex hull areas with varying dataset density, errors are divided by the average distance between data points used during interpolation to normalize them. Results are shown in Table 3.3.

Table 3.2. Average interpolation-based localization errors in meters.

Dataset	Euclidean distance	$\frac{1}{n}$	$\frac{1}{n^2}$	$\frac{1}{n^{1/2}}$	$\frac{f_s}{n}$	$\frac{f_s}{n^{1/2}}$	Equal distance
L_1	3.41278	9.00016	8.22024	6.44842	7.39327	**5.83439**	8.91287
L_2	1.27832	3.91573	3.92036	3.29607	2.68447	**2.61641**	3.94852
L_3	0.477312	1.70993	1.70934	1.59025	1.47367	**1.43405**	1.68842
L_4	4.47171	6.16996	6.17097	5.54	5.25002	**5.08474**	6.17109
L_5	1.95485	2.88191	2.84475	2.76987	2.62772	**2.62448**	2.83895

Table 3.3. Normalized average interpolation-based localization errors in meters.

Dataset	Euclidean distance	$\frac{1}{n}$	$\frac{1}{n^2}$	$\frac{1}{n^{1/2}}$	$\frac{f_s}{n}$	$\frac{f_s}{n^{1/2}}$	Equal distance
L_1	0.1030	0.2717	0.2481	0.1947	0.2232	**0.1761**	0.2691
L_2	0.0749	0.2295	0.2298	0.1932	0.1574	**0.1534**	0.2315
L_3	0.0553	0.1983	0.1982	0.1844	0.1709	**0.1663**	0.1958
L_4	0.4237	0.5846	0.5847	0.5249	0.4974	**0.4818**	0.5847
L_5	0.3061	0.4513	0.4455	0.4337	0.4115	**0.4110**	0.4446

Normalized results show that increasing the density still increases the performance, but not as significantly as before normalization. This suggests that increased overlap between the sample image and the reference images increases the performance. The algorithms with all the metrics performed better in datasets with regular grids than the datasets with random ones. A performance increase from a random dataset to a grid-based one with the same number of data points is understandable. However, the algorithms performed better even in the L_1 dataset with 22 data points than they performed in the L_5 dataset with 229 data points. This shows the importance of regular data in position estimation with the algorithm presented here.

3.4.2.2. Geometry-Based Localization Experiments. The results of geometry-based methods are shown in Table 3.4. The two-stage update method with gradient descent did not result in meaningful estimations as the depth matrix computation does not converge. Hence, it is not shown in the table. Various step sizes and iteration numbers are used to tune the algorithm with no correct results. This suggests that this approach may need to be revised.

Among the three PnP methods, Iterative PnP performs worse in all datasets. This is expected as SQPnP is guaranteed to converge to the minimum of the same problem, and PnP with RANSAC is able to filter out geometrically incorrect keypoint matches. Additionally, the Iterative PnP algorithm diverges while calculating the positions of some of the sample frames in the L_2, and L_4 datasets. SQPnP performed consistently, falling behind PnP with RANSAC in median positional errors. In terms of operation speed, measured by Frames per second (Fps), all three methods performed similarly. The slight variances of speeds over different datasets may be explained by the varying number of matched keypoints in these datasets. As the overlap ratio of the dataset images increases, the number of keypoints in the local 3D representation, hence, the number of keypoints matched with the sample image increases. This increases the load of the PnP method and thus results in a reduced Fps.

Table 3.4. Geometry-based localization results - Mean and median errors in meters.

Dataset	Iterative PnP			SQPnP			PnP with RANSAC		
	Mean	Median	Fps	Mean	Median	Fps	Mean	Median	Fps
L_2	3.26803e+09	0.382809	6.02591	2.28263	0.257438	6.09586	1.62097	0.0334849	6.19234
L_3	3.41662	0.338267	4.96318	0.969417	0.26954	5.0523	0.862949	0.0334625	5.01852
L_4	2.41069e+09	0.927578	5.13058	1.93362	0.454066	5.21058	1917.61	0.0731875	5.08816
L_5	7.11303	1.07768	4.80193	1.81029	0.749372	4.8653	0.914082	0.0682231	4.83037

Table 3.5. Interpolation-based vs geometry-based localization comparison - mean and median errors in meters.

Dataset	Interpolation, $\frac{Q}{n^{1/2}}$			PnP with RANSAC		
	Mean	Median	Fps	Mean	Median	Fps
L_2	2.61641	2.36536	3.03469	1.62097	0.0334849	6.19234
L_3	1.43405	1.26593	3.04982	0.862949	0.0334625	5.01852
L_4	5.08474	4.00873	3.07028	1917.61	0.0731875	5.08816
L_5	2.62448	2.06215	3.07376	0.914082	0.0682231	4.83037
L_5, varying sample height	10.7445	11.2233	3.29926	0.508658	0.185218	4.413462
L_6, varying sample height	5.7867	5.41893	3.58322	0.833177	0.193845	6.62062

In terms of median positional error, PnP with RANSAC performed the best with errors in the order of centimeters. This is a promising result as this level of accuracy proposes significant performance increases over most used positioning systems such as GNSS-based ones. However, the mean errors suggest that the positioning performance among sample frames may not be consistent. This can be explained by some sample frames not having proper keypoints for this type of position estimation. For example, in the L_4 dataset, the average error increased to 1917.91 meters, possibly due to the RANSAC method converging to an incorrect model. The threshold of the RANSAC may be fine-tuned to prevent this kind of error. Still, such incorrect estimations with high positional errors imply that rather than using as stand-alone position estimators, these methods should be used with some verification methods as to increase reliability.

Differing from the interpolation-based approaches, we have not observed a correlation between dataset density and estimation performance other than positional errors diverging in some sample frames. This suggests that the densities of the sparse datasets are sufficient in general to be used with these methods. Considering the mean errors as in the interpolation-based position estimations, all three methods performed better in datasets with regular grids. This result consolidates the inference about the importance of regular data in position estimation.

In Table 3.5, the best-performing methods of both interpolation (ADDIMSR) and geometry-based approaches (PnP with RANSAC) are presented. Here, one additional dataset consisting of randomly distributed 100 reference images with varying heights , L_6 is used. In addition to their lateral positions inside the polygon, the heights of the images in this dataset are determined uniformly randomly between 40 and 60 meters. Also, an additional sample set containing 100 images with their respective heights varying uniformly between 40 and 60 meters is used. In terms of median errors, the performances of both methods have degraded in the L_6 dataset. For PnP with RANSAC, this can be explained by the increased difficulty of correctly matching keypoints across different images. Although SIFT keypoints are invariant to scale changes of the image in theory, this invariance is limited by the finite resolution of the sensors.

As the distance of the respective image to the scene, its height, changes, the descriptors of the keypoints of the same location in the 3D world also change. This prevents the method from detecting correct 2D pixel position - 3D position correspondences, hence reducing the performance. The performance reduction of ADDIMSR is observed to be more dramatic. In addition to the increased hassle of correctly matching keypoints to estimate a similarity, as the position of the sample frame is estimated based on a linear combination of the positions of the respective dataset images, and dataset images are at the same height, the estimated position must lie on the plane crossing from the dataset images. This shows the limited capability of ADDIMSR in scenarios when the position of the sample image can not be represented by a linear combination of respective dataset images. With the same sample set, the performance of ADDIMSR increased significantly in the dataset with varying image heights. The low speed of ADDIMSR compared to PnP with RANSAC can be explained by the number of dataset images to estimate the position. Keypoints of four different dataset images are extracted to estimate the position of a single image in the ADDIMSR, while PnP with RANSAC requires only two. Given that keypoint extraction is a computationally expensive process compared to the required computations for the interpolation with SkNNI, it can be said that the speed of AADIMSR mainly depends on the number of dataset images used. In terms of median error, PnP with RANSAC performed significantly better than ADDIMSR in all datasets. This suggests that using geometric relations increases the estimation performance significantly. This differs from estimating the positions of the sample images solely on their similarities to images with known positions. However, the ratio of mean and median errors shows that PnP with RANSAC is not performing as consistently as ADDIMSR when estimating the positions of sample images. For example, a mean error of 1917.61 meters in the L_4 dataset shows that estimated positions may diverge from their actual values for some sample images. Although their performances are satisfactory in general, geometric relation-based methods may not be sufficient to be used as stand-alone localization systems due to high errors when their position estimations diverge from the actual value. Rather, if hybrid systems are utilized, it may be possible to attain both the consistency of interpolation-based methods and the high accuracy of geometric relation-based ones.

4. COORDINATING ROBOTS SYSTEM

This chapter presents work done on the design and development of a real coordination system. The designed system consists of the following components:

- A changeable battery system for the AR,
- A battery replacement platform,
- An MGR that carries the BRP.

4.1. Related Work

This design and development of such a system needs to develop three primary components: i) Battery replacement; ii) Battery replacement platform; and iii) MGR that is capable of carrying the platform This section presents a brief summary of work done in each area.

Autonomous charging of AR batteries has been gaining popularity. As discussed, two alternative approaches are possible: Recharging the depleted battery or swapping it with a charged battery. Initial methods for recharging generally require setting up the necessary electrical connections for each recharge. Recently, wired charging of AR batteries has been utilized in continuing flights using power transmission lines. This method has high potential in areas with power transmission lines. Wireless battery recharging has also been considered. As it requires installing wireless charging hardware on ARs, while this does not require a physical electrical connection, the weight of the AR is increased and thus its payload capability is reduced. Especially with small ARs with limited payload capacities, the benefits of this method become limited. Additionally, power losses are high with wireless charging. If the AR is being recharged with a mobile platform running on batteries, this issue becomes critical. Finally, as the power that can be transmitted with wireless charging is limited, the charging time of the AR is increased. In all, recharging the battery of an AR takes significant time.

Considering the charge and discharge rates of the current batteries, The mission time will be dominated by the recharging times.

The alternative approach is battery replacement. This method is advantageous as it reduces the ratio of battery replenishment time with respect to total mission time. Typically in ARs, batteries are secured to their position using Velcro straps. This placement method is applicable to batteries with different sizes and shapes. In this method, electrical connection is done using existing connectors such as XT60. The required physical tasks to displace a battery placed this way can be divided into two main subtasks: Ripping off the Velcro strap and disconnecting the connector. In general, both of these tasks require a human to use both hands. Alternatively, some commercial products have custom-designed electrical connectors and battery cases such that the battery case can be placed into a case holder and locked by simpler movements [69]. However, although it is simple for humans to place the battery case into the case holder, the low tolerance of such systems to positional errors while placing the battery case complicates the problem of autonomously replacing the battery.

Early work on autonomous battery replacement has focused on the BRP part of the problem [70]. Their proposed system includes the following components: i) Guidance of AR to the BRP, ii) Orienting the AR as needed, iii) Removing AR-battery connections, iv) Placing the charged battery in the AR, v) Charging the depleted battery and vi) Having the AR return to its task. A BRP focusing on an active landing guide is proposed in [71]. In a more recent work, a depleted battery is swapped to a charged battery while keeping the AR in its active state [72]. As the current consumptions of ARs are high due to low battery voltages and high power consumptions, physically tight connectors are used with them. This makes the replacement of the battery harder. However, in all these work, swap stations are stationary and hence it is not possible for the AR to fly long distances as it needs to ensure safe reachability to a charging station.

Alternatively, mobile charging stations have been proposed. Here, one approach is to use mobile ground robots that are capable of carrying the BRP around. Depending on their actuation types, various robot hardware types can be used. For example, UGVs equipped with omnidirectional wheels offer high maneuverability by enabling the vehicle to move in all lateral directions [73]. However, their wheels are sensitive to external particles and hence are not suitable for offroad usage. Tracked vehicles offer excellent performance in offroad environments due to the strong grip provided by their tracks. However, they require high maintenance due to the complex structure of their tracks. Ackerman-steered vehicles with standard wheels offer decent offroad performance. In terms of maneuverability, however, they do not offer much. Skid-steer vehicles with standard wheels offer both decent offroad performance and high maneuverability. The properties of various drive types are summarized in Table 4.1 based on related literature.

Table 4.1. Properties of various drive types. *: Bad, **: Decent, ***: Excellent.

Drive type	Offroad performance	Maneuverability	Maintenance
Omnidirectional	*	***	*
Tracked	***	**	*
Ackerman-steered	**	*	**
Skid-steered	**	**	***

Outdoor navigation requires position and orientation data. In a coordinated mission with the AR, there must be a common reference for position estimations to be able to achieve a successful rendezvous. Additionally, the MGR should be able to navigate on a road network, whose position on Earth is predetermined. Thus, GNSS modules are used. GNSS works by measuring the distance of a GNSS antenna to the satellites in orbit, whose positions are known. This way, the position of the sensor in geodetic coordinates can be calculated. In addition to the GNSS modules, an IMU, which can measure linear accelerations and angular velocities, should also be included in the system to be able to estimate angular position and enhance linear position

estimation.

Unlike the standard GNSS modules, RTK-GNSS rover modules are capable of measuring geographic position with centimeter-level accuracy using correction data [57]. This correction data includes the delays of the signals from satellites due to atmospheric conditions and is specific to region and time. This correction data can be generated by RTK-GNSS base stations whose positions are known with high accuracy. Alternatively, a moving base station whose position is known with low accuracy can generate the correction data by calculating the delays of satellite signals based on its position with low accuracy. Although this correction data does not represent the actual delays of the satellite signals, the position of an RTK-GNSS rover module that uses this correction data relative to the position of the correction source can be measured with high accuracy [74]. This is called moving base operation. Using one of the GNSS receivers as the RTK-GNSS base, the relative positions of the two modules can be used as an additional data source to estimate the heading of the MGR.

4.2. Battery Replacement Platform

In this thesis, a novel battery replacement platform has been designed. The novelties of this platform are as follows:

- The basic component of this platform is the Easily Replaceable Battery Casing (ERBC). As such, the battery replacement platform can replace the batteries of ARs equipped with an ERBC.
- A robot arm-based battery replacement system is designed.
- A camera is used to provide positional information of the AR to the arm.
- The BRP is designed to be able to stabilize the landing platform to always maintain a horizontal orientation. This is critical as landing and taking off from a platform with a slope may require modifications to the control modules of the existing ARs and hardens the work of battery replacement.
- The BRP is able to hold up to five batteries and recharge them.

4.2.1. Easily Replaceable Battery Casing

The ERBC design has been made considering the following criteria:

- It should be able to conduct the required high currents (75A in our case) from the battery to the body of the AR.
- It should be able to strongly secure the battery to the body of the AR considering its accelerations.
- It should be vibration-resistant.
- When placed, it should be easy to lock and unlock with a robot arm.
- It should have some tolerance for misplacement during battery replacement.

A casing is designed using SolidWorks software and produced using a 3D printer. The resulting casing design consists of five parts that are assembled together.

Case_bottom: The casing system requires a physical interface fixed on the AR to be placed and displaced. *Case_bottom* as seen in Figure 4.1 enables this. It is permanently mounted on top of the AR. When placed, it is responsible for firmly gripping the rest of the casing system. To do so, two tower-like structures are designed. With their pointy tops, these tower-like structures also provide some level of tolerance to positional errors when placing the rest of the casing. Additionally, rectangular holes in them are used to mount the locking mechanism. An ArUco visual marker is placed on it in order to enable vision-based positioning.

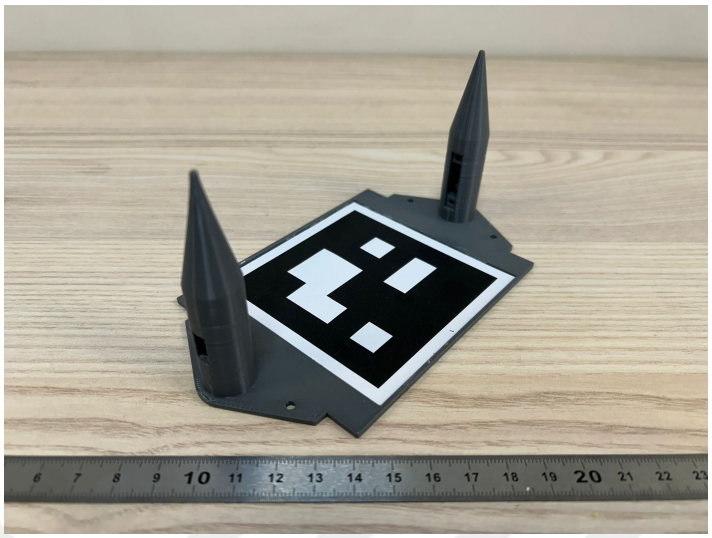


Figure 4.1. *Case_bottom*.

Case_middle: This part is responsible for holding the battery. With cylindrical holes in both ends, it can be gripped by the *Case_bottom*. Like the *Case_bottom*, it has rectangular holes for the locking mechanism as shown in Figure 4.2. The interior dimensions for battery placement are $10 \times 8 \times 6.5$ cm^3.

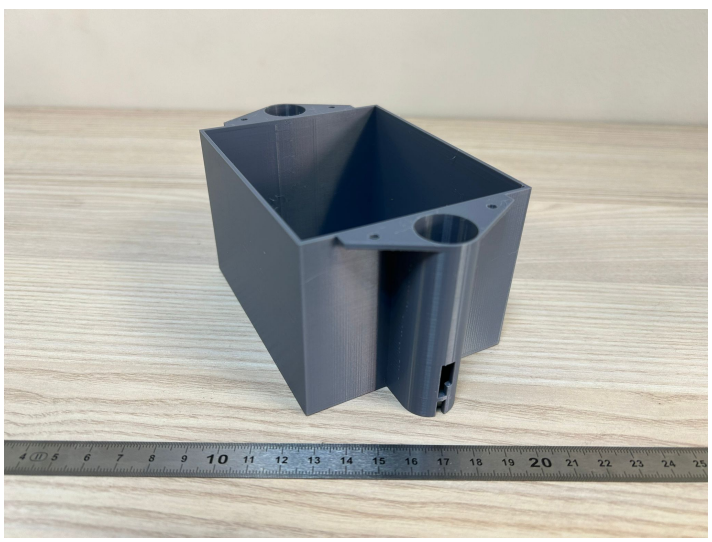


Figure 4.2. *Case_middle*.

Case_lid: This part is used as a lid to the *Case_middle* to close the top of battery in it. It can be seen in Figure 4.3. After placing the battery in *Case_middle*, *Case_lid* is permanently fixed to it.

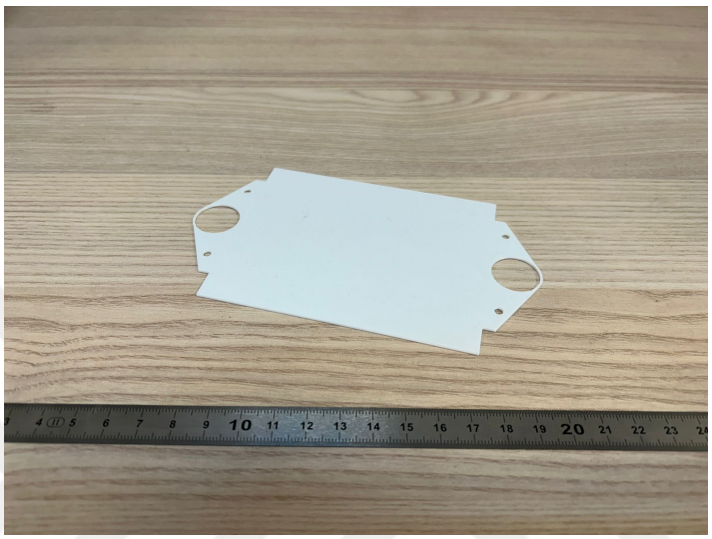


Figure 4.3. *Case_lid*.

Case_top: This part is used as a mechanical interface for the robot arm. As seen in Figure 4.4, it enables the arm to move the casing system. It is connected to the *Case_middle* in such a way that it can move up and down relative to *Case_middle* for one centimeter. This movement allows unlocking the locking system between *Case_bottom* and *Case_middle*. On the top side of it, it has an ArUco marker for positioning it.

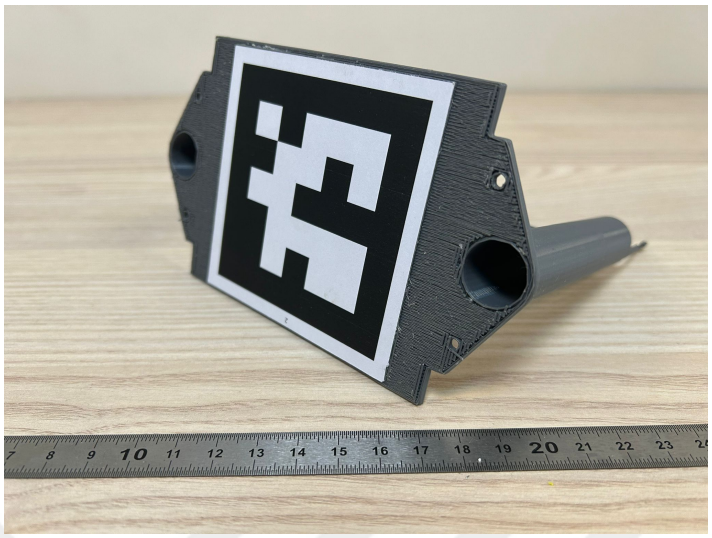


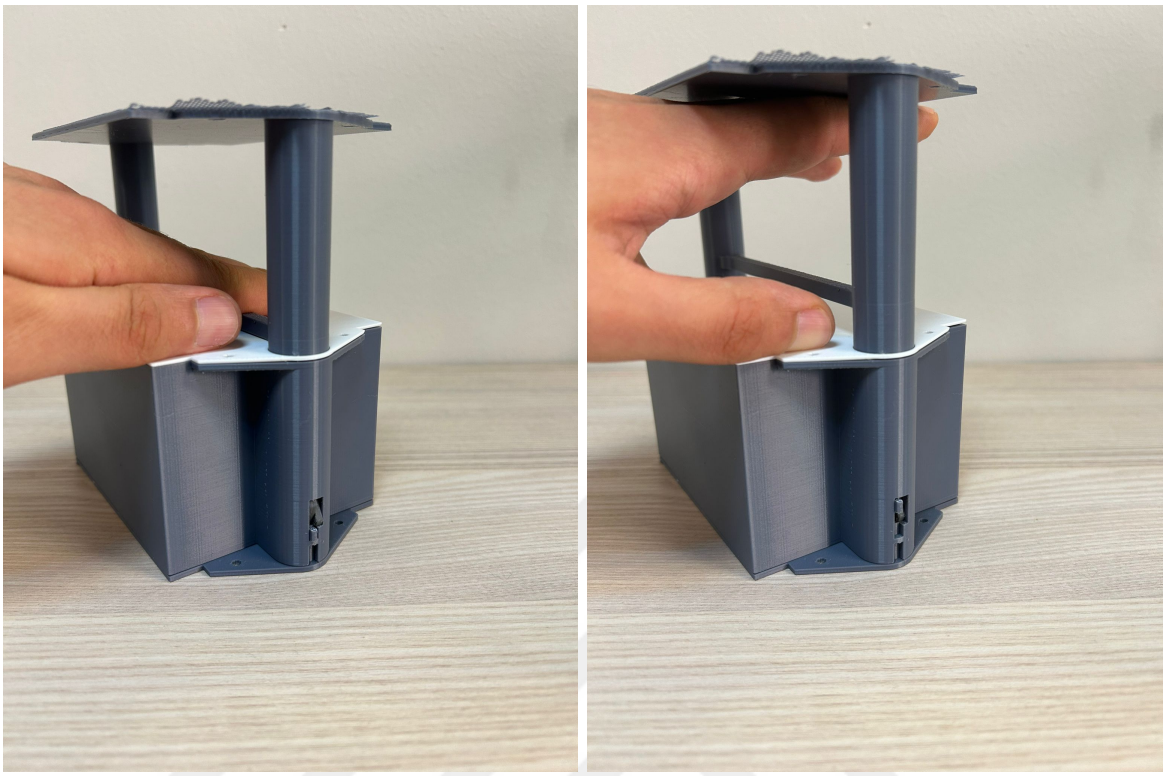
Figure 4.4. *Case_top.*

Case_latchbolt: This part is used to lock *Case_middle* and *Case_bottom* together, as seen in Figure 4.5. Similar to a door latch bolt, it is pushed out with springs to lock the system. This way, unintended displacements of the casing system are prevented. The locking mechanism can be unlocked by moving the *Case_top* to up.



Figure 4.5. *Case_latchbolt.*

The assembled system can be seen in Figure 4.6. In Figure 4.6(a), the assembly is at its base state and the *Case_bottom* locks the *Case_middle*. In Figure Figure 4.6(a), the system is unlocked by moving the *Case_top* up and the battery can be removed from *Case_bottom*. Detailed technical drawings of these parts are presented in Appendix C.



(a) Locked. (b) Unlocked.

Figure 4.6. Assembled battery case: (a) locked and (b) unlocked states.

The ERBC can be both unlocked and displaced by pushing the *Case_top* to up. To place and lock a new battery, aligning the *Case_top* of it on top of the *Case_bottom* and pushing it downward is sufficient. Thus, ease of replaceability for robot arms is ensured with this design. The strength of the connection can be arranged by changing the 3D printing settings of the tower-like structures of *Case_bottom*. Electrical connections are not done yet and are left as future work. Tolerance to placement errors is ensured with pointy ends of the tower-like structures.

4.2.2. Platform Design

The BRP system realizes battery replacement whenever AR is in the dedicated landing area. A top view of the proposed design is as shown in Figure 4.7. The components of the BRP and their functions are listed in Table 4.2.

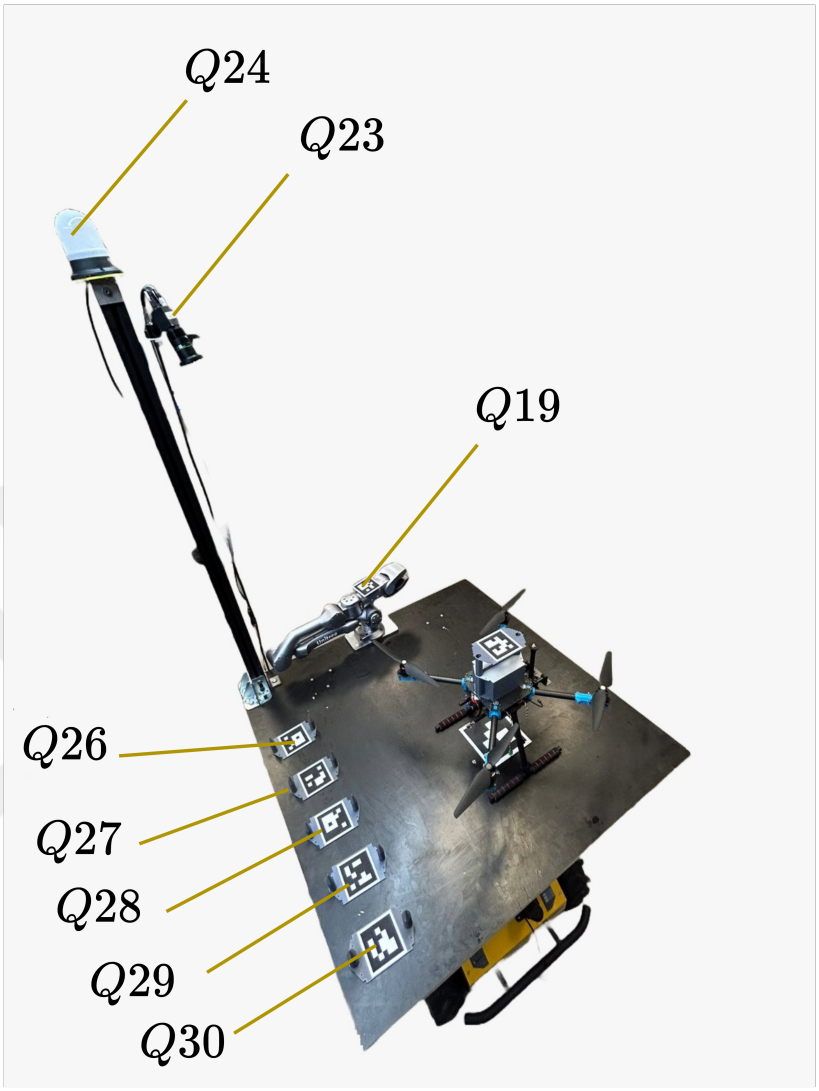


Figure 4.7. BRP top view.

A plywood plate with an area of 100×100 cm^2 and 18mm thickness is used as the basis of the platform. A 60×60 cm^2 portion of this area is dedicated as the landing and take-off area for ARs. As landing positions and orientations may vary depending on landing circumstances, it should be possible to compensate for these positional and orientational variations. The platform is stabilized horizontally, This is achieved by using three linear actuators (CNMAWAY 300mm 12V). These motors are controlled by three motor drivers (IBT-2). In particular, their speeds are controlled by changing the duty cycles of output power to the motors. This is achieved by the Pulse Width Modulation (PWM) signals from the computation unit. The input power to these

motors and DC cooling fans is supplied through a DC-DC buck converter with 12V voltage. A 6-DOF IMU is installed for orientation estimation.

A robot arm (Unitree Z1) is placed on the platform for the replacement of the battery. This robot arm operates with 24V power supplies and consumes up to 500W power. A DC-DC buck-boost converter (MeanWell SD-500) is used to regulate the battery voltage to power the robot arm. An RGB camera (FLIR BFS-PGE-200S7C-C) is used to detect visual markers to estimate the position of the AR. A lens (Fujinon CF8ZA-1S) to ensure proper FOV and focus distance of the camera is installed on the camera. A sigma profile pole with a 120cm length is placed on top of the platform to hold the camera. A warning lamp is installed on top of the pole. To connect the computation unit, robot arm, camera, and the MGR, an ethernet switch is used. As the computation unit to control the robot arm to replace the battery, the Nvidia Jetson Orin Nano Developer Kit is used. It is powered by a voltage regulator (XL4015) to ensure 19V power input. As such, the BRP is designed to work with 6S lithium batteries with voltages ranging from 19.2V to 25.2V. The electronic components of the BRP are installed on a plexiglass plate named BRP electronics plate. The connections with the components on it and out of it are done through the BRP connectors bay. The components on the BRP electronics plate are shown in Figure 4.8. The functions of the connectors in the connectors bay are listed in Table 4.3.

The charging of lithium batteries is typically done by the constant current-constant voltage (CC/CV) method [75]. In this method, the battery is charged with a predetermined constant current until it reaches a voltage such that the battery draws no more current than this predetermined current when a constant voltage is provided. At this point, charging continues with constant voltage. DC-DC buck converters with current limits can be used for such an operation. To do so, five XL4016 converters to charge five batteries are used. Output voltages of the converters are set to the voltage of 4S batteries, 16.2V, and their output currents are limited to 8A. In order to minimize the heating of these converters, DC cooling fans are placed next to them.

Table 4.2. Descriptions of the BRP components.

Component	Description
Q1	Computer, Nvidia Jetson Orin Nano Developer Kit. Responsible for camera driving, visual processing, robot arm control, informing mobile ground robot about the status of battery replacement, platform attitude estimation, and platform horizontal stabilization control signals.
Q2	Ethernet switch, Trendet TEG-S50g. Responsible for communication between Q1, Q19, Q23, and mobile ground robot.
Q3	IMU, MPU6050. Responsible for measuring linear accelerations and angular velocities.
Q4	DC-DC buck converter, XL4015. Responsible for providing 5V to Q2 and C6.
Q5	DC-DC buck converter, XL4015. Responsible for providing 19V to Q1.
Q6	DC-DC buck converter, XL4015. Responsible for providing 12V to C11.
Q7	Motor driver, IBT-2. Responsible for driving Q20 according to signals from Q1.
Q8	Motor driver, IBT-2. Responsible for driving Q21 according to signals from Q1.
Q9	Motor driver, IBT-2. Responsible for driving Q22 according to signals from Q1.
Q10	Cooling system, Profuse P5010M. Responsible for cooling Q11-Q15.
Q11	DC-DC buck converter, XL4016. Responsible for charging the battery in Q26.
Q12	DC-DC buck converter, XL4016. Responsible for charging the battery in Q27.
Q13	DC-DC buck converter, XL4016. Responsible for charging the battery in Q28.

Table 4.2. Descriptions of the BRP components. (cont.)

Component	Description
Q14	DC-DC buck converter, XL4016. Responsible for charging the battery in Q29.
Q15	DC-DC buck converter, XL4016. Responsible for charging the battery in Q30.
Q16	DC-DC buck-boost converter, Meanwell SD-500. Responsible for providing 24V to C12-, C14 when the voltage of C7 is between 0-0.8V and cutting power when the voltage of C7 is between 4-10V.
Q17	DC-DC buck converter, SZBK07. Responsible for providing 12V to Q7-10.
Q18	Connector Bay. Responsible for providing power interface of the BRP electronics plate.
Q19	Robot arm, Unitree Z1 Pro. Responsible for replacing the battery according to control commands from Q1.
Q20	Linear actuator, CNMAWAY 300mm 12V. Responsible for horizontal stabilization of the platform according to signals from Q7.
Q21	Linear actuator, CNMAWAY 300mm 12V. Responsible for horizontal stabilization of the platform according to signals from Q8.
Q22	Linear actuator, CNMAWAY 300mm 12V. Responsible for horizontal stabilization of the platform according to signals from Q9.
Q23	RGB camera, FLIR BFS-PGE-200S7C-C. Responsible for providing visual information for battery replacement.
Q24	Warning lamp, Mucco SNT-74-B-RGB. Responsible for warning people.
Q25	Reserved for illumination lamp.
Q26	*Case_bottom*. Responsible for holding a spare battery.
Q27	*Case_bottom*. Responsible for holding a spare battery.
Q28	*Case_bottom*. Responsible for holding a spare battery.
Q29	*Case_bottom*. Responsible for holding a spare battery.

Table 4.2. Descriptions of the BRP components. (cont.)

Component	Description
Q30	*Case_bottom*. Responsible for holding a spare battery.
Q31	Emergency switch, Telemecanique XAL-K174. Responsible for providing 0-0.8V to C7 when not pressed and 4-10V when pressed to cut the power of Q2 and Q24.
Q32	Reserved for battery.

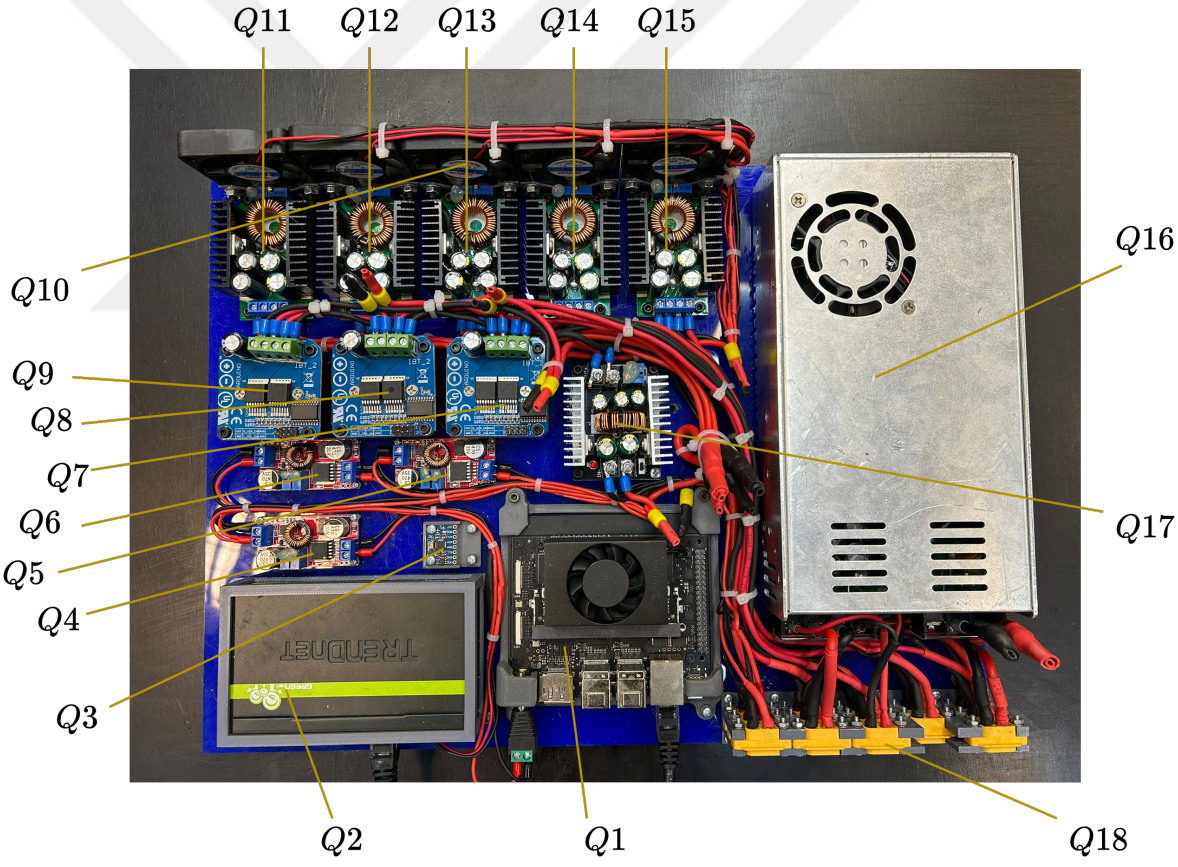


Figure 4.8. BRP electronics plate.

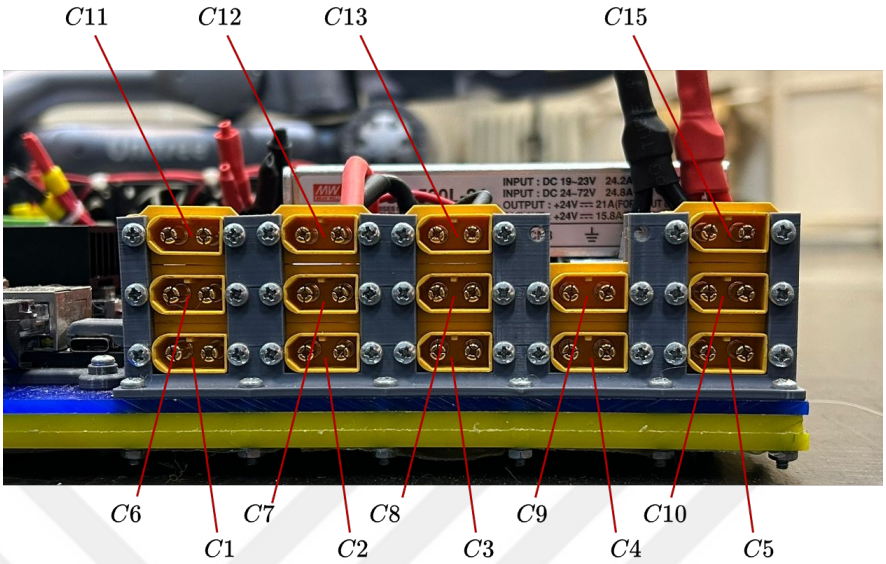


Figure 4.9. BRP connectors bay.

Table 4.3. Functions of the connectors.

Connector number	Function
C1	Q26 charging port
C2	Q27 charging port
C3	Q28 charging port
C4	Q29 charging port
C5	Q30 charging port
C6	5V for Q31
C7	Q16 emergency signal input
C8	Q20 power socket
C9	Q21 power socket
C10	Q22 power socket
C11	Q23 power socket
C12	Q19 power socket
C13	Q24 power socket
C14	Reserved for Q25
C15	BRP power input

4.3. Mobile Ground Robot

The MGR design is based on modifying the body of an existing unmanned ground vehicle that is capable of navigating off-road. As such, coordinated AR-MGR missions may take place in environments with insufficient infrastructure as well. In this work, the Husky A200 mobile ground platform from Clearpath Robotics is used [76]. This is a medium-sized skid-steer 50kg vehicle with up to 75kg payload capacity. It has a maximum speed of 1m/s and an operation time of four hours with typical usage. The following design considerations needed to be made in order to turn it into an MGR:

(i) Hardware revision: Various sensors and processors need to be added.
(ii) Software design and development: Methods for control, state estimation and environmental perception need to be designed and implemented.

4.3.1. Electronics and Electrical Design

Various sensors and computation units have been installed on the platform. These components are listed in Table 4.4. The power of the electronics is provided by Husky's battery using a DC-DC buck converter. A computation unit consisting of an Nvidia Jetson Orin AGX is used for control, navigation, and environmental perception of the robot. A 2D laser range finder (Hokuyo UTM30LX) is placed on the front side of the Husky aluminum plate. This will enable the robot to collect obstacles' data within a maximum measurement range of 30 meters.

Table 4.4. List of MGR electronic components.

ID	Name	Model
M1	Computation unit	Nvidia Jetson Orin AGX
M2	Network switch	HP switch
M3	RTK-GNSS base antenna	Harxon HX-CSX627A
M4	RC receiver	FrSky X8R
M5	RTK-GNSS base module	Holybro H-RTK F9P
M6	RTK-GNSS rover module	Holybro H-RTK F9P
M7	Pixhawk	Holybro Pixhawk 6X
M8	Data transmitter	SIYI HM30 air unit
M9	Data transmitter antenna	SIYI standard omni antenna
M10	DC-DC buck converter	SZBK07 DC-DC buck converter
M11	RTK-GNSS rover antenna	Harxon HX-CSX627A
M12	MGR electronics plate	Standard plexiglass
M13	Pixhawk power module	PM02D power module
M14	2D laser scanner	Hokuyo UTM30LX
M15	Operator camera	SIYI IP camera

The robot is also endowed with an IMU for measuring linear velocities and angular accelerations and with two RTK-GNSS modules for self-localization. Although the IMU inside the Pixhawk measures the orientation of the robot, the only information source about the geographic heading of the robot is the magnetometer in the IMU, which relies on the magnetic field of the Earth to measure the orientation. However, with a high amount of metals and magnetic field sources in the MGR, the measurements of the magnetometer may corrupt. To ensure a stable heading estimation, which is critical for the operation of the MGR, an additional data source based on two RTK-GNSS modules is used. The resulting data from these sensors should be processed in some way to achieve state estimation. This is done by using an existing software, PX4 [77]. PX4 is able to estimate position using an Extended Kalman Filter (EKF) based on data from various sensors with sensor fault detection capability. The software

can run on STM32-based microcontrollers. To run the PX4, a Pixhawk-based module, Pixhawk 6X from Holybro is used. Pixhawk [78] is a hardware architecture that was originally developed with a focus on ARs. It is designed to run PX4. This hardware architecture also contains IMU and barometer to feed the state estimator of PX4.

The main processor, Pixhawk, and BRP are connected to each other via an ethernet switch. During the operation of the MGR, operator intervention may be required especially in case of emergencies. For this, a Radio Control (RC) unit is used to send manual control commands to the robot. The receiver of the RC unit is connected to the Pixhawk 6X. Additionally, to transmit real-time video and telemetry data to the operator, including position, velocity, and battery voltage of the robot, a long-range data transmission system, SIYI HM30 is used. The transmitter of this system is connected to the operator camera on the front of the robot and to the Pixhawk 6X for telemetry data.

Except for the antennas and the sensors, all components are mounted to a plexiglass plate named MGR electronic plate as seen in Figure 4.10. This plate is mounted to the Husky, using silicon dampers underneath it to reduce the effect of vibrations and shocks on the electronic devices. In addition, antennas and sensors are mounted to the aluminum plate of the Husky as seen in the same figure.

4.3.2. Robot Control

In its base form, Husky has no autonomy. The robot is controlled by sending linear v and angular ω velocity commands via onboard serial connection. The resulting right wheel v_r and left wheel v_l velocities are

$$v_r = v + \omega * \frac{w}{2}, \qquad v_l = v - \omega * \frac{w_s}{2} \tag{4.1}$$

where w_s is the track separation of right and left wheels.

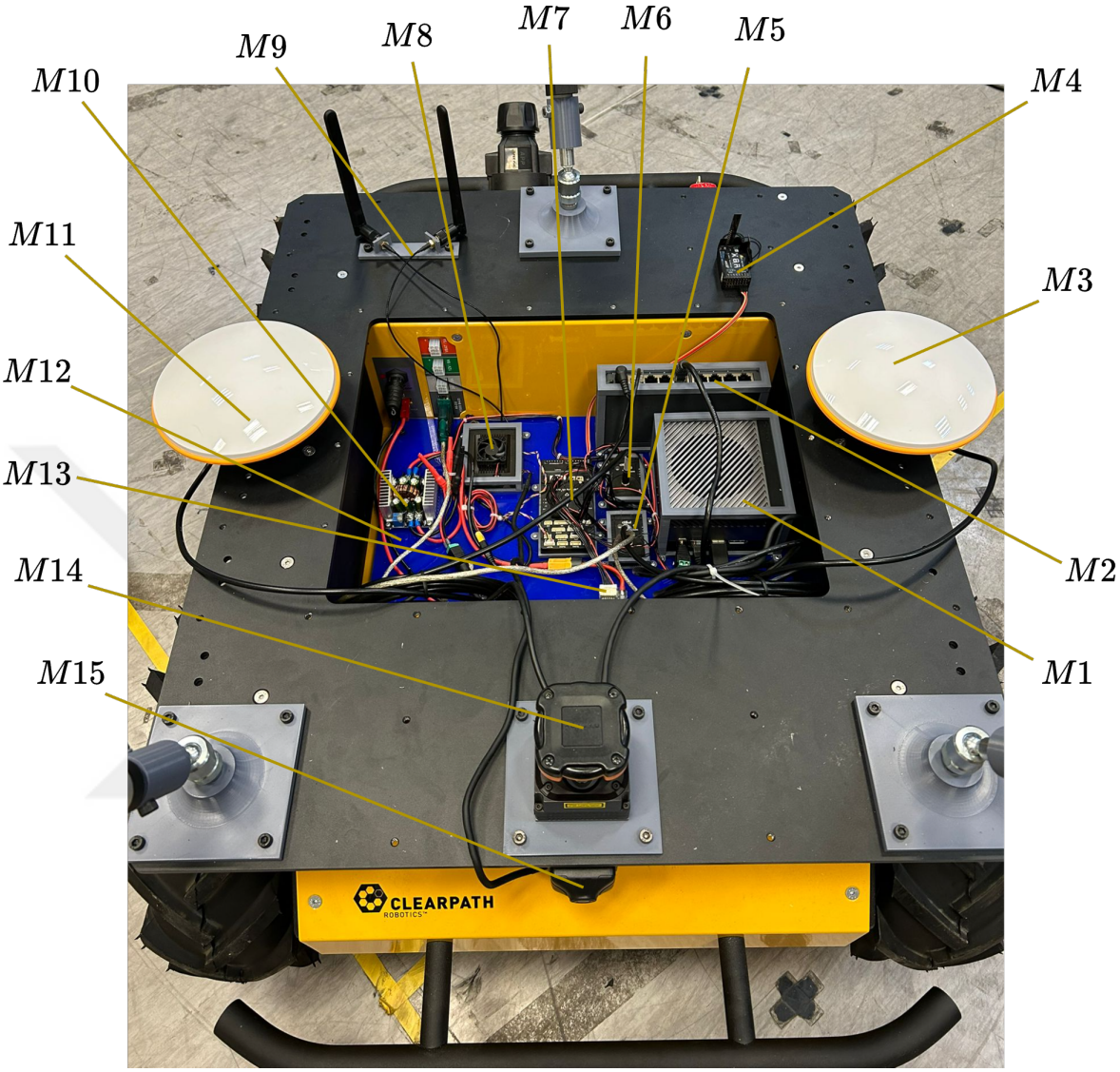


Figure 4.10. MGR hardware.

The control inputs are generated based on the 2D carrot chasing method [79]. The route of the MGR is represented by a collection of line segments that connect the positions of nodes in the road network. Hence, input to the control method is a line segment and state of the robot as shown in Figure 4.11. Consider a line segment starting at P_{src} and ending at P_{dst}. Let P_{rbt} be the 2D horizontal position of the robot. Let P_{prj} be the computed projection of P_{rbt} to the line passing through P_{src} and P_{dst}. The point P_{lad} on this line that is ahead of P_{prj} by a predetermined look-ahead distance

γ_a can be computed as

$$P_{lad} = P_{prj} + \frac{\overrightarrow{P_{src}P_{dst}}}{||\overrightarrow{P_{src}P_{dst}}||}\gamma_a. \tag{4.2}$$

Using the direction of the vector $\overrightarrow{P_{rbt}P_{lad}}$ and heading of the robot ψ_{rbt}, the heading error ψ_{err} is defined as

$$\psi_{err} = \arctan\left((\overrightarrow{P_{rbt}P_{lad}})_y, (\overrightarrow{P_{rbt}P_{lad}})_x\right) - \psi_{rbt} \tag{4.3}$$

where arctan is the four-quadrant inverse tangent function. Using PD control, angular velocity command ω_{cmd} can be calculated using ψ_{err} as

$$\omega_{cmd} = \gamma_p * \psi_{err} + \gamma_d * \frac{d(\psi_{err})}{dt} \tag{4.4}$$

where, γ_p is the proportional gain and γ_d is the derivative gain. The linear velocity command, v_{cmd} is generated based on a predetermined robot maximum speed v_{mgr} and heading error as

$$v_{cmd} = \begin{cases} 0 & \text{If } |\psi_{err}| > \pi/2 \\ v_{mgr} * \left(1 - \frac{\psi_{err}}{\pi/2}\right)^2 & \text{Otherwise.} \end{cases} \tag{4.5}$$

The idea behind limiting the linear velocity of the robot according to the heading error to the path is to prevent the MGR from drifting apart from the route when its heading is far from the reference heading. Whenever the heading error is greater than $\frac{\pi}{2}$, the linear velocity becomes zero and remains so until the robot turns to a suitable direction. The control inputs are sent as long as the P_{prj} is behind the P_{dst}, or formally, $d(P_{src}, P_{prj}) < d(P_{src}, P_{dst}) \mid d(P_{src}, P_{prj}) < d(P_{dst}, P_{prj})$. Then, P_{src} and P_{dst} are updated according to the current route until the route ends.

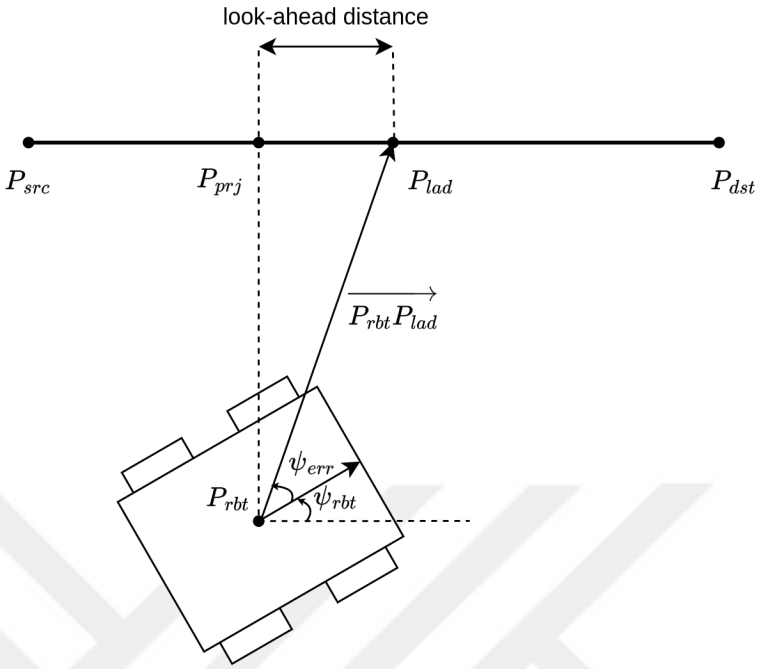


Figure 4.11. MGR motion control variables.

4.3.3. Software Architecture

The software architecture of the MGR is shown in Figure 4.12. It is developed using Robot Operating System (ROS) and C++ running on Ubuntu 20.04. The following packages are used:

- The Husky robot motion capability based on available software following [80]. In particular, the serial connection of the computation unit with Husky is established using the package *husky_base* from Clearpath Robotics.
- To drive the laser range finder, *urg_node* is used.
- *Mavros* is used to establish communication with PX4.

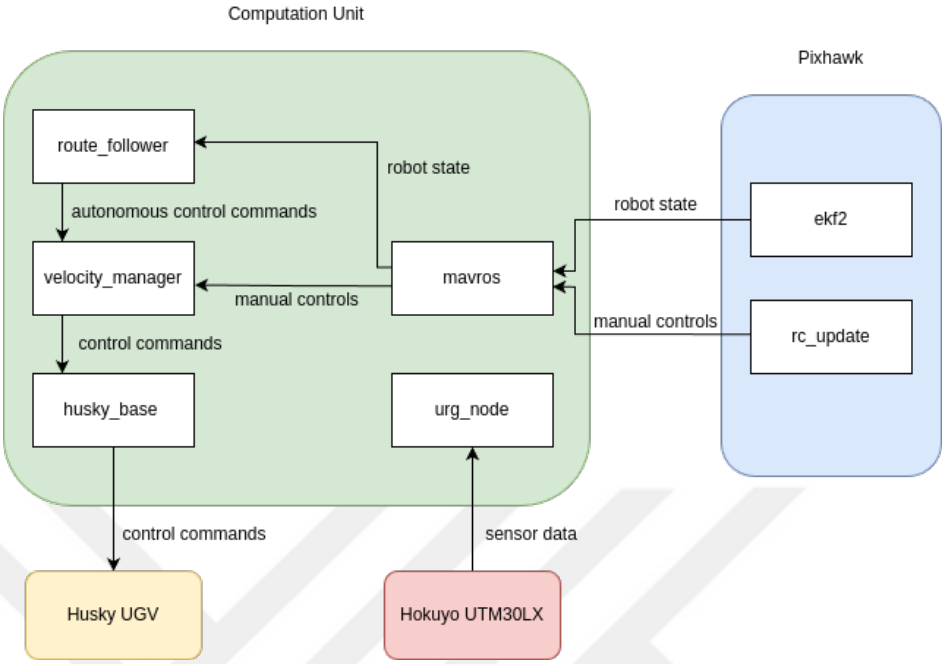


Figure 4.12. MGR software architecture.

A ROS package *drive_control* is written to provide autonomous control commands to the Husky. This package consists of two nodes:

- *velocity_manager*: This node decides whether manual control commands from the RC unit or autonomous commands from the *route_follower* node should be sent to the robot depending on the positions of control switches on the RC unit.
- *route_follower*: This node is responsible for generating autonomous velocity commands. It loads the road network and initializes the position of the robot in the robot network. Then, it waits for a destination request from an external source in the form of node identification (ID) in this road network. The details of the road network representation are explained in Appendix B. Once a destination is requested, it calculates the route to reach the destination using the A^* algorithm. In order to follow the calculated route, it applies the control method explained in Section 4.3.2 until reaching the destination or receiving a new destination request.

5. CONCLUSION

This thesis studies the coordination of an aerial robot (AR) with a mobile ground robot (MGR) for aerial terrain mapping and analysis. This problem is considered from three different aspects. The first studies the problem of rendezvous scheduling between the AR and MGR. Using such a coordination scheme enables the AR to cover large terrain as it collects the 3D terrain data. Following, spatial mapping and localization algorithms that can be used by the AR are developed. Finally, a real coordination system is designed and its hardware and software developments are both initiated.

Chapter 2 has presented a novel rendezvous scheduling approach. It should be remarked that the proposed approach differs conceptually from previous work - as its focus is on scheduling and assumes that the route sequence is predetermined or has been found using one of the available algorithms. Considering this and the navigational constraints of the MGR, the problem is formulated as an optimal control problem and solved using problem-specific properties to reduce computational complexity. The contribution of this approach is that the AR flies through all the given waypoints in the shortest time while ensuring that flight times between consecutive recharges do not exceed the flight range of the AR. The proposed formulation is realistic in the sense that all charging-related times such as landing and take-off times are accounted for. Furthermore, it covers a range of scenarios depending on the requirement of a dock for the landing of the AR and the replacement or recharge of the battery. As the resulting schedule generation problem is relatively simpler, a rendezvous schedule becomes realizable even for large-scale missions with long duration - as shown in the simulations. Additionally, time complexities are analyzed to present the scalability of the proposed methods. The low time complexities and high speeds of the methods allow for them to be used during real-time missions for recovery scenarios where the vehicle parameters change due to environmental effects and a fast new solution is required. Finally, its performance is compared with the optimistic lower bound of the solution for the case of unordered waypoints.

In Chapter 3, the problems of 3D mapping and localization are addressed. For 3D mapping, a method based on the integration of RGB and LIDAR sensor data of ARs is proposed. In this method, a 3D representation of the keypoints in an image is constructed based on interpolating the respective depth data. Two different transformation estimation methods - least squares and random sample consensus (RANSAC) - are used to transform these 3D representations into a common coordinate frame. Simulation-based results suggest better performance with the RANSAC-based method due to its ability to filter out incorrect data matches. Interestingly, performance deteriorates as the density of the visual data increases. This implies that 3D representations initially created using RGB and LIDAR data need to be refined with the new sensor data. Localization methods using RGB data, scene similarity-based ones with interpolation, and geometric relation-based ones are developed. The resulting localization performances showed that, while geometric relation-based ones offer significantly higher accuracy, they may perform very inconsistently due to incorrectly set data relations. The simulation results suggest that hybrid methods can be utilized to benefit from the high accuracy of geometric relation-based methods and the consistency of interpolation-based ones.

Finally, Chapter 4 has presented the design of a coordination system that can be used to realize coordinated AR - MGR missions. For this, a battery replacement platform (BRP) with the capability of horizontal stabilization is presented. A novel battery casing design - easily replaceable battery casing (ERBC) - is introduced as to enable battery replacement with standard robot arms. This design allows displacing and placing the battery with simple movements such as moving the robot arm up and down. Additionally, with the ability to recharge batteries, the BRP design allows extended mission times for the AR with five spare battery chasings. Through the design of a mobile ground robot (MGR) capable of carrying the BRP, a mobile battery replenishment system is achieved. The software architecture for the navigation of the MGR on given road networks is presented.

For future work, the complete system is planned to become operational. It should be noted that the hardware-software proposed and developed in this thesis provide a basis for this work. First, as ERBC is not able to provide electrical connections to the battery, further design improvements will be required. Second, control algorithms for horizontal stabilization of the BRP will be developed. The battery replacement operation will be implemented. Following, field experiments will be conducted. This will first require performing tests with the AR, MGR and BRP one at a time. Following, coordination scenarios will be tested.

REFERENCES

1. Liew, C. F., D. DeLatte, N. Takeishi and T. Yairi, "Recent Developments in Aerial Robotics: A Survey and Prototypes Overview", *arXiv preprint arXiv:1711.10085*, 2017.

2. Dutta, A., S. Roy, O. P. Kreidl and L. Bölöni, "Multi-Robot Information Gathering for Precision Agriculture: Current State, Scope, and Challenges", *IEEE Access*, Vol. 9, pp. 161416–161430, 2021.

3. Choset, H., "Coverage for Robotics–A Survey of Recent Results", *Annals of Mathematics and Artificial Intelligence*, Vol. 31, pp. 113–126, 2001.

4. Galceran, E. and M. Carreras, "A Survey on Coverage Path Planning for Robotics", *Robotics and Autonomous Systems*, Vol. 61, No. 12, pp. 1258–1276, 2013.

5. Shi, G., N. Karapetyan, A. B. Asghar, J.-P. Reddinger, J. Dotterweich, J. Humann and P. Tokekar, "Risk-Aware UAV-UGV Rendezvous with Chance-Constrained Markov Decision Process", *IEEE 61st Conference on Decision and Control*, Cancun, Mexico, pp. 180–187, 2022.

6. Maini, P., K. Yu, P. Sujit and P. Tokekar, "Persistent Monitoring with Refueling on a Terrain Using a Team of Aerial and Ground Robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, pp. 8493–8498, 2018.

7. Maini, P., K. Sundar, M. Singh, S. Rathinam and P. Sujit, "Cooperative Aerial–Ground Vehicle Route Planning with Fuel Constraints for Coverage Applications", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 6, pp. 3016–3028, 2019.

8. Eker, A. H., A. Öncü and H. I. Bozma, "Rendezvous Scheduling for Charging Coordination Between Aerial Robot-Mobile Ground Robot", *IEEE International*

Conference on Automation Science and Engineering, Mexico City, Mexico, pp. 1192–1197, 2022.

9. Alyassi, R., M. Khonji, A. Karapetyan, S. C.-K. Chau, K. Elbassioni and C.-M. Tseng, "Autonomous Recharging and Flight Mission Planning for Battery-Operated Autonomous Drones", *IEEE Transactions on Automation Science and Engineering*, Vol. 20, No. 2, pp. 1034–1046, 2022.

10. Coelho, B. N., V. N. Coelho, I. M. Coelho, L. S. Ochi, R. Haghnazar, D. Zuidema, M. S. Lima and A. R. da Costa, "A Multi-Objective Green UAV Routing Problem", *Computers & Operations Research*, Vol. 88, pp. 306–315, 2017.

11. Levy, D., K. Sundar and S. Rathinam, "Heuristics for Routing Heterogeneous Unmanned Vehicles with Fuel Constraints", *Mathematical Problems in Engineering*, Vol. 2014, No. 1, p. 131450, 2014.

12. Mitchell, D., M. Corah, N. Chakraborty, K. Sycara and N. Michael, "Multi-Robot Long-Term Persistent Coverage with Fuel Constrained Robots", *IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, pp. 1093–1099, 2015.

13. Huang, H. and A. V. Savkin, "A Method of Optimized Deployment of Charging Stations for Drone Delivery", *IEEE Transactions on Transportation Electrification*, Vol. 6, No. 2, pp. 510–518, 2020.

14. Derenick, J., N. Michael and V. Kumar, "Energy-Aware Coverage Control with Docking for Robot Teams", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, pp. 3667–3672, 2011.

15. Trotta, A., M. Di Felice, F. Montori, K. R. Chowdhury and L. Bononi, "Joint Coverage, Connectivity, and Charging Strategies for Distributed UAV Networks", *IEEE Transactions on Robotics*, Vol. 34, No. 4, pp. 883–900, 2018.

16. Zhao, C., J. Liu, M. Sheng, W. Teng, Y. Zheng and J. Li, "Multi-UAV Trajectory Planning for Energy-Efficient Content Coverage: A Decentralized Learning-Based Approach", *IEEE Journal on Selected Areas in Communications*, Vol. 39, No. 10, pp. 3193–3207, 2021.

17. Trotta, A., M. Di Felice, K. R. Chowdhury and L. Bononi, "Fly and Recharge: Achieving Persistent Coverage Using Small Unmanned Aerial Vehicles (SUAVs)", *IEEE International Conference on Communications*, Paris, France, pp. 1–7, 2017.

18. Hari, S. K. K., S. Rathinam, S. Darbha, S. G. Manyam, K. Kalyanam and D. Casbeer, "Bounds on Optimal Revisit Times in Persistent Monitoring Missions with a Distinct and Remote Service Station", *IEEE Transactions on Robotics*, Vol. 39, No. 2, pp. 1070–1086, 2022.

19. Washington, P. H. and M. Schwager, "Reduced State Value Iteration for Multi-Drone Persistent Surveillance with Charging Constraints", *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Prague, Czech Republic, pp. 6390–6397, 2021.

20. Hari, S. K. K., S. Rathinam, S. Darbha, K. Kalyanam, S. G. Manyam and D. Casbeer, "Optimal UAV Route Planning for Persistent Monitoring Missions", *IEEE Transactions on Robotics*, Vol. 37, No. 2, pp. 550–566, 2020.

21. Mitchell, D., E. A. Cappo and N. Michael, "Persistent Robot Formation Flight via Online Substitution", *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea (South), pp. 4810–4815, 2016.

22. Kilic, K. I. and L. Mostarda, "Heuristic Drone Pathfinding Over Optimized Charging Station Grid", *IEEE Access*, Vol. 9, pp. 164070–164089, 2021.

23. Liu, L. and N. Michael, "Energy-Aware Aerial Vehicle Deployment via Pipartite Graph Matching", *International Conference on Unmanned Aircraft Systems*, Or-

lando, FL, USA, pp. 189–194, 2014.

24. Santin, R., L. Assis, A. Vivas and L. C. Pimenta, "Matheuristics for Multi-UAV Routing and Recharge Station Location for Complete Area Coverage", *Sensors*, Vol. 21, No. 5, p. 1705, 2021.

25. Huang, H. and A. V. Savkin, "Optimal Deployment of Charging Stations for Aerial Surveillance by UAVs with the Assistance of Public Transportation Vehicles", *Sensors*, Vol. 21, No. 16, p. 5320, 2021.

26. Ribeiro, R. G., J. R. Júnior, L. P. Cota, T. A. Euzébio and F. G. Guimaraes, "Unmanned Aerial Vehicle Location Routing Problem with Charging Stations for Belt Conveyor Inspection System in the Mining Industry", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 21, No. 10, pp. 4186–4195, 2019.

27. Sundar, K. and S. Rathinam, "Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots", *IEEE Transactions on Automation Science and Engineering*, Vol. 11, No. 1, pp. 287–294, 2013.

28. Sundar, K., S. Venkatachalam and S. Rathinam, "Formulations and Algorithms for the Multiple Depot, Fuel-Constrained, Multiple Vehicle Routing Problem", *American Control Conference*, Boston, MA, USA, pp. 6489–6494, 2016.

29. Kannon, T. E., S. G. Nurre, B. J. Lunday and R. R. Hill, "The Aircraft Routing Problem with Refueling", *Optimization Letters*, Vol. 9, pp. 1609–1624, 2015.

30. Maini, P. and P. Sujit, "On Cooperation Between a Fuel Constrained UAV and a Refueling UGV for Large Scale Mapping Applications", *International Conference on Unmanned Aircraft Systems*, Denver, CO, USA, pp. 1370–1377, 2015.

31. Yu, K., A. K. Budhiraja, S. Buebel and P. Tokekar, "Algorithms and Experiments on Routing of Unmanned Aerial Vehicles with Mobile Recharging Stations", *Journal of Field Robotics*, Vol. 36, No. 3, pp. 602–616, 2019.

32. Mathew, N., S. L. Smith and S. L. Waslander, "Multirobot Rendezvous Planning for Recharging in Persistent Tasks", *IEEE Transactions on Robotics*, Vol. 31, No. 1, pp. 128–142, 2015.

33. Yu, K., A. K. Budhiraja and P. Tokekar, "Algorithms for Routing of Unmanned Aerial Vehicles with Mobile Recharging Stations", *IEEE International Conference on Robotics and Automation*, Brisbane, QLD, Australia, pp. 5720–5725, 2018.

34. Litus, Y., P. Zebrowski and R. T. Vaughan, "A Distributed Heuristic for Energy-Efficient Multirobot Multiplace Rendezvous", *IEEE Transactions on Robotics*, Vol. 25, No. 1, pp. 130–135, 2008.

35. Asghar, A. B., G. Shi, N. Karapetyan, J. Humann, J.-P. Reddinger, J. Dotterweich and P. Tokekar, "Risk-Aware Recharging Rendezvous for a Collaborative Team of UAVs and UGVs", *IEEE International Conference on Robotics and Automation*, London, United Kingdom, pp. 5544–5550, 2023.

36. Seyedi, S., Y. Yazicioğlu and D. Aksaray, "Persistent Surveillance with Energy-Constrained UAVs and Mobile Charging Stations", *IFAC-PapersOnLine*, Vol. 52, No. 20, pp. 193–198, 2019.

37. Zhang, P., S. Xu, W. Zhang and W. Dong, "A Cooperative Aerial Inspection System with Continuable Charging Strategy", *IEEE International Conference on Robotics and Biomimetics*, Dali, China, pp. 770–777, 2019.

38. Lin, X., Y. Yazıcıoğlu and D. Aksaray, "Robust Planning for Persistent Surveillance with Energy-Constrained UAVs and Mobile Charging Stations", *IEEE Robotics and Automation Letters*, Vol. 7, No. 2, pp. 4157–4164, 2022.

39. Du, B., D. Sun, S. G. Manyam and D. W. Casbeer, "Cooperative Air-Ground Vehicle Routing Using Chance-Constrained Optimization", *American Control Conference*, Denver, CO, USA, pp. 392–397, 2020.

40. Ribeiro, R. G., L. P. Cota, T. A. Euzébio, J. A. Ramírez and F. G. Guimarães, "Unmanned-Aerial-Vehicle Routing Problem with Mobile Charging Stations for Assisting Search and Rescue Missions in Postdisaster Scenarios", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 52, No. 11, pp. 6682–6696, 2021.

41. Zhang, M., H. Liang and P. Zhou, "Cooperative Route Planning for Fuel-constrained UGV-UAV Exploration", *IEEE International Conference on Unmanned Systems*, Guangzhou, China, pp. 1047–1052, 2022.

42. Liu, Y., Z. Luo, Z. Liu, J. Shi and G. Cheng, "Cooperative Routing Problem for Ground Vehicle and Unmanned Aerial Vehicle: The Application on Intelligence, Surveillance, and Reconnaissance Missions", *IEEE Access*, Vol. 7, pp. 63504–63518, 2019.

43. Ramasamy, S., J.-P. F. Reddinger, J. M. Dotterweich, M. A. Childers and P. A. Bhounsule, "Coordinated Route Planning of Multiple Fuel-Constrained Unmanned Aerial Systems with Recharging on an Unmanned Ground Vehicle for Mission Coverage", *Journal of Intelligent & Robotic Systems*, Vol. 106, No. 1, p. 30, 2022.

44. Arora, D., P. Maini, P. Pinacho-Davidson and C. Blum, "Route Planning for Cooperative Air-Ground Robots with Fuel Constraints: An Approach Based on CMSA", *Proceedings of the Genetic and Evolutionary Computation Conference*, Prague, Czech Republic, pp. 207–214, 2019.

45. Cormen, T. H., C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, pp. 558–570, The MIT Press, 1990.

46. Cabreira, T. M., L. B. Brisolara and F. J. Paulo R, "Survey on Coverage Path Planning with Unmanned Aerial Vehicles", *Drones*, Vol. 3, No. 1, p. 4, 2019.

47. Campos, C., R. Elvira, J. J. G. Rodríguez, J. M. Montiel and J. D. Tardós, "Orb-

SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM", *IEEE Transactions on Robotics*, Vol. 37, No. 6, pp. 1874–1890, 2021.

48. Lowe, D. G., "Distinctive Image Features From Scale-Invariant Keypoints", *International Journal of Computer Vision*, Vol. 60, pp. 91–110, 2004.

49. Bay, H., A. Ess, T. Tuytelaars and L. Van Gool, "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346–359, 2008.

50. Rublee, E., V. Rabaud, K. Konolige and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF", *2011 International Conference on Computer Vision*, Barcelona, Spain, pp. 2564–2571, 2011.

51. Botterill, T., S. Mills and R. Green, "Real-Time Aerial Image Mosaicing", *2010 25th International Conference of Image and Vision Computing New Zealand*, Queenstown, New Zealand, pp. 1–8, 2010.

52. Marburg, A. and M. P. Hayes, "SMARTPIG: Simultaneous Mosaicking and Resectioning Through Planar Image Graphs", *2015 IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, pp. 5767–5774, 2015.

53. Chen, Y., Y. Chen and G. Wang, "Bundle Adjustment Revisited", *arXiv preprint arXiv:1912.03858*, 2019.

54. Li, Z. and V. Isler, "Large Scale Image Mosaic Construction for Agricultural Applications", *IEEE Robotics and Automation Letters*, Vol. 1, No. 1, pp. 295–302, 2016.

55. Fathian, K., K. Khosoussi, Y. Tian, P. Lusk and J. P. How, "CLEAR: A Consistent Lifting, Embedding, and Alignment Rectification Algorithm for Multiview Data Association", *IEEE Transactions on Robotics*, Vol. 36, No. 6, pp. 1686–1703, 2020.

56. Wen, W., T. Pfeifer, X. Bai and L.-T. Hsu, "It is Time for Factor Graph Optimization for GNSS/INS Integration: Comparison between FGO and EKF", *arXiv preprint arXiv:2004.10572*, 2020.

57. Wikipedia, "Real-Time Kinematic Positioning", `https://en.wikipedia.org/wiki/Real-time_kinematic_positioning`, accessed in July 2024.

58. Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

59. Huamán, A., "OpenCV: Feature Matching with FLANN", `https://docs.opencv.org/4.x/d5/d6f/tutorial_feature_flann_matcher.html`, accessed in July 2024.

60. Fischler, M. A. and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, Vol. 24, No. 6, pp. 381–395, 1981.

61. Rusu, R. B. and S. Cousins, "3D is Here: Point Cloud Library (PCL)", *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9-13 2011.

62. Trempe, P., "Spherical K-Nearest Neighbors Interpolation", *arXiv preprint arXiv:1910.00704*, 2019.

63. Palitri, "Closest Point Between Two Rays", `https://palitri.com/vault/stuff/maths/Rays%20closest%20point.pdf`, accessed in July 2024.

64. Zeng, G., S. Chen, B. Mu, G. Shi and J. Wu, "CPnP: Consistent Pose Estimator for Perspective-n-Point Problem with Bias Elimination", *2023 IEEE International Conference on Robotics and Automation*, London, United Kingdom, pp. 1940–1946, 2023.

65. OpenCV, "Perspective-n-Point (PnP) Pose Computation", https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html, accessed in July 2024.

66. Terzakis, G. and M. Lourakis, "A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem", *16th European Conference on Computer Vision, Part I 16*, Glasgow, United Kingdom, pp. 478–494, 2020.

67. Bibb, C., "Northumberlandia Land Sculpture", https://sketchfab.com/3d-models/northumberlandia-land-sculpture-1beab497b6924e06adff8b407991f2ed, accessed in July 2024.

68. Pointclouds, "Point Cloud Library (PCL): pcl::registration::TransformationEstimation; Class Template Reference", https://pointclouds.org/documentation/classpcl_1_1registration_1_1_transformation_estimation.html, accessed in July 2024.

69. DJI, "Matrice 350 RTK", https://enterprise.dji.com/matrice-350-rtk, accessed in July 2024.

70. Suzuki, K. A., P. Kemper Filho and J. R. Morrison, "Automatic Battery Replacement System for UAVs: Analysis and Design", *Journal of Intelligent & Robotic Systems*, Vol. 65, pp. 563–586, 2012.

71. Fujii, K., K. Higuchi and J. Rekimoto, "Endless Flyer: A Continuous Flying Drone with Automatic Battery Replacement", *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Vietri sul Mare, Italy, pp. 216–223, 2013.

72. Herath, H. M. C. W. B., H. M. S. Herath, S. W. Sumangala, O. de Silva, D. Chathuranga and T. D. Lalitharatne, "Design and Development of an Automated Battery Swapping and Charging Station for Multirotor Aerial Vehicles", *2017 17th Inter-*

national Conference on Control, Automation and Systems, Jeju, Korea (South), pp. 356–361, 2017.

73. Kuka, "KUKA OmniMove (AGVs)", `https://www.kuka.com/en-de/products/amr-autonomous-mobile-robotics/mobile-platforms/kuka-omnimove`, accessed in July 2024.

74. U-blox, "Moving Base Applications", `https://content.u-blox.com/sites/default/files/documents/ZED-F9P-MovingBase_AppNote_UBX-19009093.pdf`, accessed in July 2024.

75. Shen, W., T. T. Vo and A. Kapoor, "Charging Algorithms of Lithium-Ion Batteries: An Overview", *2012 7th IEEE Conference on Industrial Electronics and Applications*, Singapore, pp. 1567–1572, 2012.

76. ClearpathRobotics, "Husky UGV - Outdoor Field Research Robot", `https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/`, accessed in July 2024.

77. Meier, L., D. Honegger and M. Pollefeys, "PX4: A Node-Based Multithreaded Open Source Robotics Framework for Deeply Embedded Platforms", *2015 IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, pp. 6235–6240, 2015.

78. Meier, L., P. Tanskanen, F. Fraundorfer and M. Pollefeys, "Pixhawk: A System for Autonomous Flight Using Onboard Computer Vision", *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 2992–2997, 2011.

79. Coulter, R. C., "Implementation of the Pure Pursuit Path Tracking Algorithm", *Technical Report, Carnegie Mellon University*, 1992.

80. ClearpathRobotics, "GitHub - clearpathrobotics/ros_computer_setup", `https://github.com/clearpathrobotics/ros_computer_setup/tree/main`, accessed in

July 2024.

81. Liu, C., J. Zhao and N. Sun, "A Review of Collaborative Air-Ground Robots Research", *Journal of Intelligent & Robotic Systems*, Vol. 106, No. 3, p. 60, 2022.

82. Zou, B., X. Xu, Y. Y. Gong and R. De Koster, "Evaluating Battery Charging and Swapping Strategies in a Robotic Mobile Fulfillment System", *European Journal of Operational Research*, Vol. 267, No. 2, pp. 733–753, 2018.

APPENDIX A: PROOFS OF PROPOSITIONS

The following proposition gives an equivalent expression for $J(x^0, \mathcal{C})$.

Proposition 2.1:

$$J(x^0, \mathcal{C}) = \frac{d_s(P_1, P_N)}{\gamma_A} + T_{x^0} + \sum_{i=1}^{N'} T'_{x^{i-1}x^i} \qquad \text{where}$$

$$T'_{x'x} = T_{x'x} - \frac{d(P_{x_a}, P_{x_a+1})}{\gamma_A}, \qquad T'_{x'x^{N'}} = T_{x'x^{N'}}.$$

Proof: Consider an ordered set $\mathcal{C}$ of charge selections. By definition

$$\begin{aligned} J(x^0, \mathcal{C}) =& T_{x^0} + \sum_{i=1}^{N'} \left(\frac{d_s(P_{x_a^{i-1}+1}, P_{x_a^i})}{\gamma_A} + T_{x^{i-1}x^i} \right) \\ =& T'_{x^0} + T'_{x^{(N'-1)}x^{N'}} + \frac{d_s(P_{x_a^{(N'-1)}+1}, P_{x_a^{N'}})}{\gamma_A} + \sum_{i=1}^{N'-1} \left(\frac{d_s(P_{x_a^{i-1}+1}, P_{x_a^i})}{\gamma_A} \right. \\ & \left. + \frac{d_s(P_{x_a^i}, P_{x_a^i+1})}{\gamma_A} - \frac{d_s(P_{x_a^i}, P_{x_a^i+1})}{\gamma_A} + T_{x^{i-1}x^i} \right) \\ =& T'_{x^0} + \sum_{i=1}^{N'} \left(T'_{x^{i-1}x^i} \right) + \frac{d_s(P_{x_a^{(N'-1)}+1}, P_{x_a^{N'}})}{\gamma_A} + \sum_{i=1}^{N'-1} \left(\frac{d_s(P_{x_a^{i-1}+1}, P_{x_a^i+1})}{\gamma_A} \right). \end{aligned}$$

Since $P_{x_a^0+1} = P_1$ and $P_{x_a^{N'}} = P_N$, by definition

$$\frac{d_s(P_{x_a^{(N'-1)}+1}, P_{x_a^{N'}})}{\gamma_A} + \sum_{i=1}^{N'-1} \left(\frac{d_s(P_{x_a^{i-1}+1}, P_{x_a^i+1})}{\gamma_A} \right) = \frac{d_s(P_{x_a^0+1}, P_{x_a^{N'}})}{\gamma_A} = \frac{d_s(P_1, P_N)}{\gamma_A}.$$

$$\text{Thus, } J(x^0, \mathcal{C}) = \frac{d_s(P_1, P_N)}{\gamma_A} + T_{x^0} + \sum_{i=1}^{N'} T'_{x^{i-1}x^i} \square$$

Proposition 2.2: For each non-final charge selection CS_x, if $l > p(x)$, then $CS_{v(l)} \notin \mathcal{R}(CS_x)$.

Proof: By definition, if $v(l)_a \leq x_a$, then $CS_{v(l)} \notin \mathcal{R}(CS_x)$. Consider $\forall l > p(x)$ such that $v(l)_a > x_a$. Then, from the definition of $p(x)$

$$R_x + \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} + G_{v(l)} + \frac{d_s(P_1, P_{v(l)_a})}{\gamma_A} > \eta_A - T_\uparrow - T_\downarrow + \frac{d_s(P_1, P_N)}{\gamma_A}.$$

Rewriting $d_s(P_1, P_{v(l)_a})$ as $d_s(P_1, P_{x_a+1}) + d_s(P_{x_a+1}, P_{v(l)_a})$,

$$R_x + \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} + G_{v(l)} + \frac{d_s(P_1, P_{x_a+1})}{\gamma_A} + \frac{d_s(P_{x_a+1}, P_{v(l)_a})}{\gamma_A} > \eta_A - T_\uparrow - T_\downarrow + \frac{d_s(P_1, P_N)}{\gamma_A}.$$

Noting that $d_s(P_1, P_{x_a+1}) + d_s(P_{x_a+1}, P_N) = d_s(P_1, P_N)$,

$$R_x + \frac{d_s(P_{x_a+1}, P_{v(l)_a})}{\gamma_A} + G_{v(l)} + \frac{d_s(P_1, P_N)}{\gamma_A} > \eta_A - T_\uparrow - T_\downarrow + \frac{d_s(P_1, P_N)}{\gamma_A}.$$

Arranging terms,

$$R_x + \frac{d_s(P_{x_a+1}, P_{v(l)_a})}{\gamma_A} + G_{v(l)} + T_\uparrow + T_\downarrow > \eta_A.$$

As this implies $\alpha(x, v(l)) > \eta_A$, the range conditions for both no hovering and hovering required cases are violated. Hence, a contradiction. □

Proposition 2.3: $\forall k, k' \in \mathcal{O}_\rightarrow$, if $k < k'$, then $p(z(k)) \leq p(z(k'))$.

Proof: Recall that the order of $\mathcal{O}_\rightarrow$ is such that

$$R_{z(k')} + \frac{d_s(P_{z(k')_a+1}, P_N)}{\gamma_A} \leq R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A}.$$

$$\text{Thus, } \eta'_A - R_{z(k)} - \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A} \leq \eta'_A - R_{z(k')} - \frac{d_s(P_{z(k')_a+1}, P_N)}{\gamma_A}.$$

From the definition of $p(z(k))$ and $p(z(k'))$,

$$G_{v(p(z(k)))} + \frac{d_s(P_1, P_{v(p(z(k)))_a})}{\gamma_A} \leq \eta'_A - R_{z(k)} - \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A},$$

$$G_{v(p(z(k')))} + \frac{d_s(P_1, P_{v(p(z(k')))_a})}{\gamma_A} \leq \eta'_A - R_{z(k')} - \frac{d_s(P_{z(k')_a+1}, P_N)}{\gamma_A}.$$

Since $p(z(k))$ and $p(z(k'))$ are found using sup and from the inequalities above,

$$G_{v(p(z(k)))} + \frac{d_s(P_1, P_{v(p(z(k)))_a})}{\gamma_A} \leq G_{v(p(z(k')))} + \frac{d_s(P_1, P_{v(p(z(k')))_a})}{\gamma_A}.$$

Using the ordering of $\mathcal{O}_\leftarrow$, it follows that $p(z(k)) \leq p(z(k'))$. □

Proposition 2.4: Consider $k \in \mathcal{O}_\rightarrow$ and the associated charge selection $CS_{z(k)}$. If the best successor $CS_{z(k)^*}$ of the charge selection $CS_{z(k)}$ is not a final charge selection, $CS_{z(k^*)}$, then $k^* > k$.

Proof: Let $CS_{z(k)^*} = CS_{z(k^*)}$ and suppose $k^* < k$. Similarly, let $CS_{z(k^*)^*}$ be the best successor of $CS_{z(k^*)}$. By definition, $z(k^*)^*_a > z(k^*)_a > z(k)_a$. Additionally, as

$CS_{z(k^*)^*} \in \mathcal{R}(CS_{z(k^*)})$,

$$T_\uparrow + R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)^*} + T_\downarrow \leq \eta_A,$$

$$R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_N)}{\gamma_A} - \frac{d_s(P_{z(k^*)^*_a}, P_N)}{\gamma_A} \leq \eta_A - T_\uparrow - G_{z(k^*)^*} - T_\downarrow,$$

$$R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_N)}{\gamma_A} \leq \eta_A - T_\uparrow - G_{z(k^*)^*} - T_\downarrow + \frac{d_s(P_{z(k^*)^*_a}, P_N)}{\gamma_A}.$$

From the definition of $\mathcal{O}_\rightarrow$,

$$R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_N)}{\gamma_A} \geq R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A}.$$

$$\text{Thus, } R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A} \leq \eta_A - T_\uparrow - G_{z(k^*)^*} - T_\downarrow + \frac{d_s(P_{z(k^*)^*_a}, P_N)}{\gamma_A},$$

$$R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_N)}{\gamma_A} - \frac{d_s(P_{z(k^*)^*_a}, P_N)}{\gamma_A} \leq \eta_A - T_\uparrow - G_{z(k^*)^*} - T_\downarrow,$$

$$T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)^*} + T_\downarrow \leq \eta_A.$$

Implying that $CS_{z(k^*)^*} \in R(CS_{z(k)})$. Now suppose that after recharging at $CS_{z(k)}$, $CS_{z(k^*)^*}$ is selected as the successor charge selection. Let $\mathcal{C}'_{z(k)}$ be the associated charge selections up to mission completion. Also, for the time being, let $CS_{z(k^*)^*}$ be an intermediate charge selection. Then,

$$\begin{aligned} J(z(k), \mathcal{C}'_{z(k)}) =& J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T'_{z(k)z(k^*)^*} \\ =& J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T_\uparrow + T_\downarrow + R_{z(k^*)^*} + G_{z(k^*)^*} - \frac{d_s(P_{z(k^*)^*_a}, P_{z(k^*)^*_a+1})}{\gamma_A} \\ &+ E_{z(k),z(k^*)^*} + \max\left\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} \right. \\ &\left. + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)^*} + T_\downarrow\Big)\right\}. \end{aligned} \tag{A.1}$$

For $o, p, q \in \mathbb{R}$ and $q \geq 0$, note the inequality $o + \max\{0, p\} \leq o + q + \max\{0, p - q\}$. Also, from the triangle inequality, note

$$R_{z(k^*)} + G_{z(k^*)} - \frac{d_s(P_{z(k^*)_a}, P_{z(k^*)_a+1})}{\gamma_A} \geq 0.$$

The terms on the RHS of Eq. A.1 are rearranged accordingly

$$\begin{aligned} J(z(k), \mathcal{C}'_{z(k)}) \leq& J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T_\uparrow + T_\downarrow + R_{z(k^*)^*} + G_{z(k^*)^*} - \frac{d_s(P_{z(k^*)^*_a}, P_{z(k^*)^*_a+1})}{\gamma_A} \\ &+ E_{z(k),z(k^*)^*} + T_\uparrow + T_\downarrow + R_{z(k^*)} + G_{z(k^*)} - \frac{d_s(P_{z(k^*)_a}, P_{z(k^*)_a+1})}{\gamma_A} \\ &+ \max\left\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)^*_a})}{(\gamma_A)}\right. \end{aligned}$$

$$+ G_{z(k^*)^*} + T_\downarrow + T_\uparrow + R_{z(k^*)} - \frac{d_s(P_{z(k^*)_a}, P_{z(k^*)_a+1})}{\gamma_A} + G_{z(k^*)} + T_\downarrow\Big)\Big\}.$$

Additionally, note that

$$\begin{aligned}
&\max\Big\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)^*_a})}{(\gamma_A)} + G_{z(k^*)^*} + T_\downarrow + T_\uparrow \\
&+ R_{z(k^*)} - \frac{d_s(P_{z(k^*)_a}, P_{z(k^*)_a+1})}{\gamma_A} + G_{z(k^*)} + T_\downarrow\Big)\Big\} \\
&\leq \max\Big\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)_b})}{\gamma_G} + \frac{d_s(V_{z(k^*)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)_a})}{(\gamma_A)} \\
&+ G_{z(k^*)^*} + T_\downarrow + T_\uparrow + R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)} + T_\downarrow\Big)\Big\} \\
&\leq \max\Big\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)_a})}{\gamma_A} + G_{z(k^*)} + T_\downarrow\Big)\Big\} \\
&+ \max\Big\{0, \frac{d_s(V_{z(k^*)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k^*)} + \frac{d_s(P_{z(k^*)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)^*} + T_\downarrow\Big)\Big\}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
J(z(k), \mathcal{C}'_{z(k)}) \leq & J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T_\uparrow + T_\downarrow + R_{z(k^*)^*} + G_{z(k^*)^*} - \frac{d_s(P_{z(k^*)^*_a}, P_{z(k^*)^*_a+1})}{\gamma_A} \\
& + E_{z(k),z(k^*)^*} + T_\uparrow + T_\downarrow + R_{z(k^*)} + G_{z(k^*)} - \frac{d_s(P_{z(k^*)_a}, P_{z(k^*)_a+1})}{\gamma_A} \\
& + \max\Big\{0, \frac{d_s(V_{z(k)_b}, V_{z(k^*)_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k)} + \frac{d_s(P_{z(k)_a+1}, P_{z(k^*)_a})}{\gamma_A} \\
& + G_{z(k^*)} + T_\downarrow\Big)\Big\} + \max\Big\{0, \frac{d_s(V_{z(k^*)_b}, V_{z(k^*)^*_b})}{\gamma_G} - \Big(T_\uparrow + R_{z(k^*)} \\
& + \frac{d_s(P_{z(k^*)_a+1}, P_{z(k^*)^*_a})}{\gamma_A} + G_{z(k^*)^*} + T_\downarrow\Big)\Big\} \\
= & J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T'_{z(k)z(k^*)} + T'_{z(k^*)z(k^*)^*} - E_{z(k^*)z(k^*)^*} \\
\leq & J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T'_{z(k)z(k^*)} + T'_{z(k^*)z(k^*)^*}. \qquad \text{(A.2)}
\end{aligned}$$

Now consider

$$J^*(z(k), \mathcal{C}_{z(k)}) = J^*(z(k^*), \mathcal{C}_{z(k^*)}) + T'_{z(k)z(k^*)}. \qquad \text{(A.3)}$$

Note that

$$J^*(z(k^*), \mathcal{C}_{z(k^*)}) = J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T'_{z(k^*)z(k^*)^*}. \qquad \text{(A.4)}$$

Substituting RHS of Eq. A.4 in Eq. A.3,

$$J^*(z(k), \mathcal{C}_{z(k)}) = J^*(z(k^*)^*, \mathcal{C}_{z(k^*)^*}) + T'_{z(k)z(k^*)} + T'_{z(k^*)z(k^*)^*}.$$

From Eq. A.2, this is the upper bound of $J(z(k), \mathcal{C}'_{z(k)})$ and it follows that

$$J(z(k), \mathcal{C}'_{z(k)}) \leq J^*(z(k), \mathcal{C}_{z(k)}).$$

However, this contradicts with the definition of $J^*(z(k), \mathcal{C}_{z(k)})$. $CS_{z(k^*)^*}$ is no worse successor than $CS_{z(k^*)}$ to $CS_{z(k)}$. Therefore, $CS_{z(k^*)}$ with $k^* < k$ can not be the best successor of $CS_{z(k)}$. The same can be shown for final a charge selection $CS_{z(k^*)^*}$ with arranging $T'_{z(k)z(k^*)^*}$ and $T'_{z(k^*)z(k^*)^*}$ accordingly. □

Proposition 2.5: $\forall x, x' \in \begin{bmatrix} x_a^0 & x_b^0 \end{bmatrix}^T \cup \Big(\mathcal{A} - \{N\}\Big) \times \mathcal{B}$, if $x_b = x'_b$ and $x_a < x'_a$, then $p(x) \leq p(x')$.

Proof: If $x_a < x'_a$,

$$\begin{aligned} R_x + \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} &= R_x + \frac{d_s(P_{x_a+1}, P_{x'_a+1})}{\gamma_A} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A} \\ &= \frac{d(V_{x_b}, P_{x_a+1})}{\gamma_A} + \frac{d_s(P_{x_a+1}, P_{x'_a+1})}{\gamma_A} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A}. \end{aligned}$$

Now consider

$$\begin{aligned} R_{x'} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A} &= \frac{d(V_{x'_b}, P_{x'_a+1})}{\gamma_A} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A} \\ &= \frac{d(V_{x_b}, P_{x'_a+1})}{\gamma_A} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A}. \end{aligned}$$

From triangle inequality,

$$\frac{d(V_{x_b}, P_{x'_a+1})}{\gamma_A} \leq \frac{d(V_{x_b}, P_{x_a+1})}{\gamma_A} + \frac{d_s(P_{x_a+1}, P_{x'_a+1})}{\gamma_A}.$$

Thus,

$$\begin{aligned} R_{x'} + \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A} &\leq R_x + \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} \text{ and} \\ \eta'_A - R_x - \frac{d_s(P_{x_a+1}, P_N)}{\gamma_A} &\leq \eta'_A - R_{x'} - \frac{d_s(P_{x'_a+1}, P_N)}{\gamma_A}. \end{aligned}$$

It follows that $p(x) \leq p(x')$. □

APPENDIX B: FINDING A RENDEZVOUS SCHEDULE

The rendezvous scheduling algorithm is implemented as a CMake project using C++ as a shared library named *optimal_rendezvous_scheduling*. Thus, once it is built, it can be included as a library in other projects. The project consists of the following files: *rendezvous_scheduling.h*, *rendezvous_scheduling.cpp*, *BST.cpp*, *example_optimal_mission.cpp*, and *CMakeLists.txt*. *rendezvous_scheduling.h* contains definitions of the classes and necessary methods to implement the algorithms presented in Chapter 2. *rendezvous_scheduling.cpp* contains implementations of the methods in the *rendezvous_scheduling.h* file. *BST.cpp* includes an implementation of the binary search tree to be used in the rendezvous scheduling algorithm. *example_optimal_mission.cpp* presents an example usage of the library. It is intended as an example and is not required for the library. *CMakeLists.txt* defines the CMake rules for building the library.

To run the *example_optimal_mission.cpp*, 3 input files are required: *parameters.txt*, *graph.txt*, and *waypoints.txt*.

parameters.txt contains the parameters required by the algorithm. Each parameter is written on a line without whitespaces, and there are 11 parameters for the algorithm to work. Line by line, these are: 1: the initial node MGR in the graph, integer 2: γ_A, float 3: γ_G, float 4: η_A, float 5: $T_\uparrow$, float 6: $T_\downarrow$, float 7: E, constant recharge time, float 8: λ_A, slope of the recharge function, float 9: Dock and muling parameter, integer. If 0, then the dockless landing algorithm is applied. If 1, then the docked landing algorithm is applied. If 2, then the docked landing algorithm with muling is applied. 10: Final charge station type, integer. If 0, the radiating scenario is used, and all charge stations can be included in the final charge selection. If 1, the circulating scenario is used, and only the initial charge station can be included in the final charge selections. If 2, the segment scenario is used. The charge stations in line 11 can be included in the final charge selections. 11: IDs of the charge station that can be included in the final charge selections for segment scenario, space-separated integer

array.

graph.txt contains the road network that the MGR can navigate on as a graph. Here, each line represents a node and must have the following format: ID of the node as an integer, whitespace, x coordinate of the position represented by the node as a float, whitespace, y coordinate of the position represented by the node as a float, white space, IDs of the neighbor nodes in the graph as a space-separated integer array. The node in the first line of the file must have the ID 0, and the IDs of the following nodes must increase in order.

waypoints.txt contains the waypoints to be visited by the AR in order. Each line must have the following format: x coordinate of the waypoint as a float, whitespace, y coordinate of the waypoint as a float, white space, z coordinate of the waypoint as a float.

To build the project, create a directory *build* under the directory where the project files are. Then open a terminal in the build directory and type the following commands: *cmake ...* This creates the necessary make files to compile the project. Then, type *cd ..* to move to the project directory. Here, typing *make* command will build the project.

To use the library, include it in your project as *#include "rendezvous_scheduling.h"*. Then create an object within the class *RendezvousScheduling*, e.g., *RendezvousScheduling example_mission;*. Then, read the *parameters.txt*, *graph.txt*, and *waypoints.txt* files by calling the methods *example_mission.ReadParametersFromFile(param_file); example_mission.ReadGraphFromFile(graph_file);* and *example_mission.ReadWaypointsFromFile(waypoint_file);*. Here, *param_file*, *graph_file*, and *waypoint_file* are the strings containing directories and file names of the respective files. Then, the schedule can be calculated by calling the method *example_mission.FindSchedule();*. Charge selections in the solution can be found in *example_mission.C* array.

APPENDIX C: MECHANICAL DESIGNS

The design of the coordinating robots requires the design of various mechanical parts. These designs are done using SolidWorks software. All the parts are printed then using a 3D printer, Bambulab X1 Carbon using Polylactic Acid (PLA) material. The parts that are used in the assembly of MGR are listed in Table C.1. The parts that are used in the assembly of BRP are listed in Table C.2. The design details of the individual parts are given in Figure C.1-Figure C.21.

Table C.1. Parts used in the assembly of MGR.

Part name	Function
ethernet_switch_hp_adapter	Mounting the ethernet switch to the MGR electronics plate.
gps_f9p_adapter	Mounting the GNNS receiver to the MGR electronics plate.
hokuyo_utm30lx_adapter	Mounting the Hokuyo UTM30lx laser scanner to the top plate of the MGR.
jetson_agx_orin_adapter	Mounting the AGX Orin to the MGR electronics plate.
pixhawk_6x_adapter	Mounting the Pixhawk 6X to the MGR electronics plate.
siyi_air_unit_adapter	Mounting the SIYI Air Unit to the MGR electronics plate.
siyi_ip_camera_adapter	Mounting the SIYI IP camera to the top plate of the MGR.
siyi_antenna_adapter	Mounting the SIYI antenna pair to the top plate of the MGR.

Table C.2. Parts used in the assembly of BRP.

Part name	Function
arm_aruco	Providing a flat surface on the end effector of the robot arm for placing a visual marker.
ball_joint_platform	Mounting the ball joint to the platform.
ball_joint_linear_motor	Mounting the ball joint to the linear motor.
case_bottom	Holding a case middle.
case_latchbolt	Locking the case_bottom - case_middle connection.
case_lid	Covering the top of the battery in case_middle.
case_middle	Holding a battery.
case_top	Locking - unlocking case_bottom and case_middle connection. Providing an interface for the robot arm to place and displace the battery.
ethernet_switch_tnet_adapter	Mounting the ethernet switch to the BRP electronics plate.
fan_adapter	Mounting a fan to the BRP electronics plate.
linear_motor_hinge_joint	Mounting the hinge joint to the linear motor.
hinge_joint_linear_motor	Mounting the hinge joint to the linear motor.
mpu6050_adapter	Mounting the MPU6050 to the BRP electronics plate.
XT60E_M_adapter	Mounting fifteen XT60E_M connectors to the BRP connector bay.

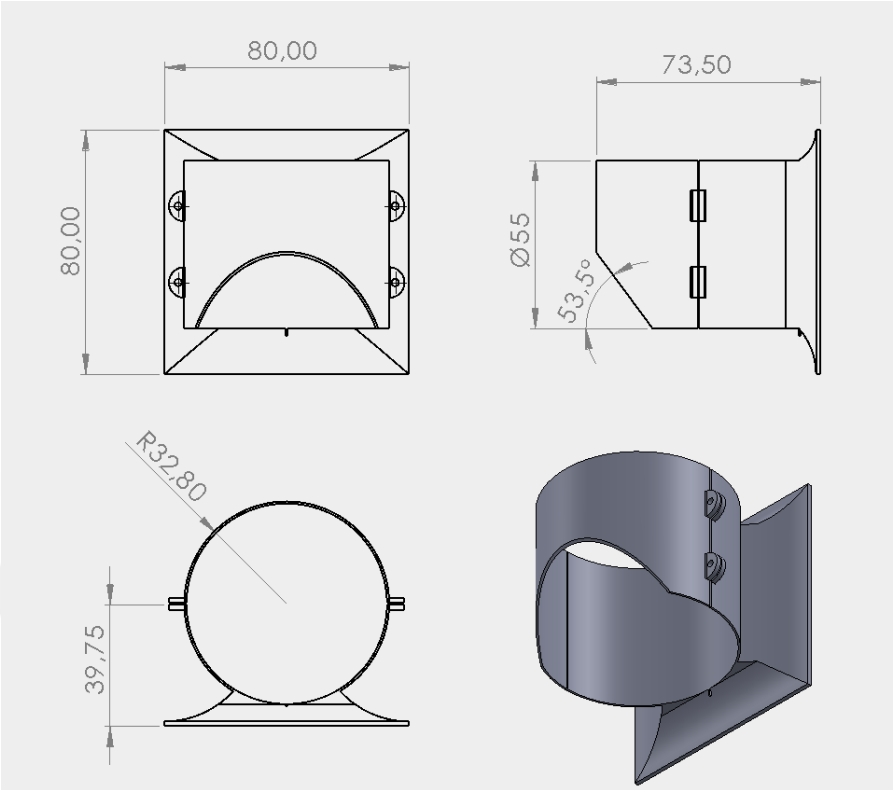


Figure C.1. arm_aruco.

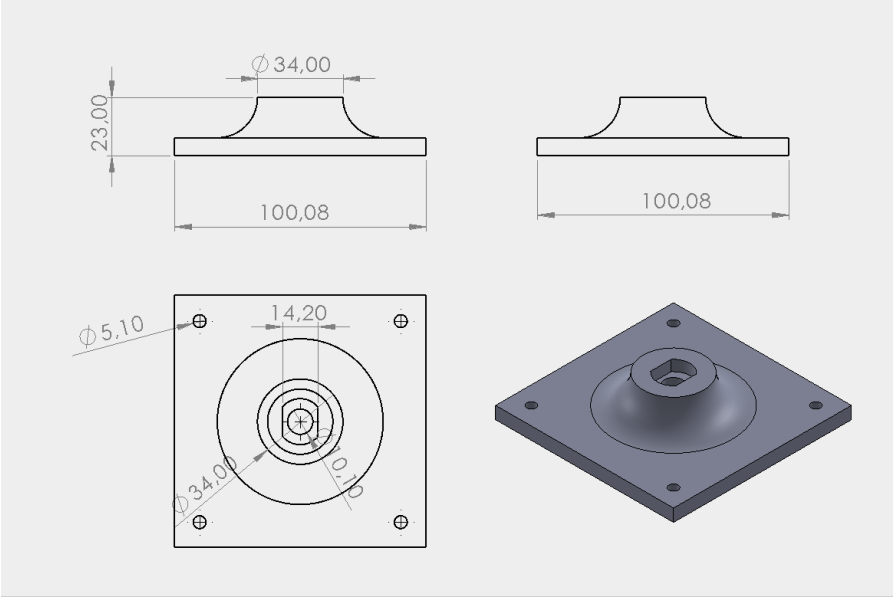


Figure C.2. ball_joint_platform.

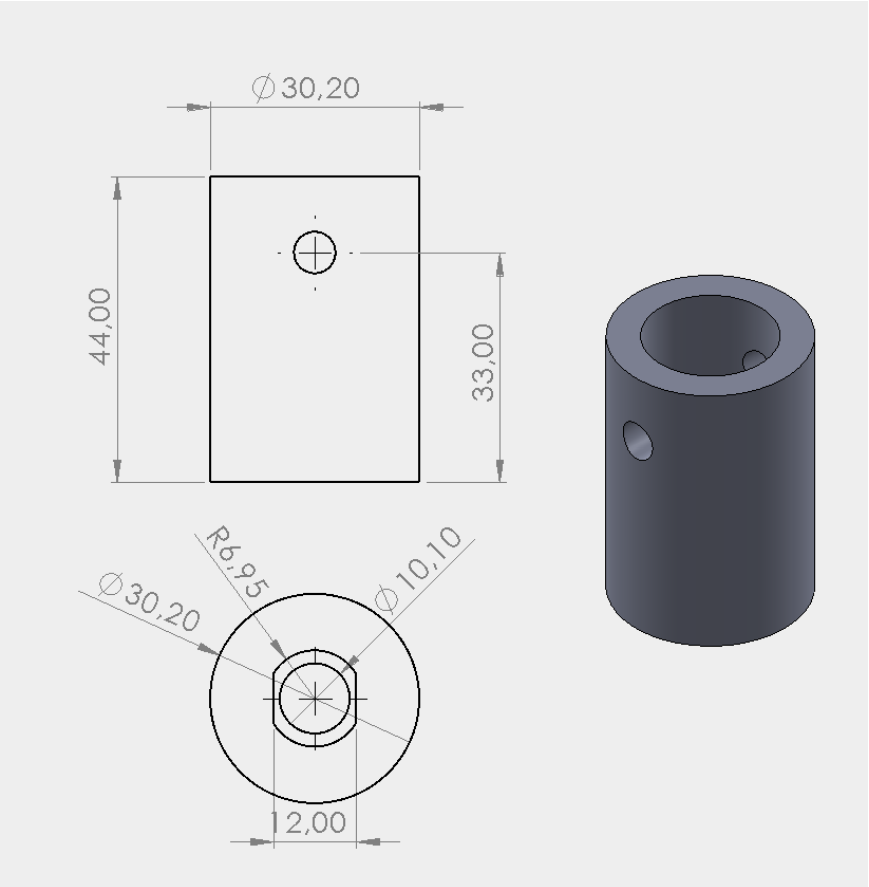


Figure C.3. ball_joint_linear_motor.

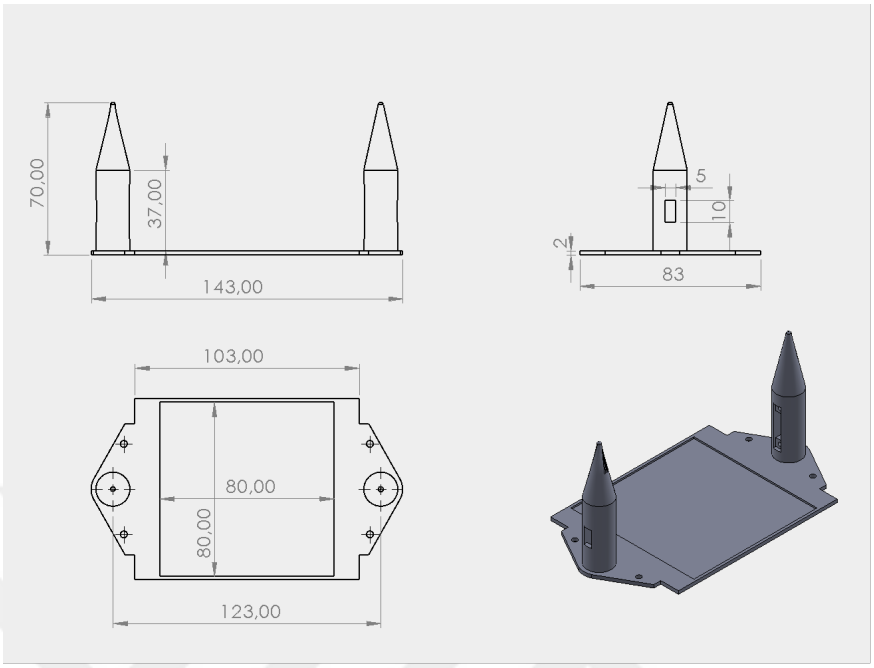


Figure C.4. case_bottom.

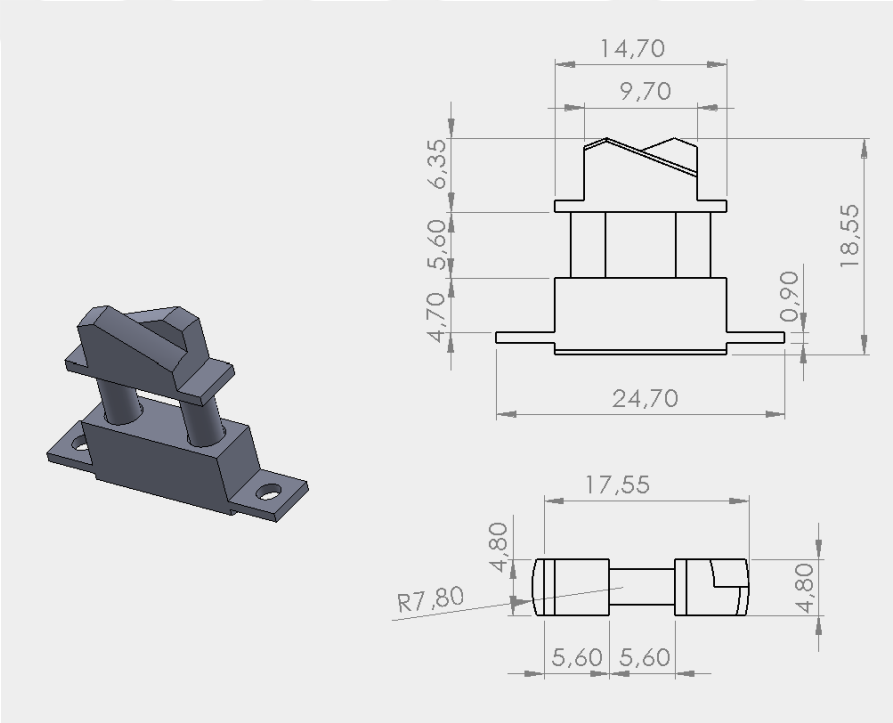


Figure C.5. case_latchbolt.

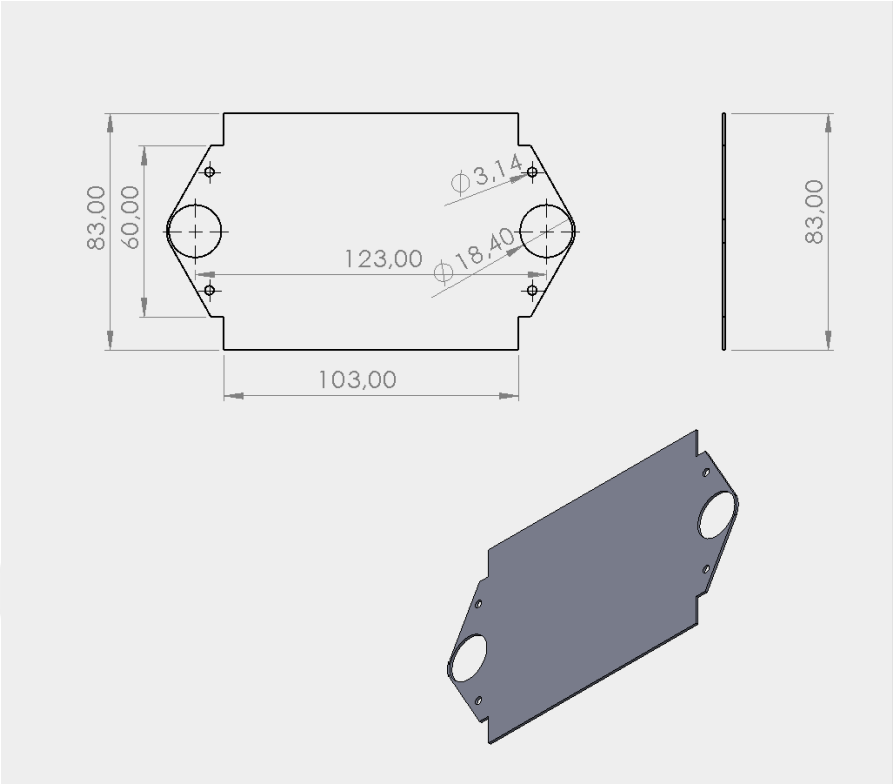


Figure C.6. case_lid.

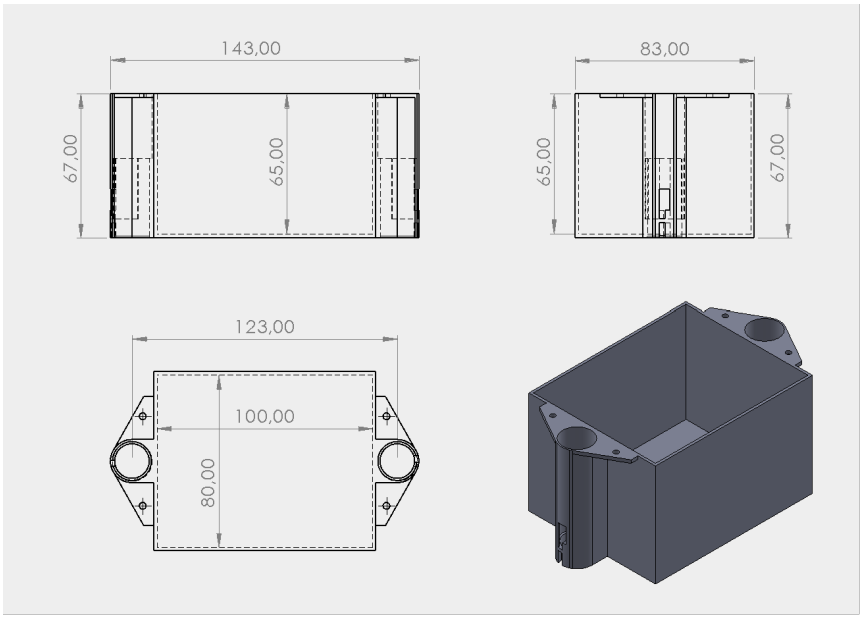


Figure C.7. case_middle.

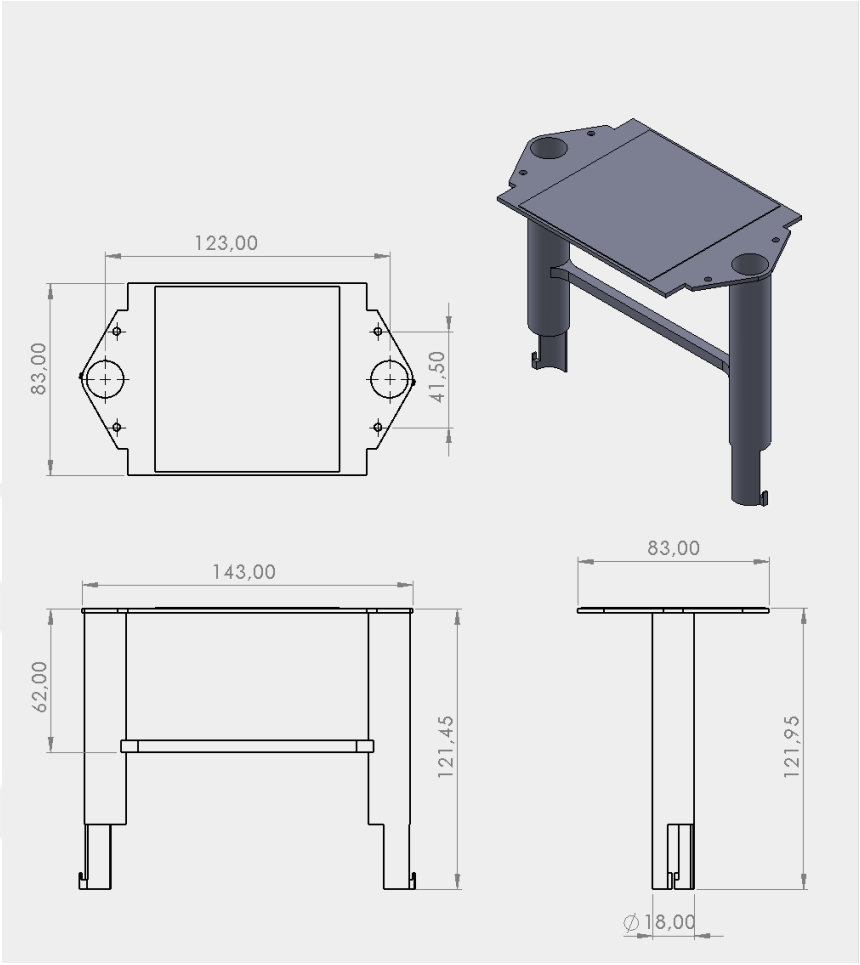


Figure C.8. case_top.

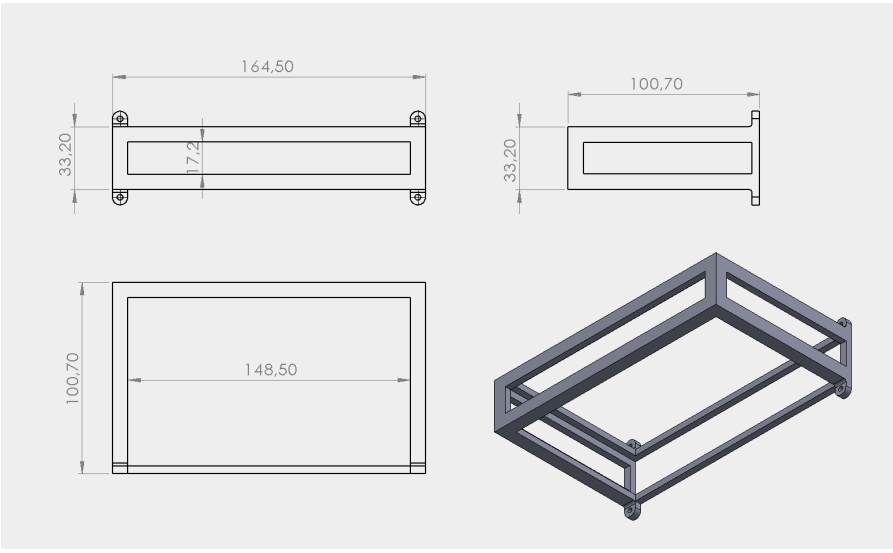


Figure C.9. ethernet_switch_hp_adapter.

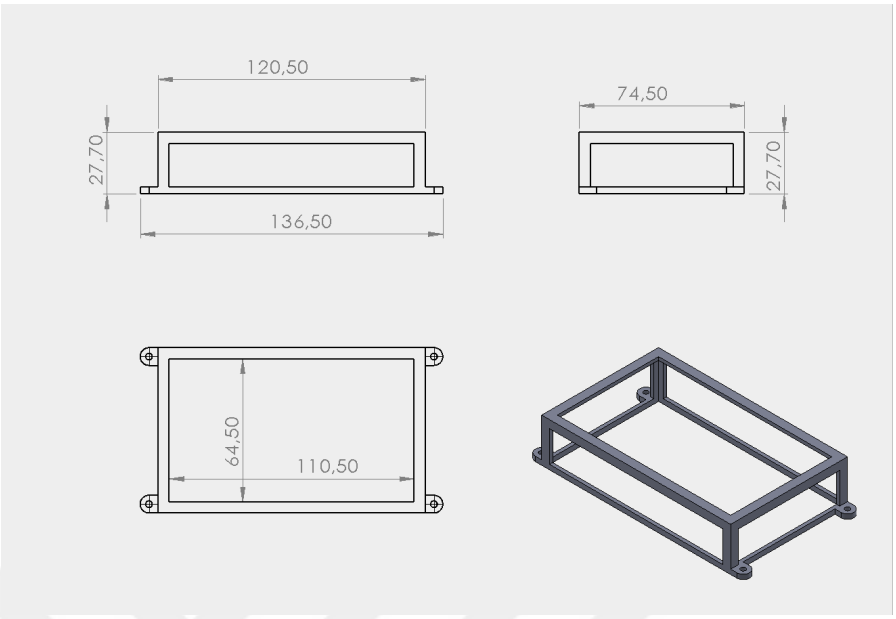


Figure C.10. ethernet_switch_tnet_adapter.

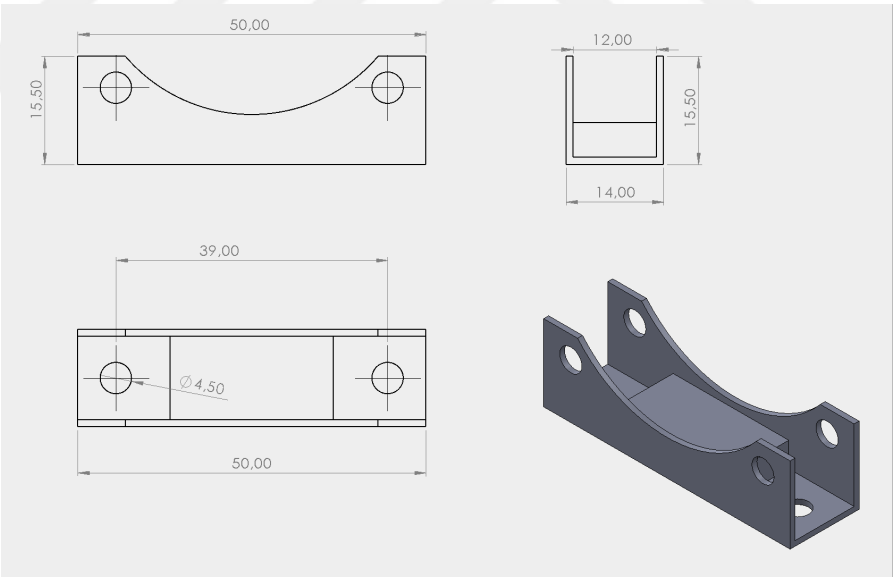


Figure C.11. fan_adapter.

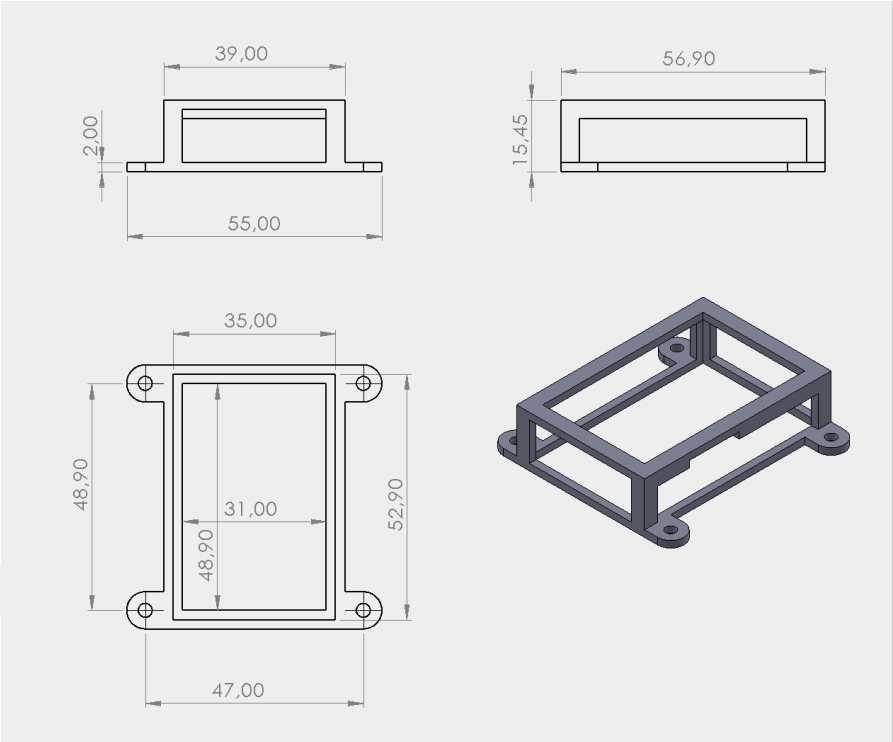


Figure C.12. gps_f9p_adapter.

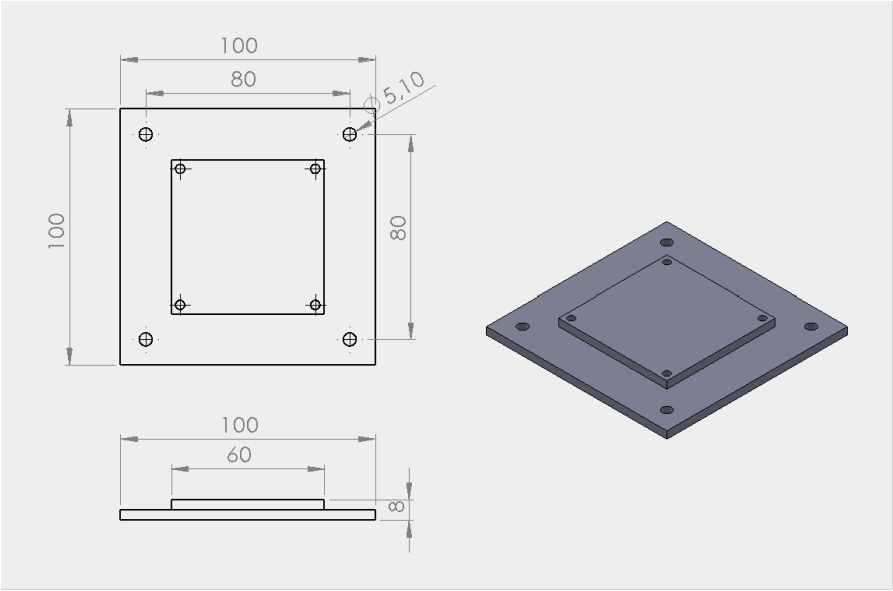


Figure C.13. hokuyo_utm30lx_adapter.

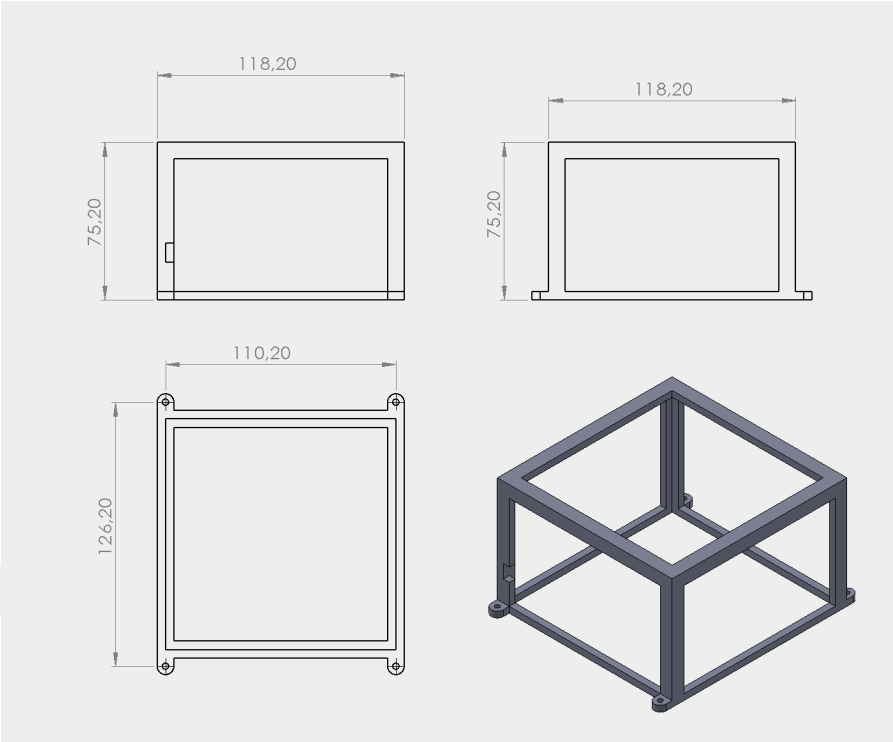


Figure C.14. jetson_agx_orin_adapter.

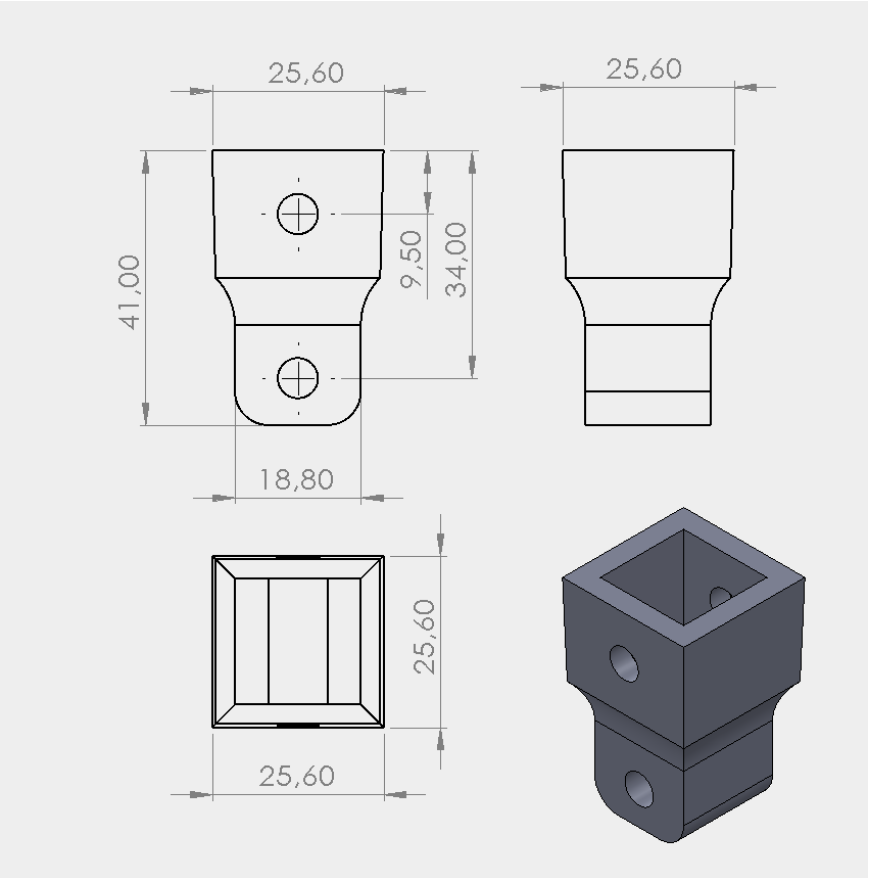


Figure C.15. hinge_joint_linear_motor.

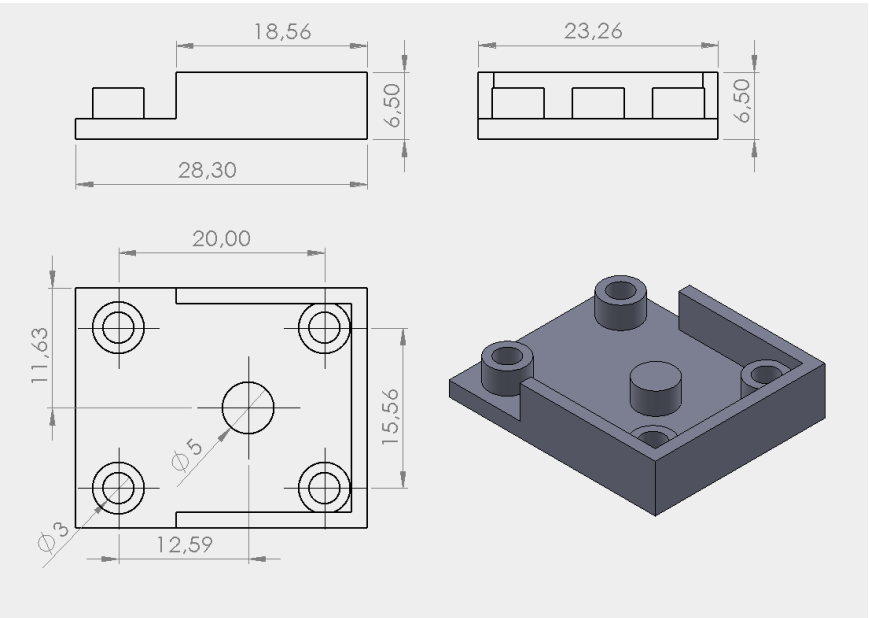


Figure C.16. mpu6050_adapter.

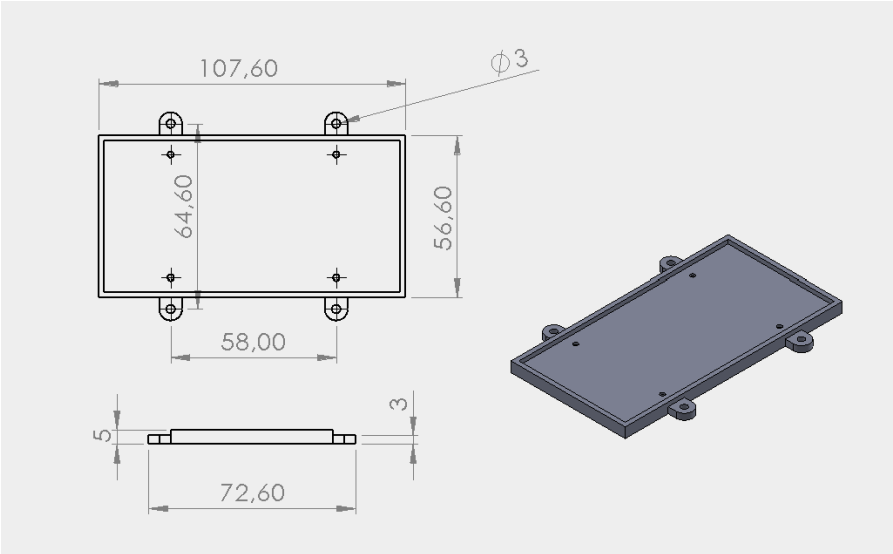


Figure C.17. pixhawk_6x_adapter.

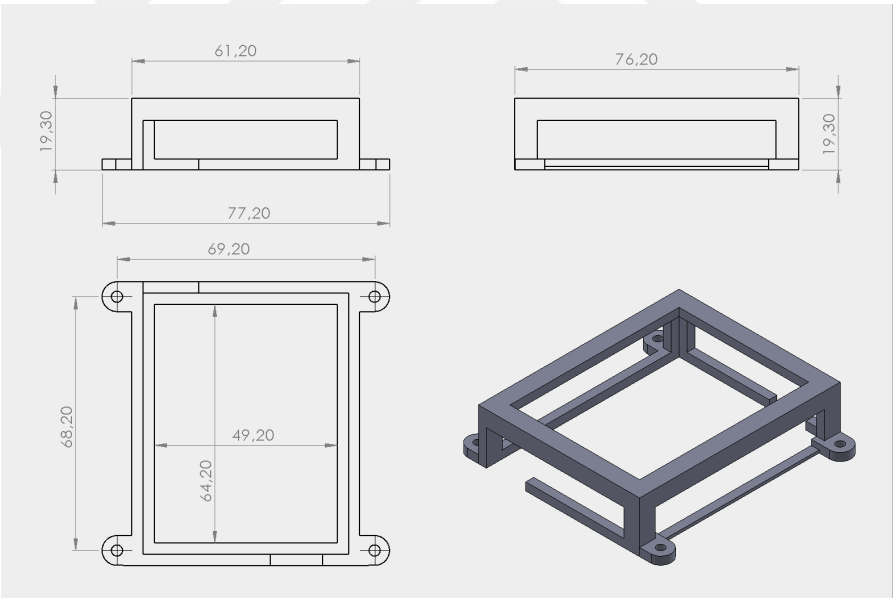


Figure C.18. siyi_air_unit_adapter.

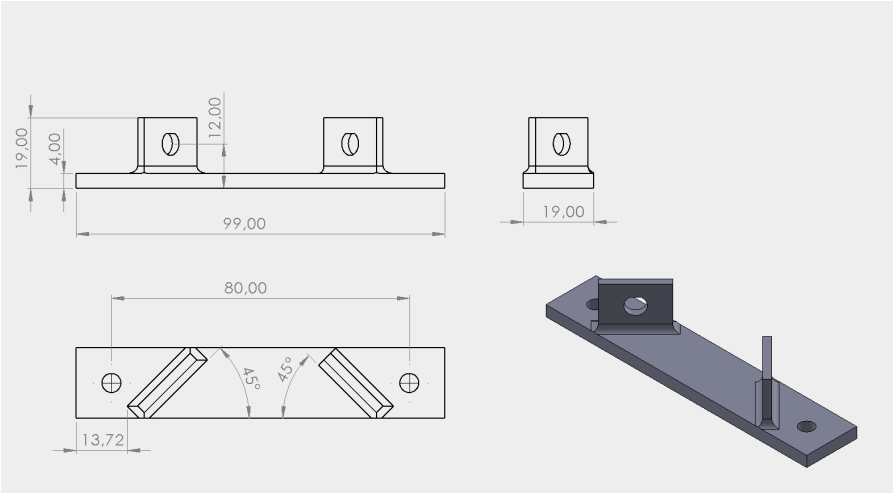


Figure C.19. siyi_antenna_adapter.

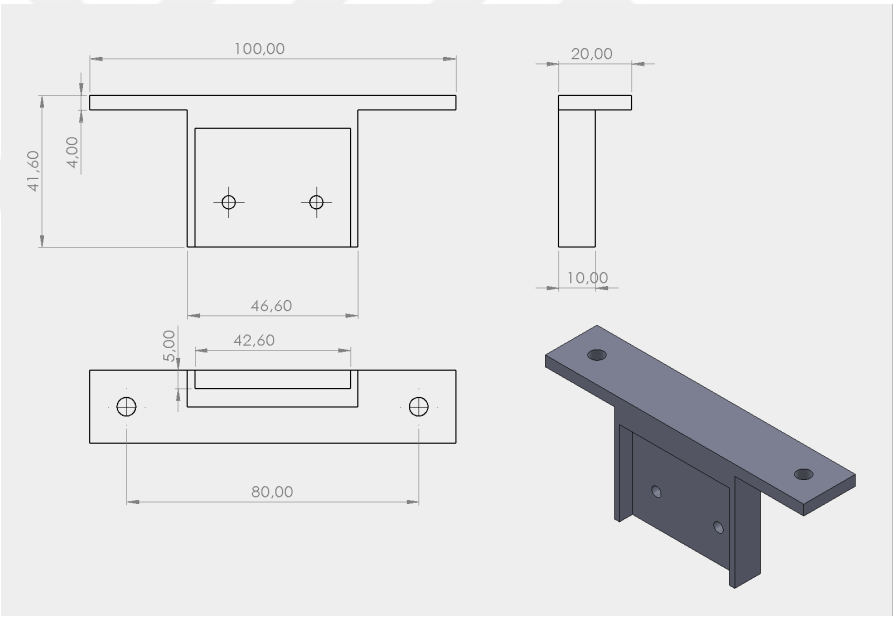


Figure C.20. siyi_ip_camera_adapter.

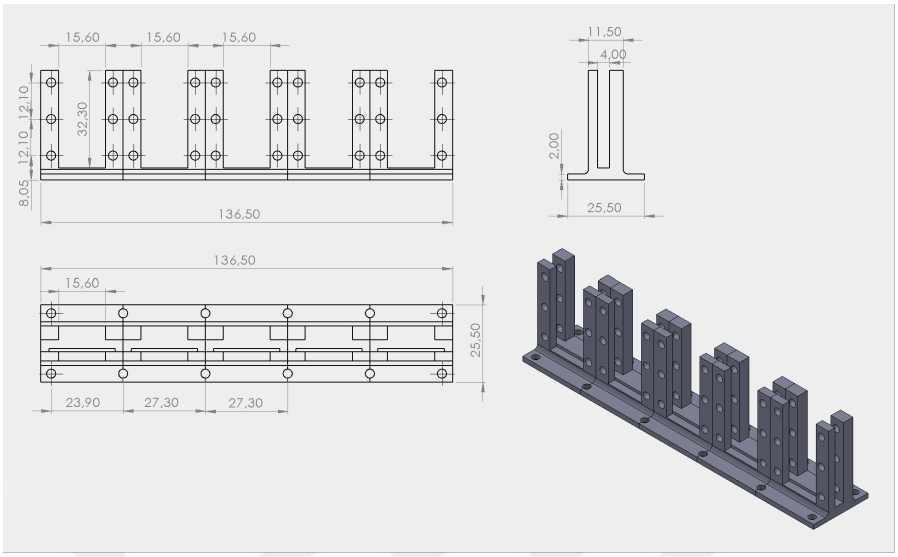


Figure C.21. XT60E_M_adapter.