Design and Implementation of an Automated Battery Management Platform

by

Tuna Toksoz

B.Sc., Computer Science
Bogazici University (2010)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Aeronautics and Astronautics
August 22, 2012

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Design and Implementation of an Automated Battery Management Platform

by

Tuna Toksoz

Submitted to the Department of Aeronautics and Astronautics
on August 22, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis describes the design and the implementation of the hardware platform for automated battery management with battery changing/charging capability for autonomous UAV missions with persistency requirement that extends the mission duration beyond the life of a single UAV battery. The platform is tested through a series of missions lasting at least 3 hours to prove it meets design requirements and to show its feasibility. This thesis also provides a method to modify existing scenarios to proactively plan for the battery maintenance so that the overall system performance is increased. The modifications made to the problem definition increased the state-space significantly, and means of solving a problem of that scale needed to be developed. To address this challenge, this thesis extends a previously developed approach called Incremental Feature Dependency Discovery (iFDD) by allowing to use caches from computer science literature to make conversion from basic features to extended features faster. By doing so, this method significantly reduces the computational complexity.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

First, I'd like to thank my advisor, Professor Jonathan How, for his vision, for his talent and sometimes for his harshness. Since the first day of my internship in his lab in 2009 and 2010, he has been really interested in what I have been doing, and he was really helpful to guide me in the right directions in what needs to be done in the end. His "big-picture" mind always asked the right question, and made me think about the answer to them. I feel lucky to have worked in his lab.

I would also like to thank to Gokhan, for putting his trust in me and recommending me to Jon for internships and for Masters, without even knowing me in person. Without him, it would have been a dream to work with all these smart people in ACL.

I would like to thank my project team, a.k.a Team Turkey. K"a"mal has been a great friend, and it would be really boring to do overnight experiments without him. I will miss his friendliness and his strong Turkish sense of humor for sure. I would like to thank Josh for simply being everything I wanted to be, and guiding me through everything in the lab.

I would also like to thank my former and current lab mates. I owe a special debt of gratitude to Mark, who has been a great friend and who helped me a lot on my hardware problems; Andrew, who was courageous enough to let me drive his manual transmission car and who shared part of our Turkish sense of humor; Sameera, who shared my complaints about academic life by heart; Vishnu, who spent countless hours tutoring me on FCP; Luke, who made me think about interesting C++ questions; Georges -the one and only- , who "let" me make his algorithms run million times faster; Dan, who always had interesting thoughts on a lot of topics; Brandon, who taught me how useful RRT could be; Alborz, who designed one of the algorithms I used in this thesis; Buddy, who gave me my ACL-name "Tunar"; Trevor, who shared my internet sense of humor.

I also would like to thank my UROPs, Dan Weber and Nick Kirkby for helping me a lot during and after the semester. Dan is one of the best hardware/embedded

software guys I have ever met, and his contribution to the recharge platform has been invaluable. Nick is the most hands on person I have met, and his help during final Boeing visit was thesis-saving. Matt Michini, I guess recharge station would still be a dream without your skills.

I would like to thank my family and old friends. Mom, I love you. Although you asked a lot of questions about how I was doing thousands of miles away from you, it was really important to me. Sister, you have always been the one understanding me. Thank you for keeping my secrets. Karolin, you have been my second thought - corrected me before making mistakes, you have always been straight with me. Tutku, thank you so much for coping with me through my emotional breakdowns. Onur, thanks for being my android friend, and referring me to the most awesome guy, Taylan.

Finally, I'd like to recognize Boeing Research and Technology for their generosity in funding my project from start to finish. Specifically, I'd like to thank Matt Vavrina, Dorina Hester and John Vian for the time they sacrificed to listen to, plan for, discuss and support my research on both theoretical and practical levels. It has been a pleasure working with each one of them.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Overview

Unmanned Aerial Vehicles (UAVs) have become the topic of significant interest in research for years in robotics and aviation. Developments in embedded computing, sensing and several other technologies made significant improvements in the capabilities of UAVs. In addition to being capable of doing sophisticated tasks, the lack of human occupant makes them well suitable for a wide range of missions including border patrol and search and rescue operations.

Many different types of UAVs for a variety of purposes have been designed. Military operations usually dominate the research and development in UAV field. In fact, in 2009 the U.S. Air Force started training more pilots to operate UAVs than to fly fighters and bombers [3]. Moreover, the U.S. congress mandated that by the year 2015, one-third of all ground combat vehicles will be unmanned [3]. To catalyze the development in unmanned aerial vehicles, FAA is currently working on regulations regarding the use of UAVs in civilian applications. This increase in the utilization of unmanned vehicles in general is due to their ability to reduce the cost of operation and training, and the risk involved in the tasks. If built well, the use of robotic systems can also amplify the ability of the operator, making many hard tasks easier to realize.

1.2 Motivation and Objective

The increasing demand for unmanned vehicles has driven many researchers into the development of planning and control algorithms for small and large scale missions. Different types of ground, aerial or naval vehicles are utilized in different mission settings as a result of the need to test planning algorithms for robustness to different factors in laboratory environment. An important class of these missions falls under the category of persistent missions, such as surveillance or target tracking, where the length of the mission is much longer than the flight time of a single agent. These types of missions are also practical scenarios in which the performance of the planning can be stressed in the long run.

Although it is ideal to test the algorithms in the long run, the flight time of an agent in indoor experiments is upper bounded by the vehicle's battery life, which limits the possible duration of the mission demonstration. This limitation has led to many different approaches to solving autonomous battery charging/changing problem. Initially, a human operator was involved in the battery changing process in persistent surveillance missions. However, this method requires at least one human constantly observing and doing repetitive work. In the presence of flying vehicles, this can also be quite dangerous. Automating this process has emerged as a necessity, and onboard autonomous charging mechanisms were introduced [1, 4–15]. However, charging batteries onboard is time consuming, and leads to low vehicle utilization. In particular, the design mentioned in [1] incorporates onboard battery charging to recover the battery after the quadrotor lands. This approach was demonstrated in a 24-hour mission and with several vehicles, but since the charging is slow, this approach was found to be very inefficient in terms of vehicle utilization. Even though it is possible to reduce the charging time by increasing the current rate [12], the charging time is still very long compared to the operational time. To increase the vehicle utilization, different mechanisms to replace the consumed battery with a fresh one are designed and implemented.

This thesis introduces a novel design that enables automated battery swaps with-

out shutting down the vehicle. The platform holds a buffer of seven batteries in a dual drum structure that ensures time-efficient swapping (on the order of seconds). Each battery is connected to a proprietary charger that supports multiple charging rates and battery chemistry for proper battery maintenance. The hot swap capability prevents vehicle from shutting down, eliminating the possibility of data loss.

The platform is demonstrated as a part of a large scale persistent search and track mission setting that is introduced in [16]. The objective of the mission is to search target vehicles in the surveillance, while continuously tracking those that are found. The mission is executed in Aerospace Control Laboratory's RAVEN test environment. Three quadrotor vehicles are utilized as the agents performing the mission, and three recharge stations were used as the battery maintenance agents to increase mission length. The details of the mission setting are given in Chapter 4. The MDP formulation in [16] is modified to incorporate battery states.

1.3 Summary of Contributions

This thesis presents several contributions to the long term battery management process.

- A detailed survey of recent studies in long term battery management area is included to provide a broad view of different techniques and devices implemented. The survey also provides the advantages and disadvantages of the proposed methods.

- The requirements of such battery management system is discussed in detail, and possible solutions are given. This also includes the decisions made in the hardware design presented in this thesis.

- A hardware platform is developed to enable missions of indefinite length.

- In order to show real world relevance, several flight tests were implemented with a team of multiple UAVs, ground robots and recharge stations, and important

results were provided. These flight experiments were executed both in ACL's RAVEN facility and Boeing's VSTL facility.

- Finally, the persistent search and track scenario is modified to incorporate battery states in the system.

1.4 Thesis Outline

The organization of the thesis is as follows: Chapter 2 discusses the need and design requirements for hardware platform for automated battery maintenance, introduces several metrics to measure performance of such system and presents the gaps in the existing designs. Chapter 3 discusses the details of the specific design introduced in this thesis. Chapter 4 provides experimental results about the impact of the hardware platform. Chapter 5 presents background on dynamic programming (DP) and Markov Decision Processes (MDP) and provides well-known algorithms to provide background on the subject. This chapter also introduces the problem formulation for the persistent search and track mission and proposes modification to incorporate battery states to provide true persistency. Finally, Chapter 6 contains concluding remarks and highlights areas for future research.

Chapter 2

Automated Battery Management Platforms: Existing Approaches and Design Requirements

As the use of unmanned aerial/ground/sea vehicles in autonomous missions increases, many efforts have been directed to the study of planning, control and navigation algorithms. Many of these applications fall into the category of persistent missions in which the planning algorithms try to optimize the system performance over a long period of time. In particular, for most of these applications, the desired mission length greatly exceeds the flight-time of a single UAV, hence monitoring the UAV fuel status is an integral part of the planning scheme. In an experimental setting, the flight-time of a UAV is limited by a battery life, making experimental demonstration and verification a challenging problem.

In order to extend experiment duration, a human operator was involved in the battery management process initially. However, this method requires at least one human constantly observing and doing a repetitive work. Automating this process has emerged as a necessity, and several approaches have been proposed by researchers. The following section provides a detailed literature review on the research done in automated battery management problem.

2.1 Literature Review

Initial work in autonomous battery charging problem has been done by the authors of [4, 5] for a ground robot. They implemented a docking platform to charge battery and demonstrated the platform through 1-week. Precise navigation during docking is implemented using two optical sensors that are used to follow a line to the docking station. The authors of [6, 7] designed a system similar to [4], but used an IR sensor to locate the docking station. A similar approach is currently being used commercially by I-Robot[17] and by Kiva Systems[18]. In [8] and [10], the docking mechanism is differentiated by including blob detection for localization. [9] uses similar docking mechanisms, and introduces estimation of operational time on a given charge level based on the data collected.

Authors of [19, 20] discusses design details of their hovercraft testbed for decentralized and cooperative control using a network of ground vehicles and how they solved the battery management problem. Proposed method uses a secure mechanism to provide electrical mating through the copper contacts at the bottom of the vehicle. Using proper LiPo charging scheme, they can make the estimation of the remaining time for the charging and plan proactively based on this information. Based on the numbers given in the papers (15min operation and 150min charge), their level of vehicle utilization is approximately 10%, which is very low.

The authors of [21] focused their attention not onto physical implementation or analysis of such a system, but onto learning aspect of recharging. Using artificial neural networks (ANN), they made their robot learn the implications of recharging on overall system performance, i.e. cumulative reward, in the long run.

In [22, 23], the idea of contactless charging using inductive power transfer and capacitive power transfer have been explored for soccer-playing robots. Similar to other online battery charging methods, this approach suffered from agent utilization. Proposed methods also had low-power transfer utilization due to disturbances in the environment.

The first and ground breaking work in battery management on UAVs was pre-

sented in [1, 12, 13], which discusses the first prototype of a battery charger platform was developed to support Aerospace Controls Laboratory's RAVEN[24–26] test bed. The authors demonstrated the functionality of the platform during a 24-Hour continuous mission. [15] uses a similar approach to charge the vehicle using direct connections to the battery through its feet. This approach, while easier to implement than the other methods, suffers from a very low vehicle utilization of around $(\leq 50)\%$, requiring at least twice the number of operational quadrotors to provide persistency.

In [27], authors proposed a significantly different approach to the problem by using high-energy laser beam to provide power to the quadrotor during flight. This quadrotor was directly powered by the laser beam, and the concept is demonstrated through 24-hour continuous flight. This method, however feasible, is costly to implement and demonstrate, as several mobile laser-emitting stations are required.

The authors of [28] and [29] focused their attention on thorough analysis of several recharge and replacement platforms and proposed a conceptual battery replacement platform. This research concluded that economically battery replacement platforms are preferable based on vehicle utilization and cost of overall system.

The authors of [14] developed the first prototype of a battery swapping mechanism as part of their ACE test bed. They designed a low-weight carbon-fiber battery pack for the vehicle which also interfaces with the charging unit. The secure attachment of the battery pack to the vehicle is provided through the use of magnets.

Based on this discussion, the next section provides analysis of existing approaches and introduce several design requirements.

2.2 Analysis of Existing Approaches

Swapping batteries manually This approach is a natural approach to swap the batteries without automation. In this approach, a human operator continuously monitors individual vehicle battery levels throughout the mission. When a vehicle spends its battery, the battery is replaced with a fresh one and the charging process is manually started by the operator.

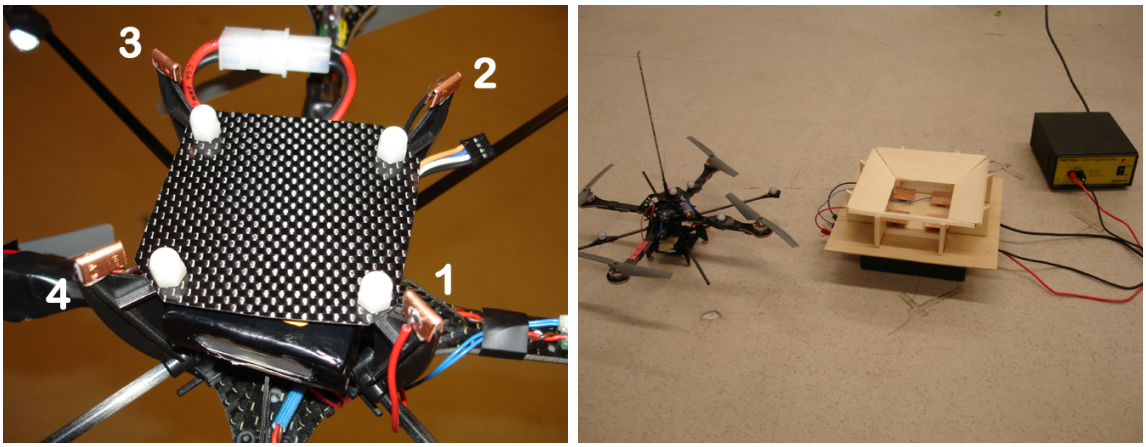


Figure 2-1: Original MIT ACL recharge platform approach [1].

However, at least one dedicated human operator must constantly monitor the battery levels. The number of human operators required to manually replace and charge batteries increases with the number of UAVs in the mission. This task is repetitive in nature, and it is a logical next step to automate it. This approach also requires a complete vehicle shutdown of the vehicle's onboard electronics as the spent battery is swapped for a new one. This adds further delay and a potential for losing onboard data and state information.

Charging batteries onboard This approach was first applied by the authors of [1, 12, 13]. In this approach, the quadrotor is modified to provide contacts with the landing pad connected to a charger as shown in Figure 2-1. This approach is considered the initial step towards automating and streamlining the process. The feasibility of the method has been demonstrated in a 24-hr flight operation.

The disadvantage of this design is that the charging time is very long compared to the life of a single battery. In experiments done in [1, 12, 13], a battery could sustain 8 minutes of flight, and charging time was around 40 minutes. Even when the charging rate is increased [12], the ratio of charging time to battery life is significantly high. In addition, there is an upper limit at the rate in which batteries can be charged safely which puts an upper bound on the vehicle utilization.

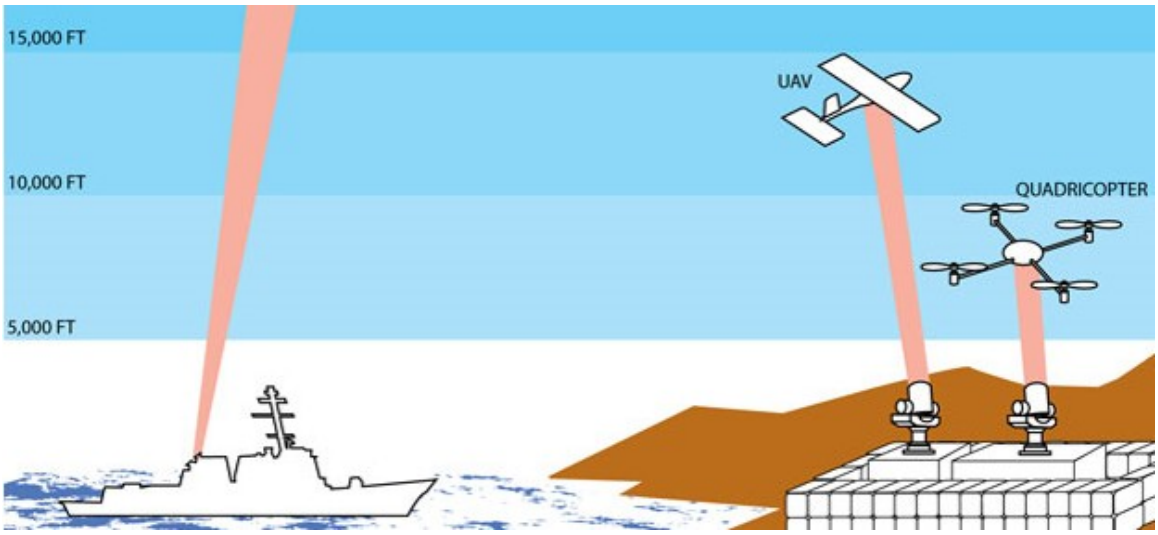


Figure 2-2: Illustration of laser-beam-powered UAVs. (Image courtesy of LaserMotiv [27])

Emitting high-energy laser beams to provide power This approach[27] is relatively new and provides a different approach to the problem. Instead of changing/charging the batteries, the power of the UAV is provided through high energy laser beams targeting the UAV. The illustration of this approach is given in Figure 2-2.

However, constructing high energy laser beam emitters in the field where a quick deployment of the UAV is necessary is not always possible. It is also expensive to construct beam emitters in such environment, and the number of emitters will need to be increased linearly with the number of operational UAVs. Furthermore, precision tracking of the quadrotor is a difficult control problem, and any deviation from the designated receiver area could potentially harm the quadrotor. This approach also suffers from obstructions and the fact that, as the distance from the emitter increases, the angle of reception will increase, yielding loss of energy.

2.3 Conceptual Design Requirements

Before the design phase, a certain set of requirements were defined for the system, as highlighted below. These requirements will also be used to evaluate existing approaches in Section 2.2 and will also help with the decision of choosing alternative

approaches.

1. **Continuous mission capability** – This is the main idea behind the platform. The designed mechanism should be able to sustain the mission indefinitely. In order to achieve this, careful selection of the number of vehicles, number of batteries, battery types, battery chargers and battery charging rate, and the charging method must be made.

2. **Minimal vehicle downtime** – Many of the methods highlighted in Section 2.1 have levels of vehicle utilization that are less than 50%. Systems with low levels of vehicle utilization require many more vehicles to provide the same level of coverage. In order to have high individual vehicle utilization, the vehicle should be put back to its operational state as soon as possible. The vehicle downtime usually depends on the renewal approach, charging time for a battery, and number of batteries available.

3. **No interruption of vehicle power** – On-board vehicle computer is usually carrying invaluable mission-related data such as photographs of a target or a video footage. The loss of power may result in loss of data. Moreover, the controller-related data is also important and loss of this data may result in reduction in the controller performance. The loss of power also causes delays due to initialization of the system when the new battery placed onboard. The uninterrupted power could be achieved using high capacity capacitors, or the design must have a mechanism to provide the quadrotor power during battery replacement. The latter is employed in the design proposed here.

4. **Reliability** – The platform will inevitably include moving parts and each moving part, in general, reduces the reliability of the system. The platform design must have a low number of moving parts, and those moving parts should be implemented with reliable actuators.

5. **Small footprint** – In space-constrained areas like lab environments, each battery management platform will reduce the effective size of the room, so it is

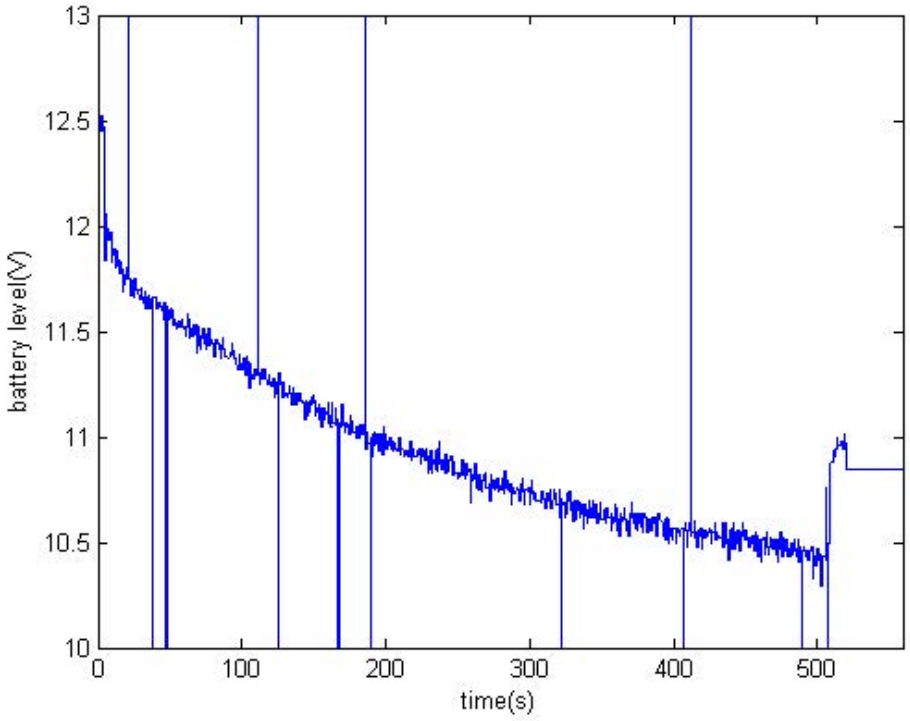


Figure 2-3: Discharge of a battery during flight

imperative to minimize the area used by the battery management platform.

2.4 Parameter Selection

In the design of such system, there are several parameters that need to be selected such as battery charging rate, number of quadrotors, level of persistency, and number of batteries per charger. In order to select the appropriate parameters, several measurements have been done. These measurements include average battery charging time from a defined low-level to fully-charged level. The hardware used in these measurements is described in Chapter 4, but in summary a quadrotor copter which uses 3-cell 1350 mAh battery. The voltage of a discharging battery in a quadrotor during flight is shown in Figure 2-3. This plot is a representative of an average battery discharge during flight. The battery is discharged until a predetermined low value of $10.5V$ (or 11.2V when the quadrotor is turned off).

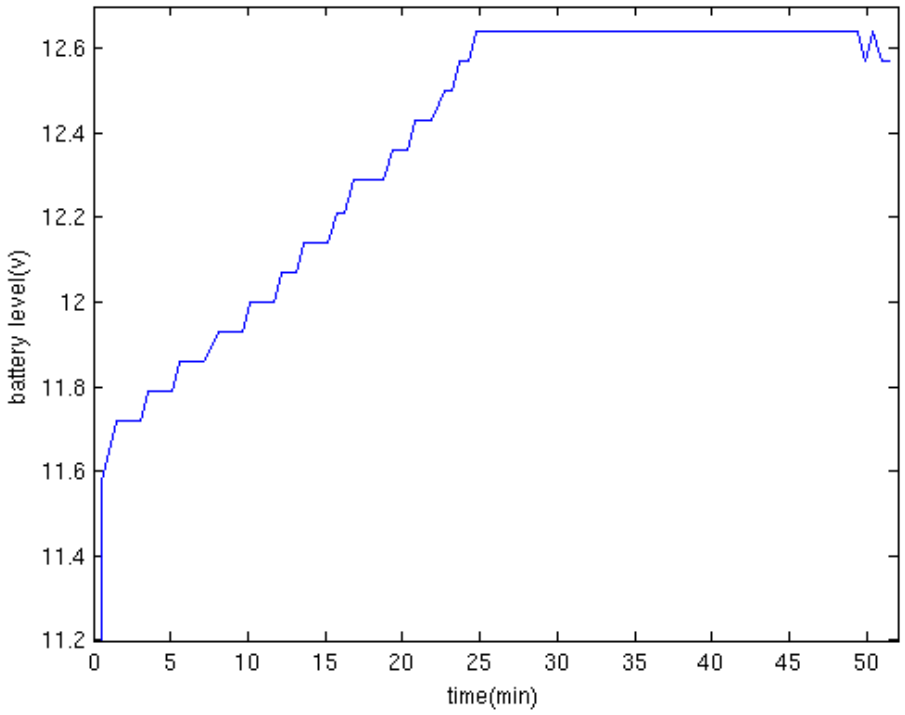


Figure 2-4: Single battery voltage during the charge process.

Figure 2-4 shows battery voltage over time during charge process. The battery is charged in constant current phase between $t = 0$min and $t = 25$min, and then it enters into constant voltage phase between $t = 25$min and $t = 50$min. Figure 2-5 shows the time required to charge the 32 batteries. On average, it took 52min to charge a battery from low ($11.2V$) to charged ($12.6V$), the variation in charge time comes from the fact that batteries have different usage pattern - some are charge-cycled more than others, or discharged more than their safe level.

Using the Petri analysis method described in [28], parameter selection could be done systematically. The platform designed needs to support one quadrotor indefinitely. However, since to provide full coverage, we need at least 1 vehicle in air at all times, $N_{UAV} = 2$ is selected. Taking $T_I = 0$ s, $T_F = 8$min, $T_C = 52.02$min, and $N_{CGR} = N_{BATT} - N_{UAV}$ and assuming a pessimistic value of $T_R = 1$min, and aiming

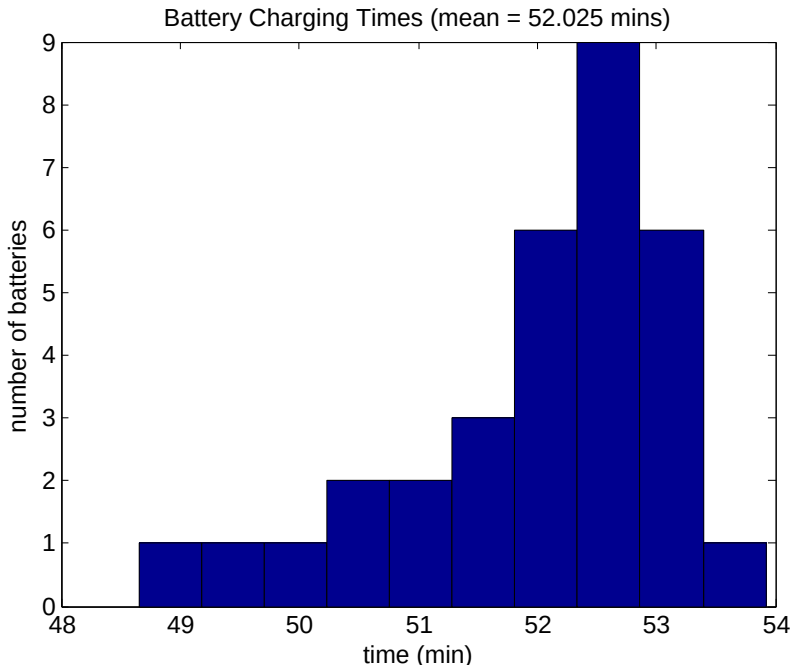


Figure 2-5: Battery charging times for 32 batteries

for $C_{SYS} \geq 1.05$,s

$$T_{L_{UAV}} = \frac{T_F + T_R + T_I}{N_{UAV}} \tag{2.1}$$

$$T_{L_{PLAT}} = \frac{T_R}{N_{PLAT}} \tag{2.2}$$

$$T_{L_{BATT}} = \frac{T_C + T_R}{N_{BATT} - N_{UAV}} \tag{2.3}$$

$$T_{L_{CGR}} = \frac{T_C}{N_{CGR}} \tag{2.4}$$

$$T_{CYC} = \max(T_{L_{UAV}}, T_{L_{PLAT}}, T_{L_{BATT}}, T_{L_{CGR}}) \tag{2.5}$$

$$C_{SYS} = \frac{T_F}{T_{CYC}} \tag{2.6}$$

it is found that $T_{CYC} \leq 7.61$, and $N_{CGR} \geq 6.95$. This number indicates that in order to provide continuous coverage for 1-quadrotor task with two quadrotors, second one taking the first's place only when battery is discharged, the recharge platform needs to have at least $\lceil 6.55 \rceil = 7$ batteries. This is also the number chosen for the design explained in the next chapter. Taking $N_{CGR} = 7$, the overall system coverage is found

to be

$$C_{SYS} = 1.056 \tag{2.7}$$

which indicates that the system could run indefinitely, and there is a 12% buffer in battery charging time, i.e. as long as batteries, on average, are charged in 53.33min, the platform will be able to sustain persistent operations. Chapter 3 discusses the specific implementation details of the platform designed that holds $N_{CGR} = 7$ batteries.

2.5 Conclusion

Based on the discussions in previous sections, the next chapter will provide design details of a battery exchange/charge platform that has a buffer of **7** batteries. The station makes use of off-the-shelf charging circuits to support missions of arbitrary lengths. The design employs rail-like structures to enable rapid battery swapping to minimize vehicle downtime. The sensors employed in the recharge platform increases system robustness by introducing checks for proper battery placement. There are also sensors that enable battery monitoring during recharge process.

Chapter 3

Conceptual Design and Implementation

This chapter discusses the design options for a battery maintenance station, and gives a detailed explanation about the choices made in the design described in this thesis. There are several aspects of such a platform that must be addressed during the design process:

1. Vehicle and power source selection
2. Battery carriage and skid design
3. UAV positioning & landing on the platform
4. Locking UAV and providing power while the battery swap is in place
5. Battery charging
6. On-board Electronics/Software
7. Off-board Controller Software

3.1 Vehicle and power source selection

In a maintenance system, it is preferable that the design can support as many different types of UAVs as possible. The selection of the vehicle determines the type of power source used by the vehicle, hence the design of the station.

Table 3.1: Battery types and their properties

Battery Type	**Cell Volt(V)**	**Energy Dens.(W.h/kg)**	**Cost(W.h/$)**
Nickel-Cadmium	1.2	$40-60$	$1.25-2.5$
Nickel-Metal Hydride	1.2	$30-80$	2.75
Lithium-Ion	3.6	$150-250$	$2.8-5$
Lithium-Ion Polymer	3.7	$130-200$	$2.8-5$

In laboratory environments, battery powered vehicles are used due to their ease of refueling [25, 30]. There are a number of rechargeable battery types in the market whose properties are summarized in Table 3.1. Due to their high energy density and high discharge rate, Lithium-Ion Polymer (Li-Po) based batteries are widely used, and will be the battery of choice for this platform. This decision reduces the total weight of the payload the aircraft has to carry while increasing flight duration.

There are several important points to keep in mind when using Li-Po batteries:

1. Each cell of a Li-Po battery has $3.7V$ nominal voltage, and $4.2V$ full voltage.

2. The battery should not be charged and discharged more than the charging rate advertised by the manufacturer. This charging rate is usually 1C[1] for non-balanced charging, and 2-6C for balanced charging. Charging the battery significantly above these rates is dangerous as the battery may catch fire. It will also significantly reduce its lifetime. Discharge rates are usually relatively high, and they are usually around 20-40C.

3. As a rule of thumb, the battery should not be discharged below 80% of its capacity to prolong its lifetime. A 80% discharged battery will give approximately $3.7V$ per cell under no load, which corresponds to $11.1V$ for a 3S Li-Po pack.

Due to their simple structure, ease of control and stability [25, 31–33], a quadrotor is used as the vehicle of target. The properties and the design of the quadrotor are explained in detail in Chapter 4. Even though the demonstration is done using a

[1]Capacity indicates how much energy the battery pack can hold and is usually given in mAh. This means that using 1C discharge rate, it would take 1 hour to drain the battery completely.

quadrotor, the design will be able to support any rotor-craft vehicle that can do vertical take-off and landing.

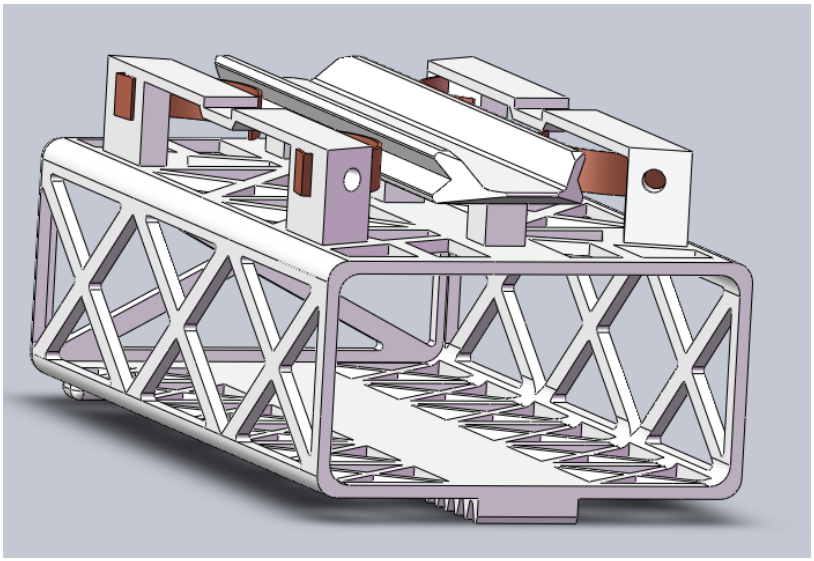
The onboard controller for the quadrotor provides useful health information such as battery voltage and individual motor currents. This information is currently used to reactively [34] respond to dropping battery voltage. In the future, batteries could be tagged with RFID stickers and their collected health data may be used to accurately estimate their remaining flight time.

3.2 Battery carriage and skid design

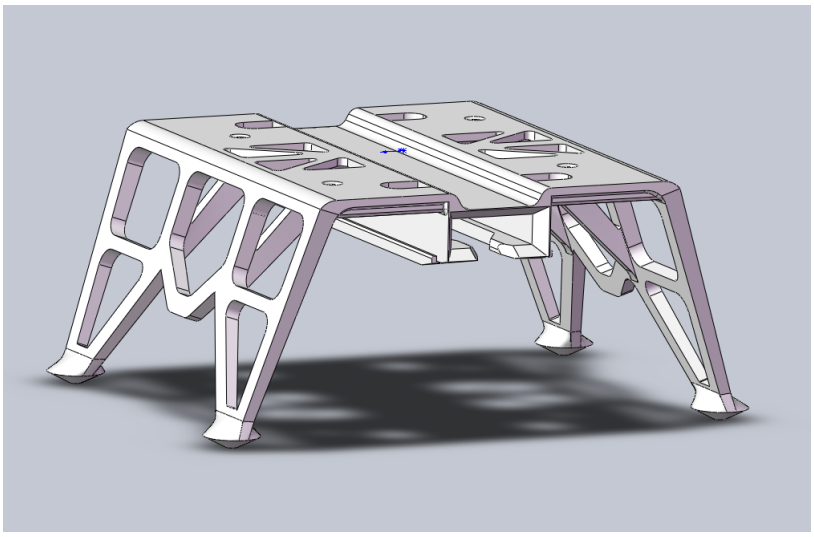
In a normal R/C aircraft, the aircraft is connected to the battery through wires and connectors such as Ultra Deans. These connectors typically provide a really strong connection ensuring stable connection. However, they are not suitable for rapid battery swapping. A special means of packing the battery in a rapidly-swappable way is necessary. In [28], the authors propose a method involving electromagnets on the batteries and the carriage. This method, although applicable, is complex and lengthy in the sense that swapping a battery is realized in a 3-step process involving actuation in multiple axis. The battery is first extracted from the aircraft, the magazine of arrays is rotated to get the battery with highest potential, and the battery is pushed into the vehicle. The overall process reportedly takes $47.5s$ on average. There is an opportunity to reduce the swap time, and an alternative design is proposed.

The battery carrier is a rectangular prism-shaped structure that the battery is placed inside. Power contact is provided through the copper strips on both sides of the T-shaped rail. A gear is placed on the bottom of the carriage to allow linear motion when it engages with the pinion gears found on the landing pad. The CAD design is shown in Figure 3-1.

The quadrotor receiver is designed to be attached to any aerial vehicle that could do vertical landing and take of while mating with the sloped landing plate on the recharge station and accepting the battery carriage underneath. One of the objectives in such design is to minimize the combined weight of the carriage and the receiver so



(a) Battery Carriage



(b) Battery Basket

Figure 3-1: The battery carriage and the battery receiver with T-shaped channel.

that the quadrotor can carry them without significantly sacrificing the useful battery life. The strength of the carriage is provided in both vertical and horizontal directions by using an "X" structure pattern. This enabled a great reduction in the component weight, while still maintaining enough strength.

The basic philosophy behind the entire design is to create a battery swapping process with one linear motion that performs the steps of replacing the old battery with a new one as shown in Figure 3-2. This is in contrast to other possible options with multiple steps, such as removing the old battery into an empty bay, aligning a new bay with fully charged battery, and then inserting the new battery. This is how a human would do the task manually, but it involves multiple steps that complicate the overall process. The alternative approach taken here was to align three battery slots (e.g, an empty one on the left, the vehicle one in the middle, and a bay holding a charged battery on the right.) The aligned bays now provide a nearly continuous T-rail support from the far left to the far right on the device (The T-rail supports and the associate copper plates in the drum bays are visible at the top of Figure 3-6), with small gaps in between. Note that the two ends of the T-rail in Figure 3-1(a) are beveled to ease the transition across the gaps from one rail support to another it simplifies insertion into the new rail support, and then the T-rail can force the proper alignment as it moves across.

This process is clearly shown in Figure 3-1(b), which illustrates how the battery at the left (back) slides out while at the same time the one at the right (front) slides in to the battery receiver on the quadrotor. Figure 3-3 further illustrates how one battery carriage slides out while the other slides in. The combination of the curved copper strips on both sides and T-rail provide a pressure-fit with the receiver that prevents the carriage from sliding out in mid-flight while also providing electrical contact.

The quadrotor receiver is designed to be attached to any quadrotor helicopter while mating with the sloped landing plate on the change/charge station and accepting the battery carriage underneath. One of the objectives in such design was to minimize the combined weight of the carriage and the receiver so that the quadrotor can carry them without significantly sacrificing the battery life. In order to meet this objective,

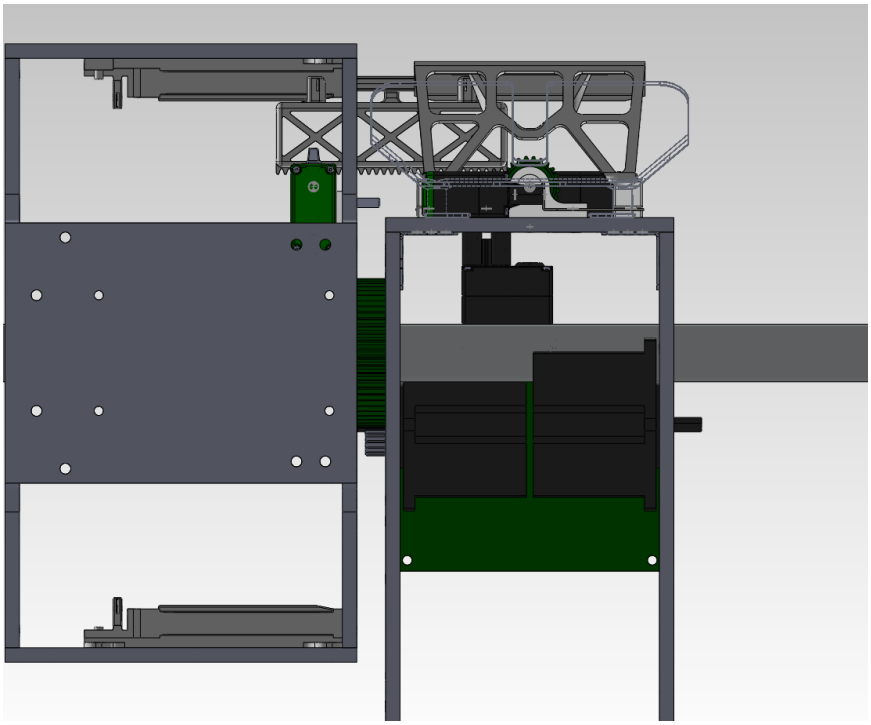


Figure 3-2: The battery basket is half way into sliding into the battery receiver on the landing platform. This simple motion enables fast and reliable battery swapping. Note that only left drum is shown for simplicity.

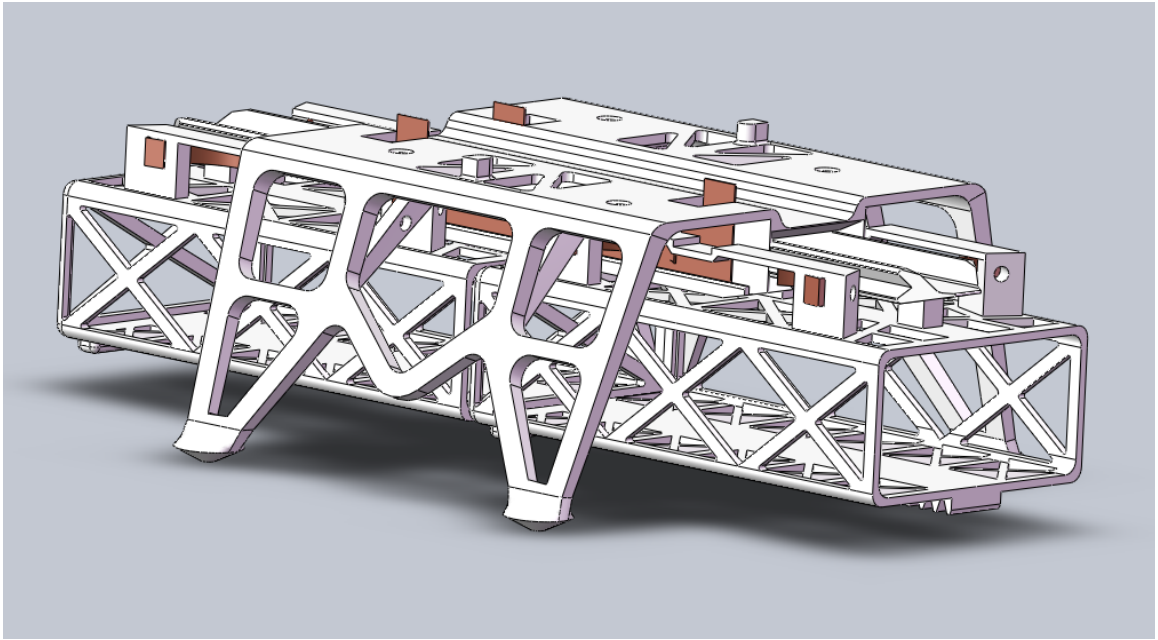


Figure 3-3: The Battery carriages sliding into and out of a quadrotor battery receiver.

Algorithm 3.1 The routine used for landing

function LANDINGTHREAD($x_{land}, y_{land}, z_{land}$)
  $landSpeed \leftarrow 0.07cm$
  $xyTolerance \leftarrow 0.02cm$
  $zTolerance \leftarrow 0.01cm$
  $approachHeight \leftarrow 0.5cm$
  $t_s \leftarrow 0.01seconds$
  $z_{goal} \leftarrow z_{land} + approachHeight$
  loop
    $x_{current}, y_{current}, z_{current} \Leftarrow getCurrentPosition()$
    $distance \leftarrow \sqrt{(x_{current} - x_{land})^2 + (y_{current} - y_{land})^2}$
    $z_{diff} \leftarrow |z_{land} - z_{current}|$
    if $distance \leq xyTolerance$ **then**
      $z_{goal} \leftarrow z_{goal} - landSpeed * t_s$
    end if
    if $distance \leq xyTolerance$ and $z_{diff} < zTolerance$ **then**
      $turnMotorsOff()$
      Break
    end if
  end loop
end function

careful structural optimization and material selection were performed.

3.3 UAV positioning & landing on the platform

This section introduces the measures implemented to minimize the position during landing, and also focuses on dealing with small errors that could still occur during landing. This improves the overall robustness of the total battery swapping process against possible disturbances or degrading landing performance.

In order to improve the landing performance, a special landing algorithm is introduced. During regular landing, a waypoint tracking method is generally used. However, when landing on a specific coordinate in the $x - y$ frame, the landing routine was modified such that instead of giving a step altitude command, the altitude command is decreased only if the position error in $x-y$ is within a given bound. This ensures that deviations in x-y frame could be corrected without further reducing the altitude. The landing algorithm is described in Algorithm 5.9.

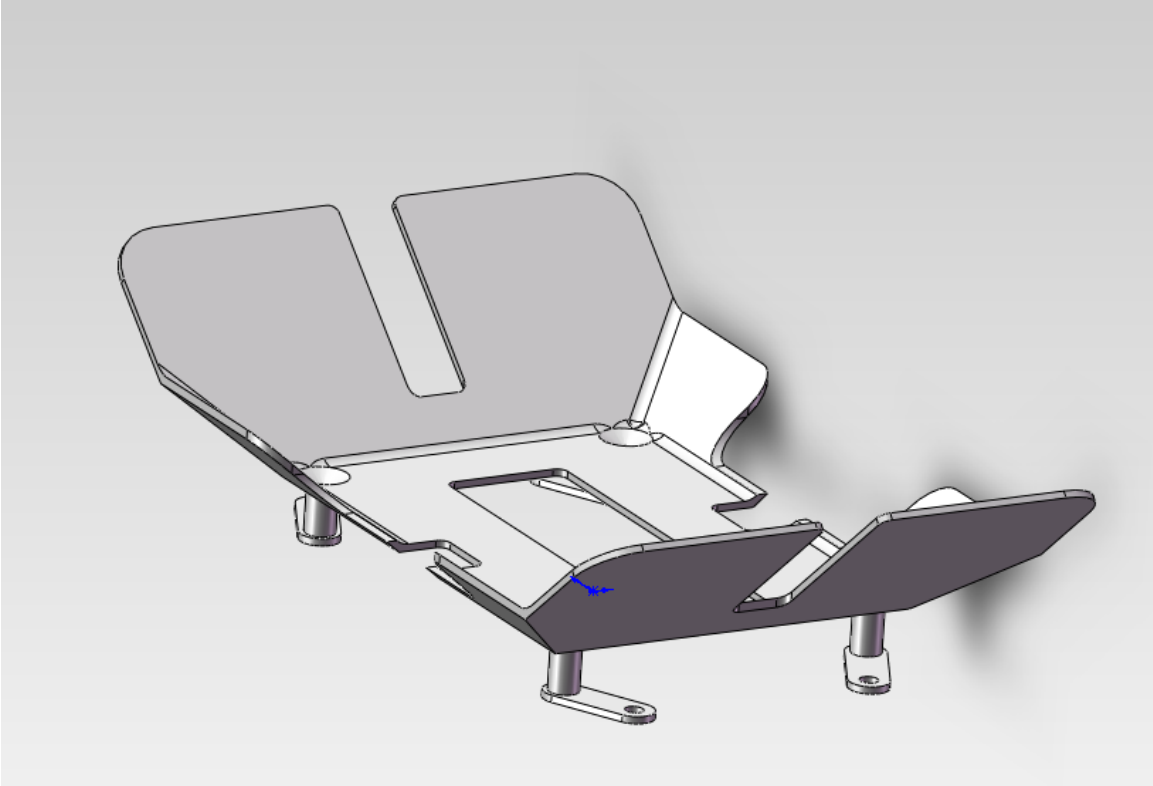


Figure 3-4: Sloped landing plate that guides the quadrotor into swap spots

Even when a specifically designed landing routine described in Algorithm 5.9, deviations from the final pose is still possible, and the station needs to have measures to handle that. For that purpose, a sloped landing plate is designed. If the quadrotor lands within that plate, the quadrotor is going to slide and it will be guided to the spots designed for the feet. This allows deviations as big as $5cm$ in one direction and $2cm$ in the other. The CAD is shown in Figure 3-4.

3.4 Locking UAV in place and providing power

Even after a near-perfect landing, there may still need to be a small correction in position and orientation. In addition, the quadrotor needs to be locked in place during swapping process, otherwise the battery carriage may get jammed before it could get into T-shaped rail. This component is implemented using two servo motors on each side of the landing platform attached to 3-D printer arms and a matching

section on the receiver. The arms will lock down the quadrotor in place from the beginning of the process, until after the fresh battery is pushed back into the vehicle.

One of the premises of this design is to keep the quadrotor powered when the swap process is taking place. This is crucial in case the communication with the base at all times is important, or cold-start of the quadrotor takes significant time. This can be realized in several ways, 2 of which are explained below.

1. The sliding mechanism could provide connection to both spent and fresh batteries simultaneously. This method ensures that the quadrotor is attached to at least one battery any given time, but safety checks such as ensuring the spent battery is fully pushed out is tricky.

2. The arms could provide shore power to the vehicle. This is the decision made in this design due to its simplicity. The arms have copper conductors attached to their tips, which are connected to $12.0V$ output on the board. The battery receiver has a matching connection parallel to the battery circuitry. This method makes sure that even when the spent battery is fully pushed out, it is powered through the arms.

3.5 Battery Charging - Charger Integration

The electronic circuitry used for battery charging is tightly coupled with the chemistry of the battery. Li-Po batteries, despite their high energy density and high discharge rates, need proper charging. They require to be charged through a method called constant current - constant voltage (cc/cv). The meaning of it is that a constant current is applied until the potential across the leads reach $12.6V$ for 3S battery, and then the current will start dropping while ensuring that voltage remains constant. The charging process will stop once the current drops to $0A$.

In this design, a commercially available multi-chemistry smart battery charger T6 by Thunder Power is used. This charger has the lowest footprint per battery, and it supports different battery chemistries such as NiMh, Li-on and LiPo. It is capable

of doing both balanced and unbalanced charging. One downside with this charger is that it doesn't provide any means of communication with it through protocols like UART or I2C. Therefore, a separate circuitry to control when to start and stop the charger, and measure the battery voltages is implemented.

3.6 On-board Electronics/Software

The system with previously described capability is composed of many circuit components.

The overall control of the station is realized through 4 microcontroller units. A robostix with ATMega 128 is used to communicate with the off-board controller, on-board charger controller and actuation of the drum steppers that is responsible from the alignment.

The actuation of the drums is realized by two stepper motors rotating in opposite directions. Previously, a motor module coupled with an encoder was used for the same purpose. However, precision control of the motor module was hard and it took longer to align the drums to a specific position. Stepper motors provide an easy way to control using step commands as opposed to Pulse-Width Modulation (PWM). They also provide high-torque rotation so that they can resist against the moment of inertia of the drum structure and 4 batteries.

Each drum has 1 dsPIC to control the motors that match the bottom gear of battery bays. They are responsible from pushing the new battery out, and pulling the new battery in. The dsPIC shares the same UART line with the XBee. This is beneficial because there are 2 UARTs on the Robostix which are already used to communicate with XBee and Charger controller board.

The management of chargers and measurement of the battery voltages are implemented on an ATMega 256 board. One important point is to isolate the charging circuit from the measuring circuit using operational amplifiers. Failure to do so results in ground loops, inaccurate voltage readings and dangerously high current levels. When the charger technology evolves, and chargers with computer control emerge, it

will be possible to read the battery voltages, current, and energy put into the battery.

3.7 Off-board Software

The off-board software is responsible from higher level actuation commands that are sent to the onboard controller through X-Bee. It also communicates with the mission manager, which decides when the swap should take place. These commands include arm lock, drum rotation, central sensor check for proper battery placement, voltage and position retrieval, bay and central motor actuation.

The off-board software is written as a part of Raven Framework in Python, and uses same messaging protocol. The software has a number of operational safety measures in place. The operation is suspended in case the swap could not be completed due to jammed basket/receiver. This is the only instance where a intervention of a human operator would be required.

3.7.1 Mission Manager

The mission manager is implemented in Python and is composed of a messaging protocol and a number of messages for operational, tasking and health purposes, position data protocol, and a number of utility functions. Mission manager is a general framework to implement scenarios for a given mission description. Aside from providing methods for commanding vehicles, it also provides methods for plotting data, observing vehicle states, and logging them. Using Object Oriented Abstractions, the mission manager can talk to systems in different labs and also to different types of vehicles. This was mainly designed because of the requirement of being able to transition software to Boeing facilities.

The messaging protocol ensures that the mission manager can talk to different vehicles and command them. As long as the bandwidth is sufficient, and vehicles implement the same messaging structure, the vehicle can talk to any number of vehicles that are on the same network.

The mission manager also has a built-in simulator to enable the developer to test

the code s/he writes before running the mission with actual vehicles. Simulation doubles for ground vehicles, quadrotors and recharge stations are implemented. It is also possible to do real/sim experiments, ie a mission run of both simulated and actual vehicles. Mission manager is also responsible from managing recharge station as a resource. In order to do this, recharge station keeps track of occupied recharge stations, vehicles occupying them, and the vehicles waiting for an empty recharge station. With more than one recharge station, the selection of which one to allocate becomes important. In missions where there are no higher level algorithms that decide which one to allocate, the mission manager makes the selection using several different algorithms as discussed in Section 3.7.2.

3.7.2 Station Selection Process

The mission manager needs to select an empty charger when a quadrotor goes low on battery. Currently, 3 selection mechanisms has been implemented based on requirements on previous experiments:

1. Round-robin: With this method, the charge stations are assigned to quadrotors in order, one after the other. This method ensures that each charge station has similar battery charge levels. This method also makes it easy to debug the recharge station use in the mission as what is going to be used next is known.

2. Preassigned: This method is the simplest to implement. Each quadrotor is assigned to a specific recharge station. This method is used mostly for debugging purposes.

3. Closest-First: This method makes sense when the environment is so large that the navigation time is no longer negligible. Previous methods don't use the vehicle location, and it is likely that one quadrotor could be sent to the furthest recharge station. When a quadrotor requests to refuel, the distance to each recharge station is calculated and the closest non-occupied one is selected.

All of these methods are able to distinguish a simulated vehicle from a real one so that simulated vehicles are prevented from going to real recharge stations and vice versa.

3.7.3 Dispatching Process

After selecting an available recharge station for the quadrotor, the mission manager has to perform a number of steps to ensure mission is continuing as desired and proper actions are taken before the battery swap operation begins:

1. The mission manager constantly monitors each vehicle state. Depending on the problem formulation used by the mission, either an algorithm proactively decides to send the quadrotor to the station to swap its battery, or the mission manager acts reactively and send the quadrotor to the station when the voltage goes below a certain threshold to ensure battery and vehicle safety. Usually it is a combination of both, the reactive planning will kick in if proactive planning fails to do so.

2. When the quadrotor is about to be sent to the station, it needs to be taken out of the available vehicles in the mission and all tasks assigned to the vehicle need to be dropped.

3. After mission manager marks the vehicle as busy for recharge operation, the quadrotor is commanded to land on a selected recharge station.

4. After landing, to ensure proper alignment and position on the station, the current position of the vehicle is checked against previously recorded values. If the error in recorded and the current pose is greater than what is tolerable by the station, the quadrotor is commanded to take off and land again. If the landing is successful, recharge station is commanded to initiate the sequence.

5. Depending on the mission type, after recharge station notifies the mission manager, the vehicle can be set to stay on the recharge station, set to take off right

immediately, or the quadrotor is taken off and land at a designated area. In a scenario like PSM, where number of available vehicles may exceed the number of flying vehicles needed, the 3rd option is logical, while the required number of vehicles is equal to the number of available vehicles, the first one is preferable.

3.7.4 Station Management Process

The following steps explain the charger management process step by step. The mission manager refers to the software that coordinates the vehicles to achieve a mission objective. It also monitors each vehicle's individual health information. The following step is also illustrated in Figure 3-5:

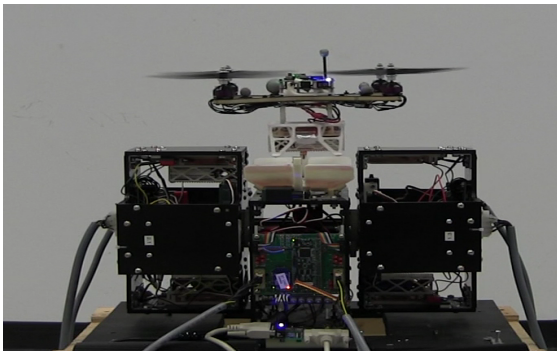
1. Mission manager continuously monitors the quadrotor health state, which includes the battery voltage and other components such as motor temperature and performance.
2. When quadrotor battery voltage is considered to be low, the mission manager calls the quadrotor back for refueling.
3. The quadrotor lands on the recharge station. The motors are turned off and the mission manager is informed of a successful landing.
4. The Mission manager then commands the recharge station to start the swap process.
5. The recharge station then locks & aligns the quadrotor using two servo-driven arms. A proper lock & alignment provides shore-power to the quadrotor so it can operate when the battery is removed, and toggle a sensor onboard the station that enables Step 7.
6. Recharge station ensures using the central photo-electric sensors that the quadrotor has successfully landed and battery is properly aligned.
7. The recharge station pulls the discharged battery from under the quadrotor and into an empty bay in one of the drums. Simultaneously, a charged battery is pushed from the opposing drum into position under the quadrotor. Sensors

on the station detect the proper positioning of the new battery, which triggers Step 8.

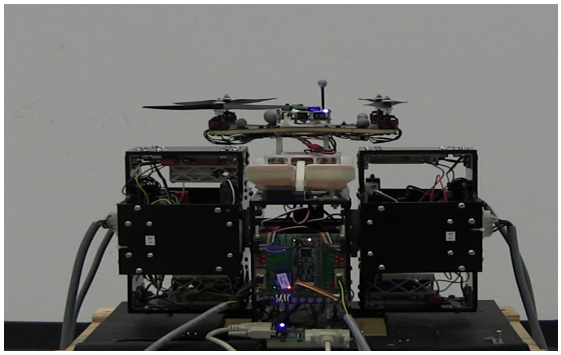
8. The recharge station releases the locking arms and notifies the mission manager that the quadrotor is ready for take-off. The battery station then scans the voltage level of the batteries in each bay and rotates the drums as necessary in order to place a bay containing a fully-charged battery on top, aligned with the empty bay. At this point, the quadrotor is available to be taken back into mission.
9. The recharge station will start charging the recently inserted battery.
10. When the quadrotor is needed back in the mission, the mission manager will send a take-off command to the quadrotor. The quadrotor will send a message back when it has successfully taken-off. The mission manager will then release the recharge station and make it available for the next battery change process.

The complete implementation of the platform is shown in Figure 3-6. The platform has a sloped landing plate to guide the quadrotor into the swap place and two arms are used to lock down the quadrotor and hold it in place. On each side of the platform is a rotating drum, each of which contains 4 battery bays. Each of the drums is connected to a battery charger to recharge the battery without human interaction with the system.

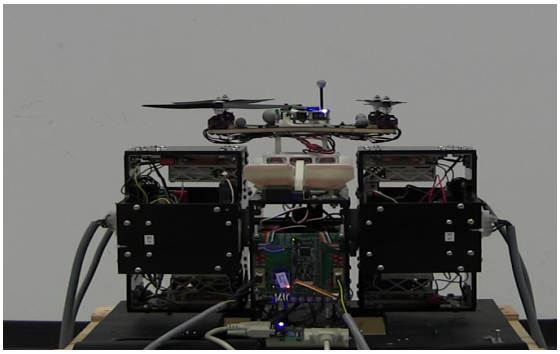
In operation, a quadrotor modified with a battery receiver and carriage is placed on a sloped landing plate and is locked down securely with two arms. The drums are rotated to align the appropriate battery bays. The battery on the quadrotor is then swapped out for the newly charged battery, and the old battery is placed into an empty bay on the opposite drum. The charging for that battery then starts until that battery is needed (approximately 1 hour later given the current system). Since all steps are automated, the platform provides the capability to automatically change and charge batteries without requiring intervention of a human operator.



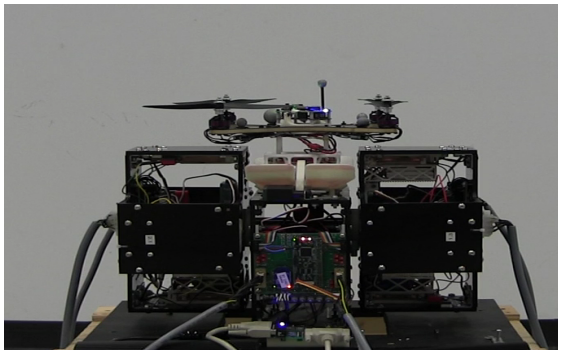
(a) Quadrotor hovers over pad and descends to land



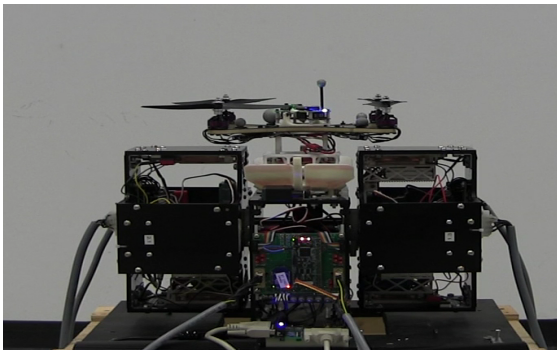
(b) Quadrotor clamped to pad with shore power



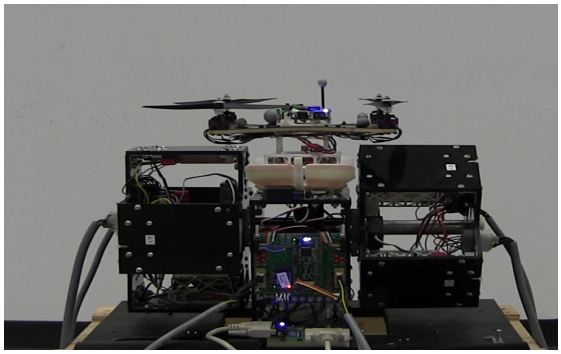
(c) Spent battery is pushed into the empty spot



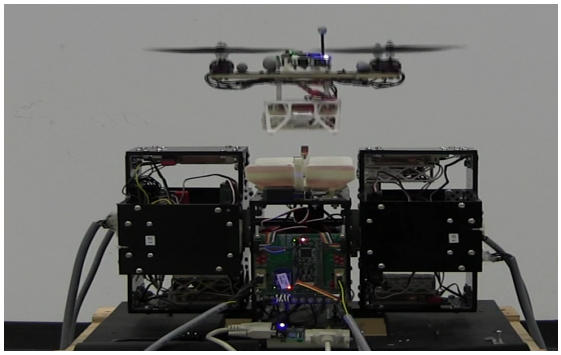
(d) New battery is pulled under the quadrotor



(e) Locking arms are released, quadrotor is ready to take off



(f) Next best battery is selected and drums are aligned accordingly



(g) The quadrotor is commanded to take off by the mission manager

Figure 3-5: One battery swap sequence from a multi-swap mission.

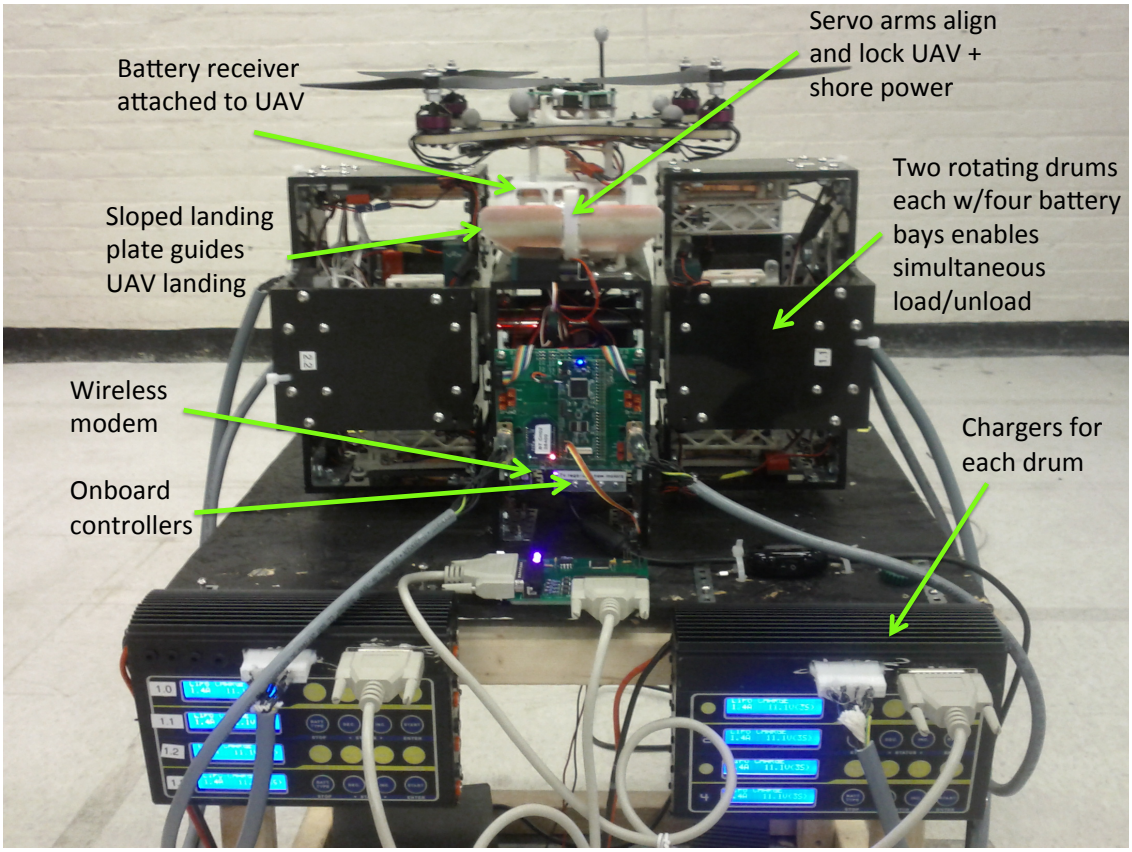


Figure 3-6: The final implementation of the recharge station.

3.8 Conclusion

This chapter provided design details based on the design requirements introduced in Chapter 2. The designed platform enabled rapid battery swapping, and fast battery charging. Using the arms attached to the landing gear, the quadrotor is provided with power even during battery swap operation, which prevented any data loss that would have been caused by loss of power. The sensors introduced in the platform increased system robustness. The next chapter is going to present hardware results that were obtained through multi-hour experiments.

Chapter 4

Experimental Recharge Results

4.1 Hardware Setup

There are three categories of hardware in the hardware setup: robots, maintenance station, and fast and accurate indoors positioning system. This section details each of these categories and details the specific components used in the persistent surveillance scenario.

4.1.1 Mobile Robotic Agents

Four different types of agents are used in the Persistent Search and Track scenario: An aerial agent (a quadrotor), ground vehicles (IRobot Roomba robots) as targets and ground surveillance vehicles and as civilians.

In addition, to answer the need for inexpensive aerial mobile robots, an in-house quadrotor is developed to be used in PSM and other related flight experiments. The quadrotor, shown in Figure 4-2, is built on a carbon-fiber and foam sandwich plate frame with brushless motors, electronic speed controllers [35] capable of measuring temperature and current, and an off-the-shelf autopilot board with accelerometers, gyros and a pic-based microcontroller [36]. The firmware for the autopilot was also developed in-house to close the attitude loop onboard for stable, hovering flight [37].

Figure 4-1: Four agent platforms used in flight experiments: Team UAV (top left), Team UGV (top right), Target UGV (bottom left), and a Civilian UGV (bottom right).

4.1.2 Indoor Metrology

Flight tests were carried out in Aerospace Controls Laboratory's RAVEN Indoor Flight environment[24, 25] and Boeing's VSTL environment[2]. Figure 4-3 shows the general layout of the MIT RAVEN facility, and Figure 4-4 shows the general layout of the Boeing's VSTL that enable rapid prototyping and testing of a variety of unmanned vehicle technologies, such as different robotic agents, flight controls algorithms, higher level task planning algorithms, coordinated flight algorithms in a controlled environment. RAVEN/VSTL utilizes a vision-based motion capture system to simultaneously track multiple robotic agents, and provide position and orientation information with sub-millimeter accuracy about these vehicles in real-time. This information is then distributed to a group of command and control computers responsible for managing the autonomous execution of the mission.

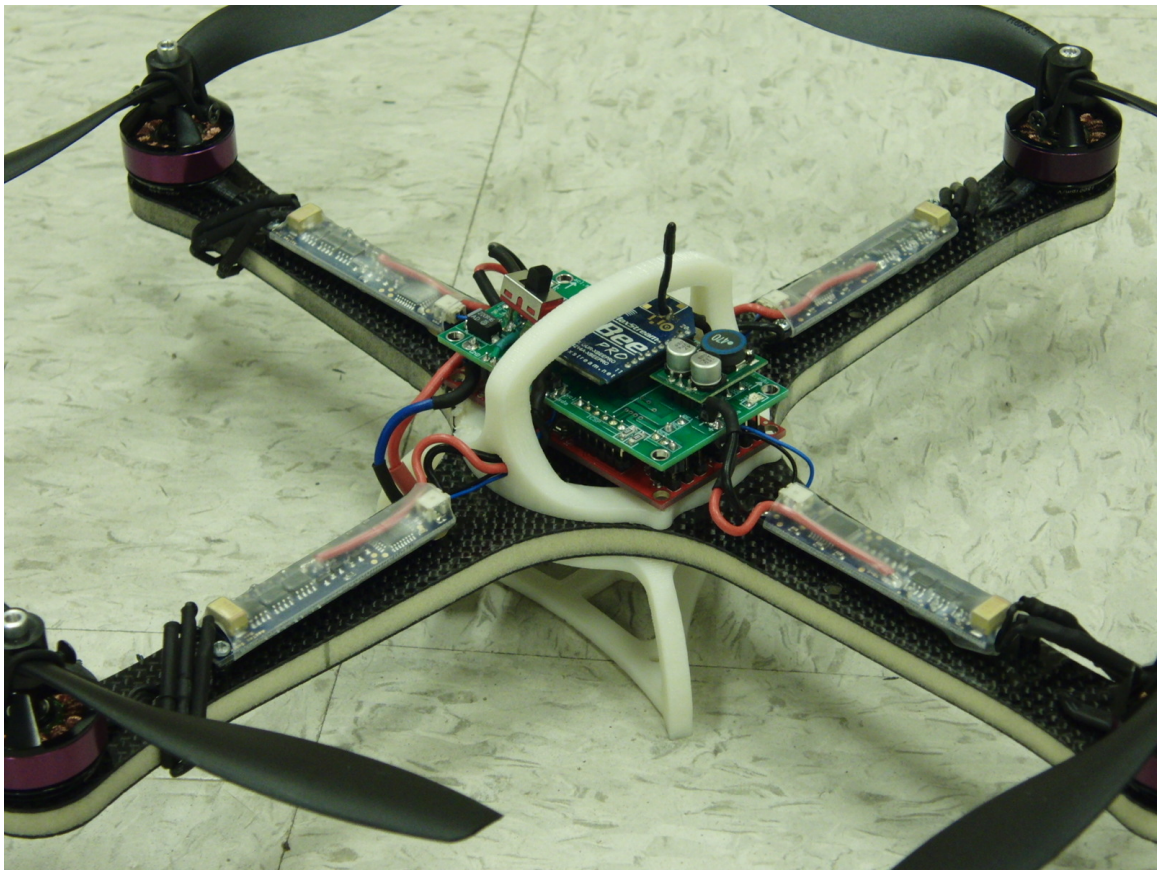


Figure 4-2: Quadrotor helicopter built in-house to answer our need for an inexpensive, autonomous, aerial mobile robot.

4.2 Results

This section presents and discusses the results obtained from hardware flight experiments of the recharge platform. An experiment for testing the proposed properties of the recharge station was performed and the results are given in Figure 4-5 and Figure 4-6. The numbers on the plot indicate which battery is being used by the quadrotor.

One cycle of the experiment consists of the quadrotor taking off from the ground, flying and operating as a part of the mission until the mission time expires, then landing on the charge/change station. After the swapping process takes place, the quadrotor takes off and repeats the flight. The quadrotor is given a fully charged battery, and the battery voltage is monitored by the mission manager. All batteries

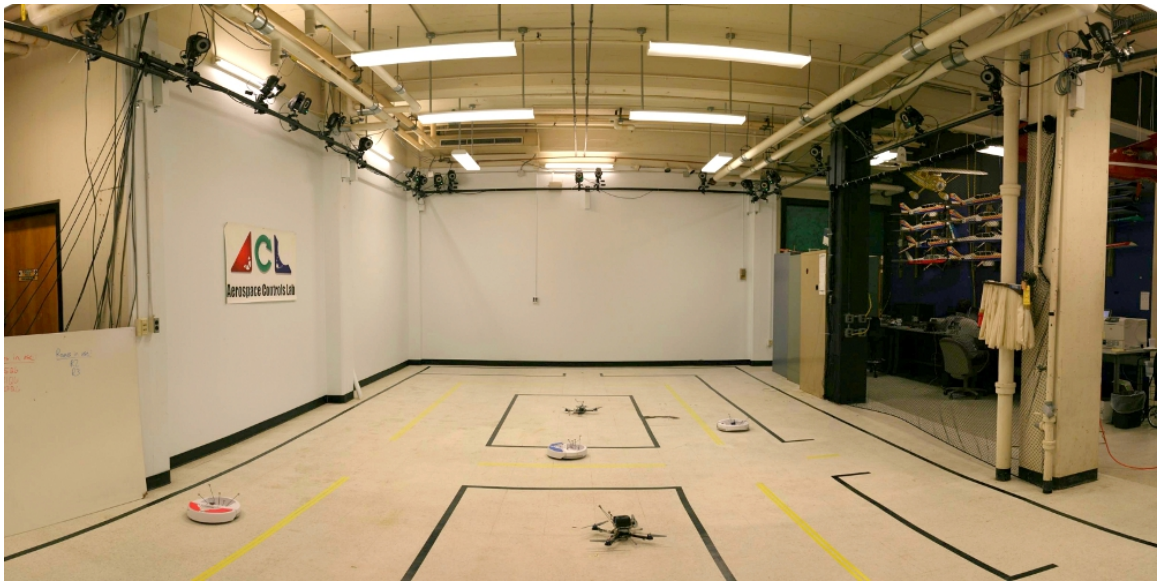


Figure 4-3: The Real-time Autonomous Vehicle test ENvironment (RAVEN) in the Aerospace Controls Lab at MIT [25].

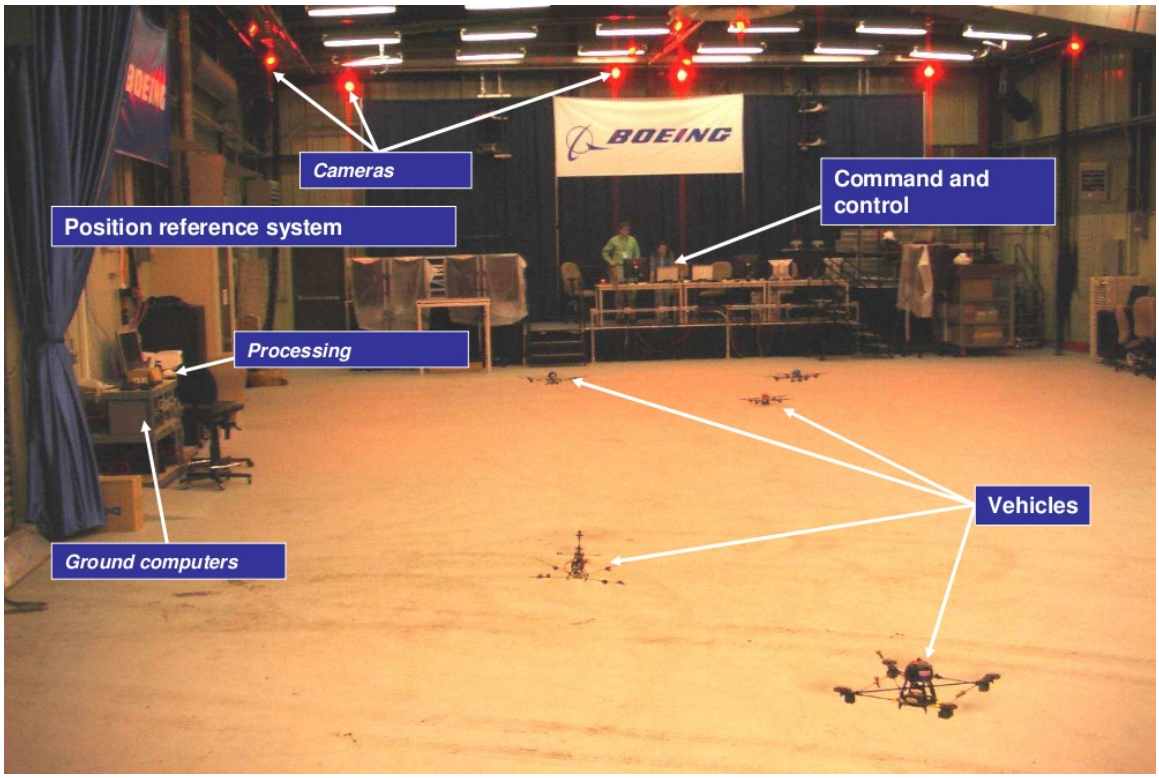


Figure 4-4: Boeings Vehicle Swarm Technology Lab

in the station are also fully charged initially. As the voltage of the on-board battery approaches 10.5 V (there is ≈0.7V voltage drop when the quadrotor is flying due to current flow and the quadrotor impedance), the process outlined above is executed, and the station swaps it with a fresh battery. All other batteries in the recharge station are continuously charging.

Figures 4-5, 4-6 and 4-7 show the results of approximately 5 hour (conducted on September 15, 2011), 4 hour autonomous flight experiments (conducted on January 4, 2012) and 3.5 hours (conducted on April 27, 2012) that cycled through the entire set of batteries several times. The experiments shown in Figures 4-5 and 4-6 were ended intentionally as they were long enough to prove the concept. The experiment shown in Figure 4-7 was ended because of a quadrotor failure. Each peak in the figures indicate that quadrotor had received a fresh battery – and after time $t = 71$ min, the first battery used has already been recovered while the quadrotor is flying. Note that each of the batteries identified by numbers is recovered in the next cycle, since they reach the same battery voltage after charging. This shows that the persistency is ensured.

Figure 4-8 shows the voltages of the batteries held in the recharge station. In the beginning, all batteries are fully charged. The first battery is swapped at about ≈ 8.5 minute. Each drop in a battery voltage indicated that that battery is pushed into the quadrotor while the battery in the quadrotor is pushed into the station for recharging. As it is seen in figures, the battery is charged through constant-current period, and it reaches constant-voltage, $12.6V$, in about $34mins$. The recharge process is finished during constant-voltage period.

Furthermore, note that since each landing, battery swap, and take-off takes approximately 1 min, and the quadrotor flights are 8 min, the system is operating with a vehicle utilization of approximately 90%, which far exceeds the approximately 10% utilization (8 min flight time, 70 min recharge) achieved by the original recharge approach [1, 12]. Moreover, even if the charge time is reduced to approximately 20 min by increasing the charge rate to ≈2 A, the utilization of the original approach would still be limited to about 28% (8 min flight and 20 min recharge).

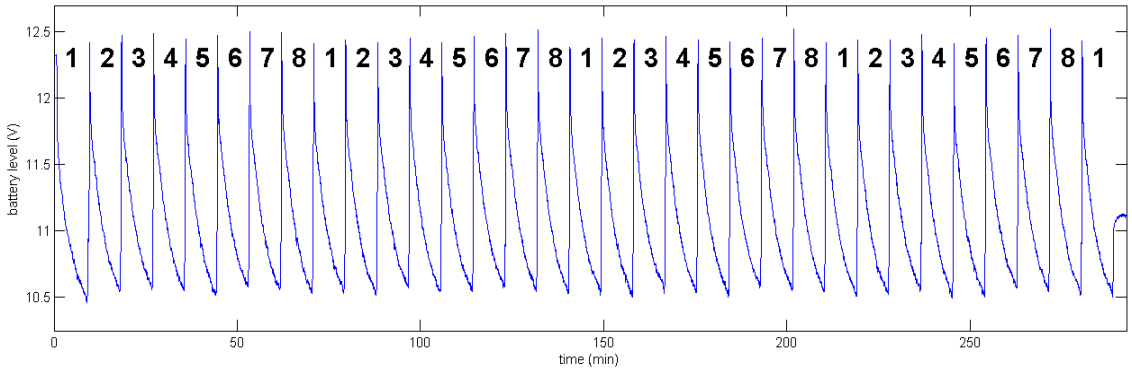


Figure 4-5: The voltage level of the batteries carried by the quadrotor through many battery swaps during ≈ 5-hour take-off land mission

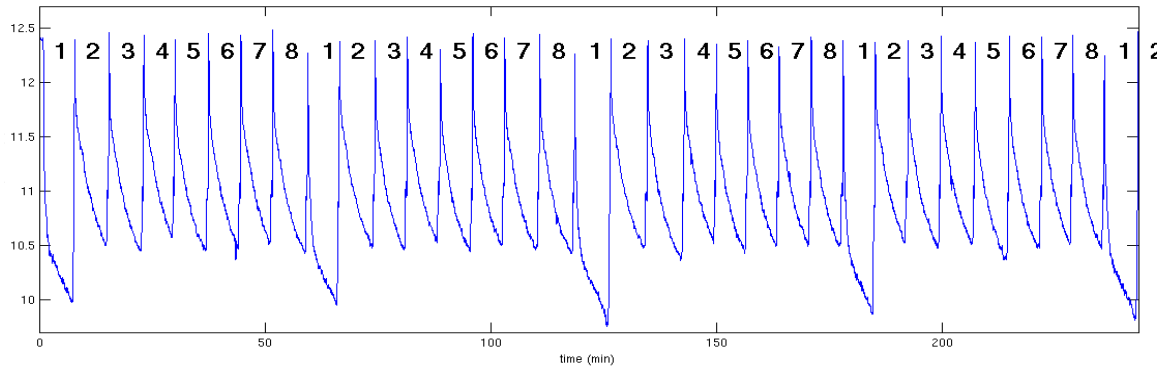


Figure 4-6: The voltage level of the batteries carried by the quadrotor through many battery swaps during ≈ 4-hour car chase mission. The battery labeled with **1** has relatively poor performance, yet, the platform was able to recover it every time it is recharged.

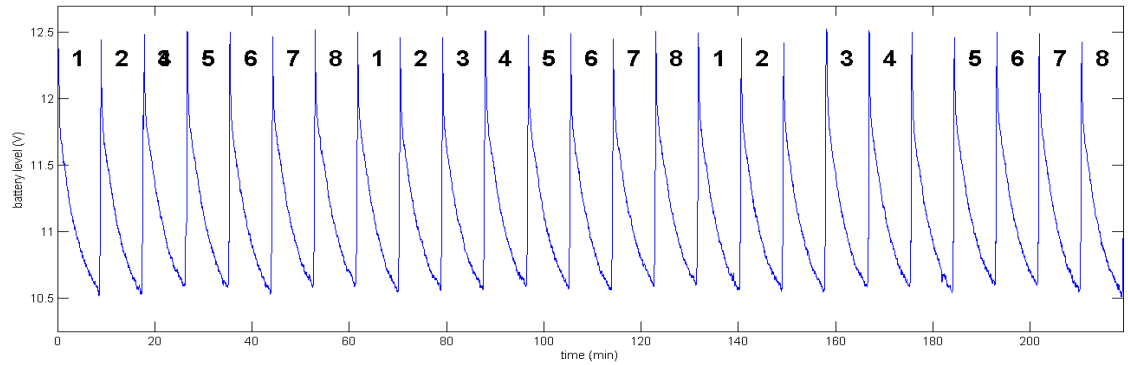


Figure 4-7: The voltage level of the batteries carried by the quadrotor through many battery swaps during ≈ 3.5-hour take-off land mission

Figure 2-5 was obtained through the same 5-hour mission as that of 4-5, and it shows the time it takes to fully charge batteries. On average, it takes 52.02 minutes. Time spent in the station by each battery has a mean of 61.1 minutes. This indicates that the arbitrarily long flights feature is achieved as all batteries had enough time to be recharged to their previous voltage level.

Figure 4-9 shows the collective controller input as it changes for single quadrotor during a relatively long mission. The controller is counteracting to the dropping voltage by increasing the throttle. Around $\approx 8 - 8.5mins$, the voltage reaches to a critical level, and it needs to be swapped. After the battery swap takes place, the integrator is reset, and the controller starts with nominal throttle. In this mission, the quadrotor has swapped its battery 32 times. The resulting downtime due to swapping process is found to be $\approx 5\%$. This measures the ratio the quadrotor were in OFF state vs ON state.

A similar mission to prove capability and feasibility is demonstrated in Boeing's VSTL facility[2]. The mission consisted of 2 quadrotor UAVs developed in MIT ACL, and one RC Tank vehicle. The mission is to chase the ground vehicle with two aerial vehicles. When one of the vehicles ran low on battery, the other one sitting on the recharge platform took off, and the chase task is handed off to this vehicle once two UAVs were within $1.5m$ of each other. The quadrotor is then guided to the recharge station for refueling. After the swap operation is completed, the quadrotor sits until the other one runs low on battery, and this cycle goes on. This mission continued for about $75mins$, until it was ended because of a programming error in the quadrotor controller. Throughout the mission, each battery was used more than twice. The reason each battery is used for short amount of time is that the batteries used in VSTL were old compared to the ones used in ACL, thus they had shorter flight time. However, the observation suggested that each battery, right before they are put back into the quadrotor, was able to reach full voltage, which means sustainability is achieved.

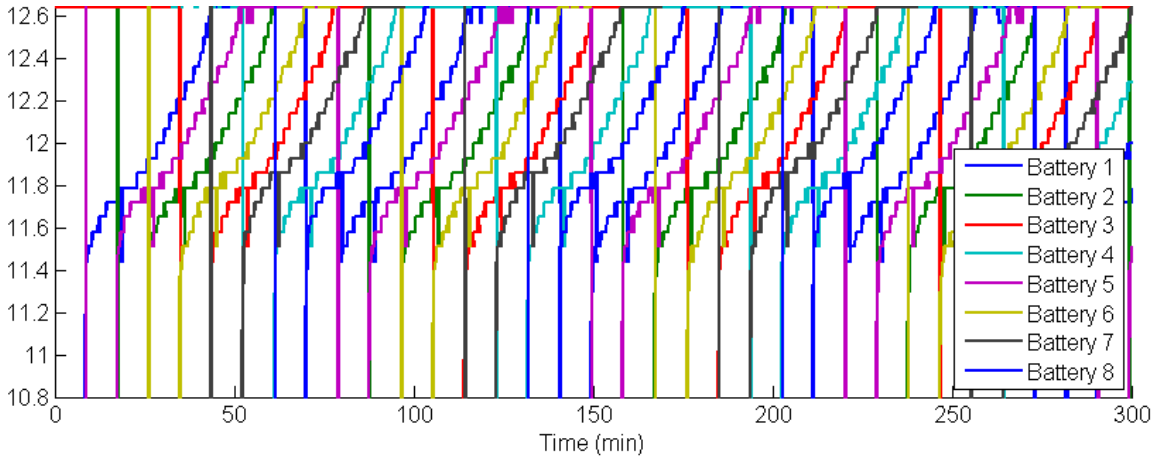


Figure 4-8: Battery potentials as they are being charged in the recharge station through 5-hour experiment. Each segment represents the area where a different battery is used. It took, on average, 52.02 minutes to charge a battery, while each battery spent 61.1 minutes in the recharge station.

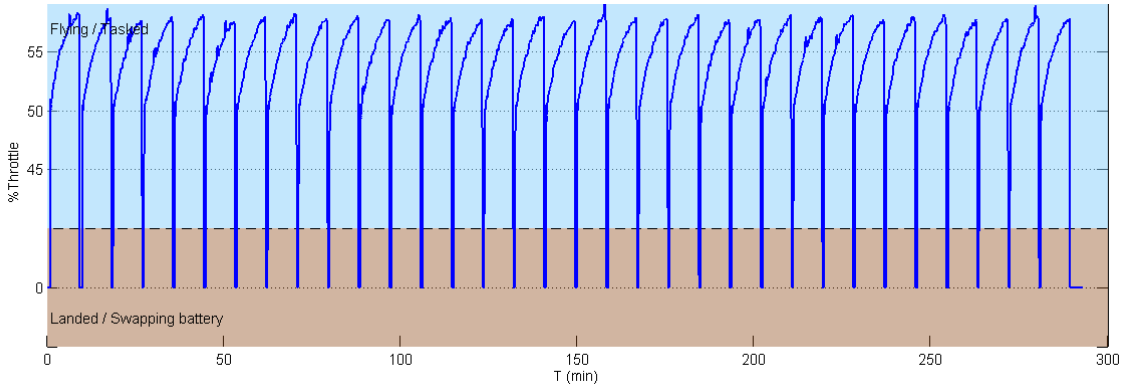


Figure 4-9: Plot of the collective control input of a quadrotor during the mission lasting about 5-hours.

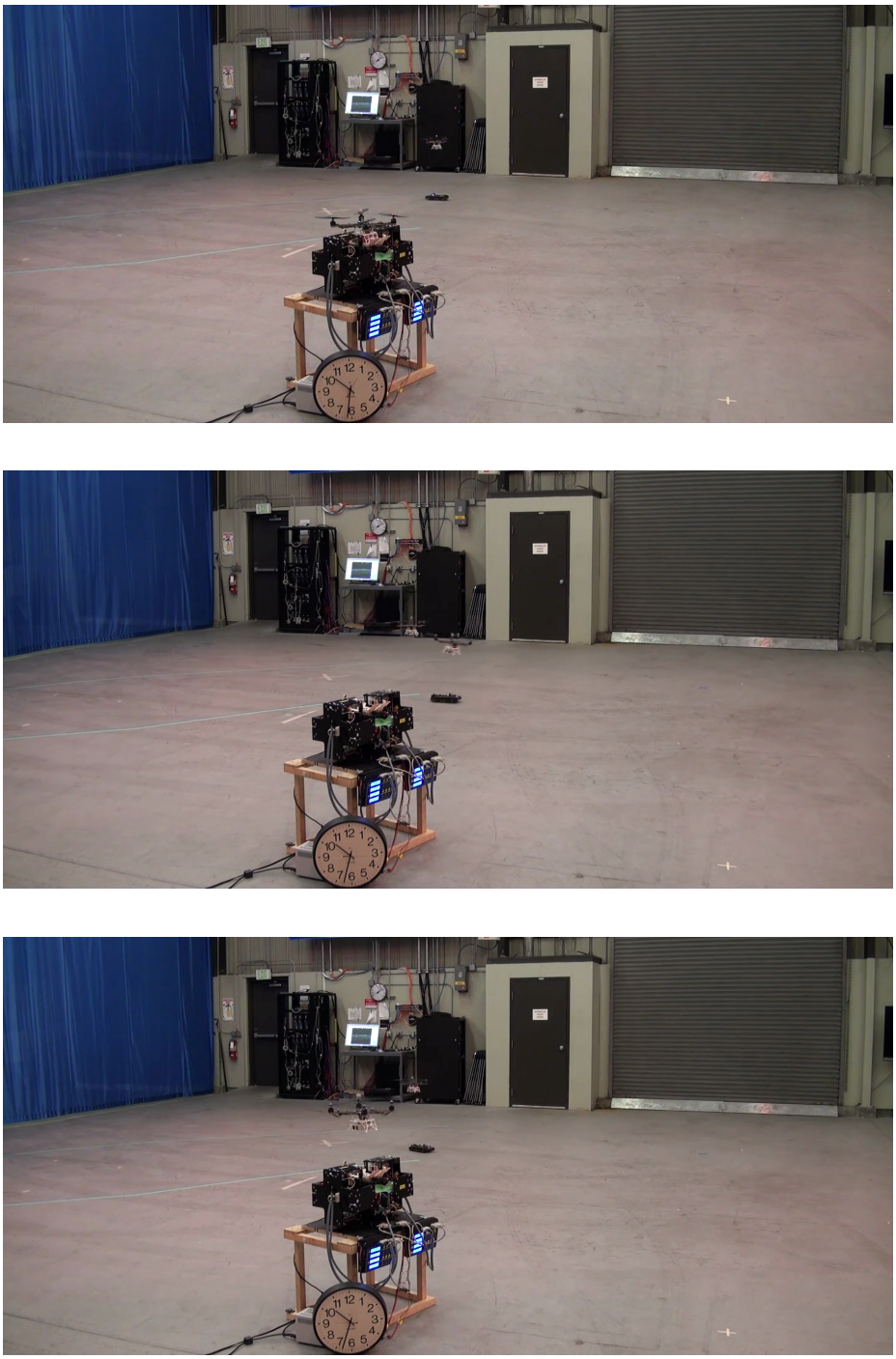


Figure 4-10: Snapshots from a car-chase mission conducted at Boeing facilities. In this mission, 2 UAVs shared one recharge station. One of the vehicles sits on the recharge station until the other one runs low on battery. Then the quadrotors hand off the chase task and refuel.

Chapter 5

Planning With Battery Maintenance

5.1 Background

The purpose of this section is to provide preliminaries for the formulation used in planning with battery maintenance. Markov Decision Processes, Linear Function Approximators and model-free MDP solvers are explained.

5.1.1 Markov Decision Processes

A Markov Decision Process (MDP) [38] is a tuple defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P}^a_{ss'}$ is the probability of getting to state s' by taking action a in state s, $\mathcal{R}^a_{ss'}$ is the corresponding reward, and $\gamma \in [0, 1]$ is a discount factor that balances current and future rewards . A *trajectory* is a sequence $s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$, where the action $a_t \in \mathcal{A}$ is chosen probabilistically according to a *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ mapping each state-action pair to a probability. The agent bases its decision for a given state using the policy. Each step is generated by environment based on the transition model. For every state, the total probability of

all transitions add up to one, i.e. $s \in S$, $\pi(s, .)$ forms a probability distribution:

$$\forall s \in \mathcal{S}, \sum_{a \in \mathcal{A}} \pi(s, a) = 1. \tag{5.1}$$

Given a policy π, the state-action value function, $Q^\pi(s, a)$ of each state-action pair, is the expected sum of the discounted rewards for an agent starting at state s, taking action a, and then following the policy π.

$$Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a \right]. \tag{5.2}$$

In finite discrete spaces, $Q^\pi(s, a)$ can be represented by a table that maps state action pairs to values.

The goal of solving an MDP is to find the optimal policy which maximizes the expected cumulative discounted rewards in all states. In particular, the optimal policy π^* is defined as:

$$\forall s, \pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{\pi^*}(s, a). \tag{5.3}$$

that is, for a given state, the action with the highest value in the value function is picked. The state value function for a given policy π is defined as:

$$V^\pi(s) \triangleq \max_{a \in \mathcal{A}} Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s \right]. \tag{5.4}$$

The optimal value function is defined as:

$$V^*(s) \triangleq V^{\pi^*}(s) = \max_{a \in \mathcal{A}} Q^{\pi^*}(s, a) = Q^{\pi^*}(s, \pi^*(s)). \tag{5.5}$$

The optimal value function satisfies the Bellman equation:

$$\begin{aligned} \forall s \in \mathcal{S} \quad V^*(s) &= \max_{a \in \mathcal{A}} E_{s'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \middle| s' \sim \mathcal{P}^a_s \right] \\ &= \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \right]. \end{aligned} \tag{5.6}$$

5.1.2 Dynamic Programming

Dynamic Programming is a method of simplifying a decision problem by dividing it into sequence of smaller decision steps over time. This is achieved by finding the optimal value function and corresponding policy.

Policy Iteration

The basic idea behind policy iteration[39, 40] is that, once we know the value of each state under current policy, the policy may be improved by changing the first action taken. If there is an improvement, the policy is modified to take that new action whenever it is in that state. This guarantees to improve the performance each time policy is modified.

---

function POLICYITERATION($\mathbf{R}, \mathbf{P}, \gamma$)
  $\pi(s) \leftarrow Random(\mathcal{A})$ **for** $s \in \mathcal{S}$
  `changed` $\leftarrow$ **True**
  while `changed` **do**
    $V^{\pi} \leftarrow (I - \gamma P^{\pi})^{-1} R^{\pi}$
    for $s \in \mathcal{S}$ **do**
      $\pi^{+}(s) \leftarrow \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]$
    end for
    `changed` $\leftarrow (\pi^{+} \neq \pi)$
    $\pi \leftarrow \pi^{+}$
  end whilereturn π
end function

Value Iteration

This is a classic DP algorithm[39, 40] that updates state-action values by visiting all state-space and applying the *Bellman update*

$$Q(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \text{max}_{a'} Q(s', a')\right], \tag{5.7}$$

until no significant change is observed.

function VALUEITERATION($\mathbf{R}, \mathbf{P}, \gamma$)
  $V(s) \leftarrow Random()$ **for** $s \in \mathcal{S}$
  $changed \leftarrow False$
  while not changed **do**
    for $s \in \mathcal{S}$ **do**
      $v \leftarrow V(s)$
      $V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
      $\pi(s) \leftarrow \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
      `changed` $\leftarrow$ `changed` **or** $v \neq V(s)$
    end for
  end while
  return π
end function

5.1.3 Linear Function Approximation

The look-up table representation of the Q function by storing a value for each state-action pair is not practical for problems with large state space. An approximation is usually possible by grouping states into larger groups. The approximation is usually of the form $Q^\pi(s,a) = \theta^T \phi(s,a)$. The feature function $\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^n$ maps each state-action pair to a vector of scalar values. Each element of the feature function $\phi(s,a)$ is called a *feature*; $\phi_f(s,a) = c \in \mathbb{R}$ denotes that feature f has scalar value c for state-action pair (s,a). The vector $\theta \in \mathbb{R}^n$ As it can be noted in Equation 5.5, finding the optimal policy requires the ranking of the Q values for a given state. It is often a good practice to avoid approximating the value of $Q^\pi(s,a)$ based on $Q^\pi(s,a')$ where $a \neq a'$. The approach to do this is to map each state to a set of features and create a vector with these features copied to the appropriate action slot, and set remaining elements to 0. This process is demonstrated in the following example with 2 actions and 2 features.

$$\phi(s) = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \rightarrow \phi(s,a_1) = \begin{bmatrix} \phi_1 \\ \phi_2 \\ 0 \\ 0 \end{bmatrix}, \phi(s,a_2) = \begin{bmatrix} 0 \\ 0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.8}$$

Adaptive Function Approximation using iFDD

Adaptive Function Approximators, aside from updating the weights of the θ, also modify the set of the features based on the observed data using the following update rule:

$$\begin{aligned} \hat{Q}^{k,l}(s) &= \phi^{k,l}(s)^\top \theta^{k,l}, \\ \phi^{k+1,l+1}(s) &= h(z^k, \theta^{k,l}, \phi^{k,l}), \end{aligned} \tag{5.9}$$

where h is the representation expansion function that adds new features to the feature vector based on sampled trajectories, weight vector, and previous set of features. Based on the successful results in representing value functions, iFDD [39, 41] is used as the adaptive function approximator for our framework to represent the uncertainty. The idea behind iFDD is to expand the feature representation by adding conjunctions of a given set of initial features based on temporal difference error, thus reducing the error in parts of the state space where the feedback error persists.

5.1.4 Approximate Dynamic Programming

The idea of approximate dynamic programming (ADP) is to approximate the value function by representing in a lower dimensional space using $n \ll |\boldsymbol{S}|$ parameters. In this thesis, the focus is on family of approximators, particularly on IFDD, as explained in Section 5.1.3. The main reason IFDD is used in this thesis is that the quality of approximation is strongly related to the features selected, and selection of these features is a hard problem. Using a TD-error threshold based method, IFDD incrementally expands the representation, thus increasing the power of approximation while still being computationally tractable.

5.1.5 Reinforcement Learning

In most practical domains, the dynamics of the system (i.e. $\mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}$) are too complex to express analytically or most often unknown. Using exact Dynamic Programming

methods explained in Section 5.1.2 is infeasible or most of the times not possible. In contrast, Reinforcement Learning techniques do not need the exact knowledge of the MDP. Algorithms solve the MDPs with unknown models by interacting with the environment at each time step using a deterministic policy.

RL methods, like model-based MDP solvers, can be grouped into 2 categories namely 1) Value-Based Methods and 2) Policy Search techniques. This thesis makes use of Sarsa and an improvement on it called Trajectory Based Value Iteration which belongs to the former category.

Trajectory Based Value Iteration (TBVI)

This algorithm focuses on applying the Bellman updates on trajectories that are sampled through Monte-Carlo simulations. The policy used for generating trajectories are $\epsilon - greedy$ with respect to the current value function:

$$\pi^{\epsilon}(s,a) = \begin{cases} a = \text{argmax}_a\, Q(s,a) & \text{with probability } 1-\epsilon \\ a = \text{random action} & \text{otherwise} \end{cases}$$

The random action selection with ϵ probability ensures that in the limit, all states are updated infinitely, which guarantees convergence to the optimal value function.

In an exact DP algorithm, such as Policy Iteration and Value Iteration, all state-action pairs (s,a) need to be updated. However, this is not computationally tractable for problems with millions of states. This has driven researchers to consider Asynchronous Dynamic Programming [42], which updates only a subset of (s,a) pairs. By using trajectories produced through Monte-Carlo simulations, TBVI updates only most-frequently seen (s,a) pairs.

Algorithm 5.1 Generate Sparse Feature Vector

```
function GetExtendedFeatures(φ⁰(s), χ ⊆ F_n)
    φ̊(s) ← 0̄
    activeInitialFeatures ← {i|φ⁰_i(s) = 1}
    Candidates ← SortedPowerSet(activeInitialFeatures)
    while activeInitialFeatures ≠ ∅ do
        f ← Candidates.next()
        if f ∈ χ then
            activeInitialFeatures ← activeInitialFeatures ∖ f
            φ̊_f(s) ← 1
        end if
    end whilereturn φ̊(s)
end function
```

5.2 Reducing Computational Complexity of iFDD using Caching

iFDD relies on feature sparsification, as explained in [39]. In summary, in order to create a sparse set of features, a greedy set covering algorithm is employed. Using the greedy set covering algorithm [43], the complexity of algorithm is reduced to polynomial time. The method works as follows: Given an initial feature vector $\phi^0(s)$ and the current pool of features χ, the resulting features are found by identifying the active initial features and calculating the power set sorted by the set sizes. Every set in the power set is then compared with the initial active features and the set is taken if all elements are included in the power set, and then the elements in the set are removed from the initial active feature set. The algorithm is explained more formally in Algorithm 5.1.

There is still an improvement possible to speed up the sparsification process by introducing caching mechanisms from computer science literature [44, 45] . A cache is a mechanism that stores data produced as a result of computationally expensive process for faster access. Usually, they are implemented through the use of hash maps, where each element (known as *keys*) is mapped to another element (known as *values*). They make use of hash function to generate an index known as the index, through the use of *hash functions*.

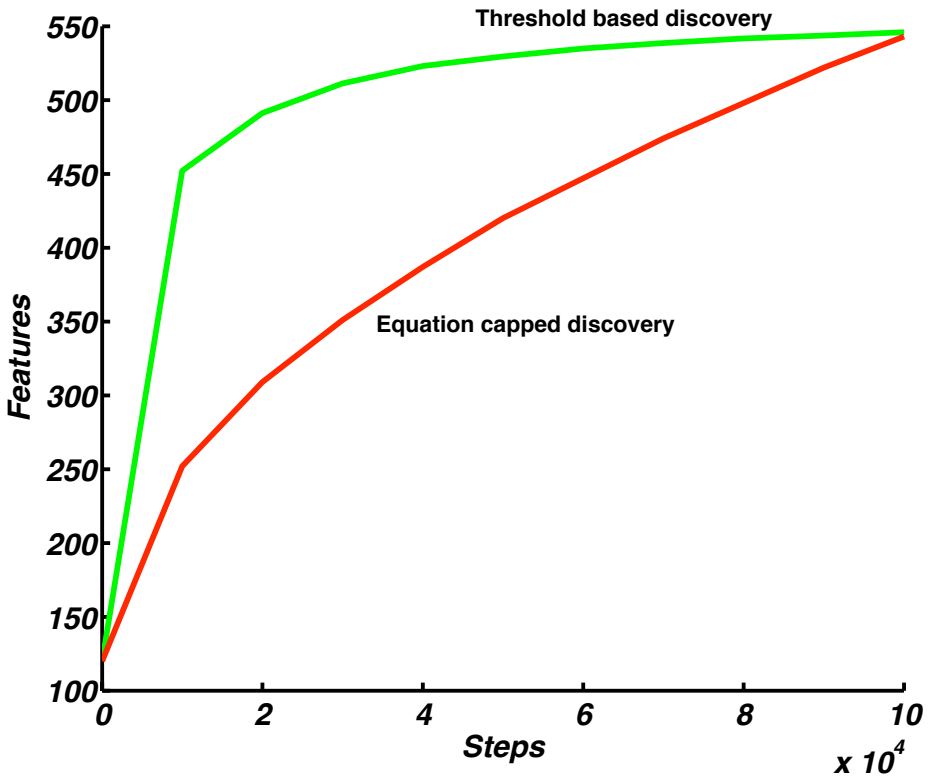


Figure 5-1: The rate of number of features discovered decreases as the algorithm proceeds. The slow down in the feature discovery allows to represent the value function with bounded number of features, which also limits the memory consumption.

One important aspect of every cache implementation is cache invalidation, in other words, there needs to be a way to invalidate cache once new information is available so that fresh information is served as opposed to stale cached one. In iFDD, the invalidation needs to happen when new features are discovered. The performance improvement comes from the fact that the number of features discovered in each timestep decreases as the algorithm proceeds since the algorithm works using a threshold of temporal difference error, and as the algorithm proceeds the TD error will reduce.

This situation is shown in Figure 5-1. In the limit, the number of features found will converge to the number of states. Moreover, even when new features are discovered, most of the previous features are not affected by this change since the current set will likely not intersect with the other feature sets in the cache. When a new feature is expanded, features that include the basic features represented by the two parent feature need to be invalidated.

For problems with small stochasticity, i.e., small probability of visiting unseen states when sampling, the cache size will remain small as the probability of seeing new states will be slim. Algorithms 5.2 and 5.3 explain caching and cache invalidation.

Algorithm 5.2 Generate Sparse Feature Vector with Caching

function GETEXTENDEDFEATURESCACHE($\phi^0(s), \chi \subseteq \mathcal{F}_n$)
  if $\sim cache.HasKey(\phi^0(s))$ **then**
    $cache[\phi^0(s)] \leftarrow GetExtendedFeatures(\phi^0(s), \chi)$
    for all $f \in \chi$ **do**
      $dependency[f] \leftarrow dependency[f] \cup \{\chi\}$
    end for
  end if
  return $cache[\phi^0(s)]$
end function

Algorithm 5.3 Cache Invalidation Using Newly Discovered Features

function INVALIDATECACHE($\hat{\phi(s)}$)
  for all $f_{new} \in \hat{\phi(s)}$ **do**
    $p_0 \leftarrow f_{new}.parent_0$
    $p_1 \leftarrow f_{new}.parent_1$
    for all $f \in p_0 \cup p_1$ **do**
      for all $f_{dependent} independency[f]$ **do**
        $cache.removeKey(f_{dependent})$
      end for
      $dependency.removeKey(f)$
    end for
  end for
end function

A particular statistic of significant importance about caches is miss/request ratio. This statistic signifies the success of implemented cache mechanism. The lower this ratio is, the faster the access will be. Figure 5-2 shows a sample execution of TBVI on PSM domain with 70 episodes, and Figure 5-3 shows the improvement in the processing time. The cache miss ratio in Figure 5-2 reaches to a steady state as the feature discovery also slows down. This steady state value of $\leq 2\%$ is typical for most domains. Reducing the amount of unnecessary calculation by increasing cache hit ratio, the performance improved around $40 times$ on this example problem.

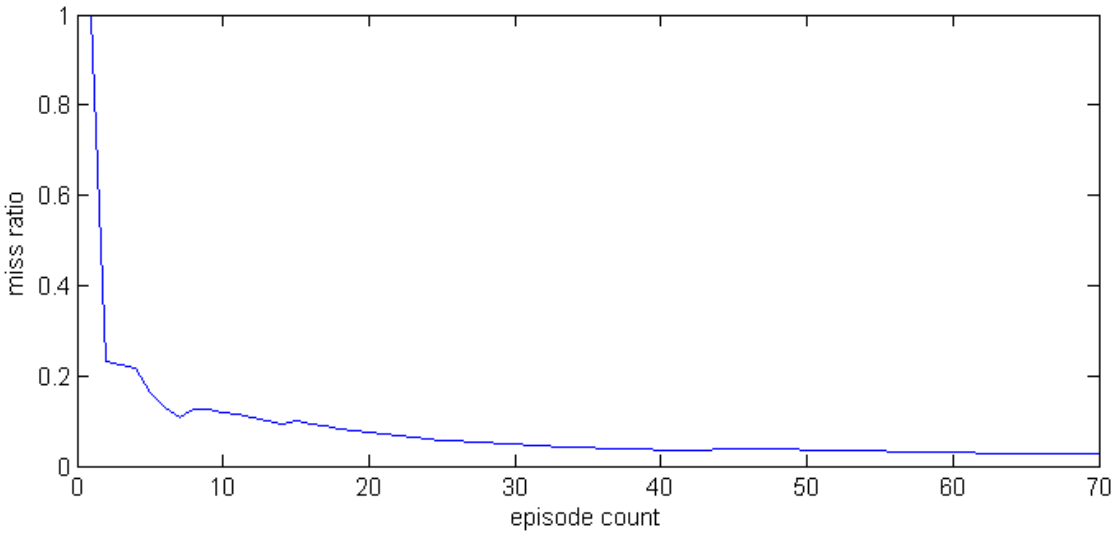


Figure 5-2: Cache miss ratio of the implemented technique for PSM domain

Figure 5-3: Time required to process 25000 episodes with and without cache

5.3 Proactive Planning with Battery Health Information

With the introduction of the battery maintenance platform in Chapters 2 and 3, more information about the actual system dynamics is available, and the MDP model needs to be modified to more realistically reflect the actual dynamics due to following reasons:

- Batteries have different potential levels at any given time.
- Battery potential and the flight time it can provide has nonlinear relation. The charging is also nonlinear due to CC-CV battery charging scheme.
- Batteries should not be discharged to less than 20% percent.
- Battery performances may be different from each other.

This thesis addresses these issues in the context of the PSM problem.

5.3.1 Persistent Search and Track Mission

The persistent search and track mission (PSM) is a multi-agent mission planning problem where a number of UAVs perform surveillance on a group of targets, while maintaining communication and health constraints [16]. The high-level mission outline is shown in Figure 5-4.

Each UAV's individual state at time $t = t_j$ is a tuple of 3 components:

$$s(a_i, t_j) = (L_{a_i,t_j}, F_{a_i,t_j}, H_{actuator_{a_i,t_j}}, H_{sensor_{a_i,t_j}}) \tag{5.10}$$

where L_{a_i} denotes the location of agent a_i, and is described by a discrete set of locations:

$$L_{a_i} \in \{Base, Communication, Tasking\} \tag{5.11}$$

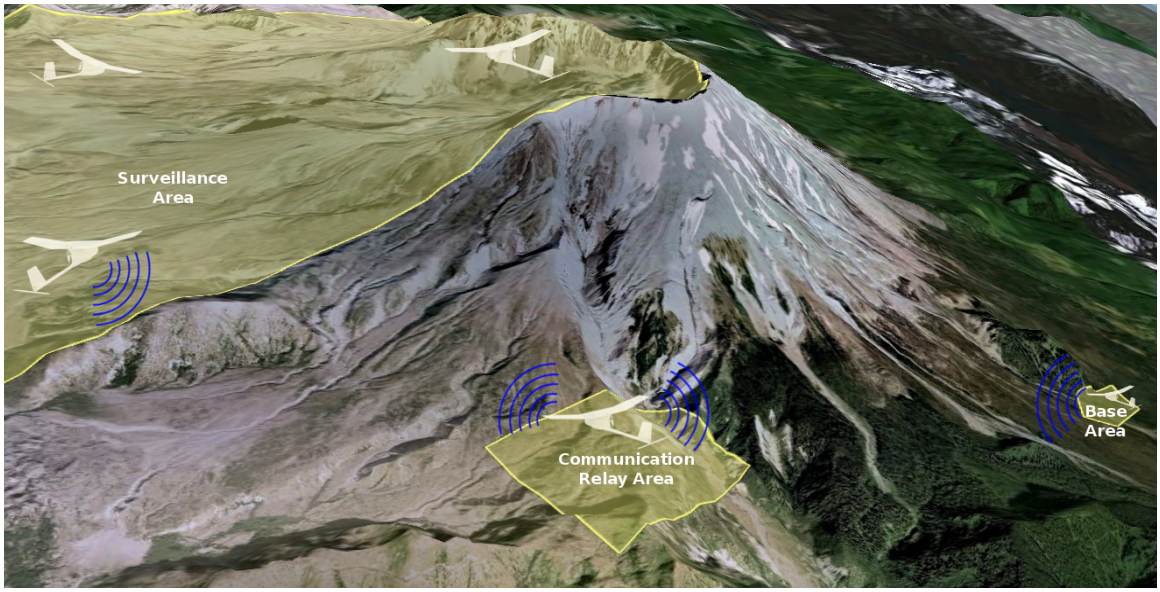


Figure 5-4: The objective of persistent search and track mission is to have as many agents as required in the tasking area, while maintaining a communication link in the relay area. Each agent's state is subject to uncertainty in actuator,sensor health and fuel consumption. The objective is to maximize the coverage in the tasking area while preventing agents from crashing.

F_{a_i} denotes the fuel level of agent a_i, and is described by a continuous set

$$F_{a_i} \in [0, Fuel_{max}] \tag{5.12}$$

$H_{actuator_{a_i}}$ denotes the actuator health status of agent a_i, and has values

$$H_{actuator_{a_i}} \in \{Healthy, Failure\} \tag{5.13}$$

Similarly, $H_{sensor_{a_i}}$ denotes the sensor health and its domain is given by

$$H_{sensor_{a_i}} \in \{Healthy, Failure\} \tag{5.14}$$

The state-space of the whole problem is combination of the states of each agent. There are three available actions for each UAV: $\{Advance, Retreat, Loiter\}$. The objective of the mission is to keep as many UAVs as possible in the Tasking area,

while one UAV remains in the Communication area to provide the data link between Tasking area and the Base area. Each UAV starts with $Fuel_{max}$. Fuel burn doesn't have a deterministic model as it depends on external conditions such as weather conditions and internal conditions such as the maneuvers made. Hence, it is modeled using discrete Bernoulli Distribution [46]. The vehicle burns one unit of fuel for all actions with probability p_{nom} and 2 units with probability $1 - p_{nom}$. A UAV with failed sensor cannot perform surveillance whereas a UAV with failed actuator cannot perform neither surveillance nor communication. If a UAV runs out of fuel, it crashes and can no longer continue the mission. When a UAV returns to the base, its failures are repaired.

The state-transition model $\mathcal{P}^a_{ss'}$ captures the mission dynamics and is defined as follows. Each agent's location at the next time step depends on the current location and the action taken, and it is deterministic in nature.

$$L_{a_i,t_{j+1}} = \begin{cases} L_{a_i,t_j}, & \text{if } F_{a_i,t_j} = 0 \text{ or } u_{a_i,t_j} = Loiter \\ Base, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} = Retreat \\ Tasking, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} = Advance \\ Communication, & \text{if } L_{a_i,t_j} = Tasking \text{ and } u_{a_i,t_j} = Retreat \\ Communication, & \text{if } L_{a_i,t_j} = Base \text{ and } u_{a_i,t_j} = Advance \end{cases} \tag{5.15}$$

Each agent's fuel at the next time step is stochastic with parameter p_{fuel} representing the probability of burning fuel at the nominal rate of 1. With probability $1 - p_{fuel}$, the fuel reduces by 2.

$$F_{a_i,t_{j+1}} = \begin{cases} 0, & \text{if } F_{a_i,t_j} = 0 \\ F_{max}, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} = Retreat \\ F_{a_i,t_j} - 1, & \text{if } L_{a_i,t_j} = Comm \text{ or } L_{a_i,t_j} = Tasking \quad Prob = p_{fuel} \\ F_{a_i,t_j} - 2, & \text{if } L_{a_i,t_j} = Comm \text{ or } L_{a_i,t_j} = Tasking \quad Prob = 1 - p_{fuel} \end{cases} \tag{5.16}$$

The actuator and sensor health are modeled using a discrete probability distribution with 2 outcomes:Failure or RemainSame. The uncertainty could also be modeled as using state dependent uncertainty [47, 48], but for simplicity,it is modeled using Bernoulli distribution [46].

$$H_{actuator_{a_i,t_{j+1}}} = \begin{cases} Healthy, & \text{if } L_{a_i,t_j} = Comm. \text{ and } u_{a_i,t_j} = Retreat \\ Failure, & \text{if } \quad Prob = p_{actuatorfail} \\ H_{actuator_{a_i,t_j}} & \text{otherwise} \end{cases} \tag{5.17}$$

$$H_{sensor_{a_i,t_{j+1}}} = \begin{cases} Healthy, & \text{if } L_{a_i,t_j} = Comm. \text{ and } u_{a_i,t_j} = Retreat \\ Failure, & \text{if } \quad Prob = p_{sensorfail} \\ H_{sensor_{a_i,t_j}} & \text{otherwise} \end{cases} \tag{5.18}$$

The available actions set for a given state is defined as follows:

$$u_{a_i,t_j}(s_j) \in \begin{cases} \{Advance\}, & \text{if } L_{a_i,t_j} = Base \\ \{Advance, Retreat, Loiter\}, & \text{if } L_{a_i,t_j} = Communication \\ \{Retreat, Loiter\}, & \text{if } L_{a_i,t_j} = Tasking \end{cases} \tag{5.19}$$

The overall state transition is given in Algorithm 5.4

The cost function $g(s_t, u_t, u_{t+1})$ is chosen such that any favorable outcome is rewarded while unfavorable ones are punished. For the PSM mission, favorable outcomes include: (1) Having as many agents in surveillance area as possible while there is a communication link. Unfavorable outcomes include: (1) Having crashed vehicles. Moreover, each UAV move incurs additional cost C_{move}.

$$g(s_t, u_t, u_{t+1}) = N_{tasking} C_{coverage} CommLink + C_{crash} N_{crashed} + C_{move} N_{move} \tag{5.20}$$

Algorithm 5.4 State Transition Routine

function sampleNextState($currentState$, $action$)
  for $i \in \{0, .., N_{agent}\}$ **do**
    $L_{a_i,t+1} \leftarrow AdvanceLocation(F_{a_i,t}, L_{a_i,t}, u_i)$ ▷ Set the location
    if $random() \leq p_{fuel}$ **then**
      $F_{a_i,t+1} \leftarrow F_{a_i,t} - 1$
    else
      $F_{a_i,t+1} \leftarrow F_{a_i,t} - 2$
    end if
    $H_{actuator,a_i,t+1} \leftarrow AdvanceActuatorHealth(H_{actuator,a_i,t}, p_{actuatorfail}, L_{a_i,t+1})$
    $H_{sensor,a_i,t+1} \leftarrow AdvanceSensorHealth(H_{sensor,a_i,t}, p_{sensorfail}, L_{a_i,t+1})$
    if $L_{a_i,t+1} = Base$ **then**
      $F_{a_i,t+1} \leftarrow F_{max}$
    end if
    $F_{a_i,t+1} \leftarrow saturate(F_{a_i,t+1}, 0, F_{max})$
  end for
end function

Algorithm 5.5 Location Transition Routine

function AdvanceLocation($fuel$, $location$, $action$)
  if $location = Base$ and $action = Advance$ **then**
    return $Communication$
  else if $location = Communication$ and $action = Retreat$ **then**
    return $Base$
  else if $location = Communication$ and $action = Advance$ **then**
    return $Tasking$
  else if $location = Tasking$ and $action = Retreat$ **then**
    return $Base$
  else if $action = Loiter$ **then**
    return $location$
  end if
end function

Algorithm 5.6 Actuator Health Transition Routine

function AdvanceActuatorHealth($actuatorHealth$, $p_{failure}$, $nextLocation$)
  if $random() \leq p_{failure}$ **then**
    $actuatorHealth \leftarrow Failure$
  end if
  if $nextLocation = Base$ **then**
    $actuatorHealth \leftarrow Healthy$
  end if
end function

Algorithm 5.7 Sensor Health Transition Routine

```
1: function AdvanceSensorHealth(sensorHealth, p_failure, nextLocation)
2:     if random() ≤ p_failure then
3:         sensorHealth ← Failure
4:     end if
5:     if nextLocation = Base then
6:         sensorHealth ← Healthy
7:     end if
8: end function
```

5.3.2 Modifications to Incorporate Battery State

These transition and reward functions do not take into account of the individual battery states, and they always assume that going back to base will ensure a fully charged battery. This is not necessarily true, as the best battery at the platform might not have completed the recharge cycle, and its voltage level might be less than F_{max} fuel. The transition and reward functions also do not take into account individual battery states, and the nonlinear charging rate due to battery safety and CC-CV charging scheme as discussed in Section 3.1 and Section 2.4. When these dynamics are not considered during planning, the resulting policy may not necessarily have persistency. The battery will be driven below safe levels as the sole purpose is to keep the quadrotor in the tasking area as long as possible, and the time it arrives to base for refuel will be just above 0 to prevent it from crashing. Driving battery that low will require slower recharge, hence lengthening the charge time. The former dynamics also assumes the battery will be fully charged, which isn't necessarily true.

To corporate the battery states, the MMDP state is modified to include charger states. Each charger state is composed of individual voltage level of batteries in the drums scaled to the range $[0, 10]$

$$S_{c_i,t} = (b_{c_i,0,t}, ..., b_{c_i,m-1,t}) \tag{5.21}$$

where c_i is the charger index, t is time, and m is the number of batteries per charger.

The overall problem state could be modeled as follows:

$$S_t = (s_{a_0,t}, s_{a_1,t}, ..., s_{a_{n-1},t}, s_{c_0,t}, ..., s_{c_{m-1},t}) \tag{5.22}$$

where n is the number of available agents, and m is the number of recharge stations, and t is the time indicator, $s_{a_x,T}$ is agent x's state with fuel, location, sensor and actuator health information, s_{c_x} is recharge station x's state which is composed of individual battery levels.

As the first step, the action space is modified (Eq. 5.23) to include $Recharge_x$ as an operation where x is the index of the charger. An agent is only allowed to invoke $Recharge_x$ when it is in Communication area, and at a particular time step agents can swap batteries only from different chargers, giving

$$u_{a_i,t_j}(s_j) \in \begin{cases} \{Advance\}, & \text{if } L_{a_i,t_j} = Base \\ \{Advance, Retreat, Loiter, R_{c_0}, ..., R_{c_{m-1}}\}, & \text{if } L_{a_i,t_j} = Comm \\ \{Retreat, Loiter\}, & \text{if } L_{a_i,t_j} = Tasking \end{cases} \tag{5.23}$$

where $R_{c_{m-1}}$ is the action for going to battery change station c_{m-1}.

The state transition is modified to include (1) the possibility to have a lower battery potential than F_{max}, and (2) increasing individual battery potential due to charging to better reflect real system dynamics. Eq. 5.24 shows the modification one

to the vehicle fuel transition.

$$F_{a_i,t_{j+1}} = \begin{cases} 0, & \text{if } F_{a_i,t_j} = 0 \\ F_{a_i,t_j} - 1, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} \neq Recharge_x \quad Prob = p_{fuel} \\ F_{a_i,t_j} - 2, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} \neq Recharge_x \quad Prob = 1 - p_{fuel} \\ F_{a_i,t_j} - 1, & \text{if } L_{a_i,t_j} = Tasking \quad Prob = p_{fuel} \\ F_{a_i,t_j} - 2, & \text{if } L_{a_i,t_j} = Tasking \quad Prob = 1 - p_{fuel} \\ F_{t_j,b_{x_max}}, & \text{if } L_{a_i,t_j} = Comm \text{ and } u_{a_i,t_j} = Recharge_x \end{cases} \tag{5.24}$$

where $F_{t_j,b_{x_{max}}} = \max(b_{c_x,0,t_j}, ..., b_{c_x,n-1,t_j})$, i.e. the battery with highest potential. For each agent, $Recharge_{a_i,t_j} \neq Recharge_{a_k,t_j}$, that is they cannot go to the same recharge station at the same time.

The health transition is kept the same as in Eq. 5.17 and Eq. 5.18, as they are independent of the battery state. Although the battery state transition could be learned for each battery using state-dependent uncertainty learning techniques given in [47, 48], but for simplicity, the state transition (when there is no swap operation taking place) is assumed to be of the following form:

$$b_{c_i,n,t_{j+1}} = \begin{cases} b_{c_i,n,t_j} + 0.3b_{increment}, & \text{if } b_{c_i,n,t_j} < 2 \\ b_{c_i,n,t_j} + b_{increment}, & \text{if } b_{c_i,n,t_j} \geq 2 \end{cases} \tag{5.25}$$

where $b_{increment} = T_{charge}/T_{flight} = 53\text{min}/8.5\text{min}$. The reasoning behind this is that when the battery is drained more than 20%, it needs to be charged really slowly to keep the battery healthy. This should not happen often, and depending on the mission objective, it should be allowed only very rarely. If there is a need to incorporate state-dependent uncertainty in battery transition, the learner could learn the behavior, and this learned behavior could be used in trajectory generation that is fed into TBVI.

When a battery swap occurs, the battery that is swapped is replaced with the battery in the vehicle, and battery transition also reflects that. The decision of when to go to recharge station plays the most important role in persistency. All other

components contribute to the total cumulative reward. In order to have persistency in the system, the total energy consumed by the system should be greater than the energy put into the system. The exception is when one or more battery is fully charged. In that case the battery is assumed to take hypothetical over-charge to ease the calculations.

The reward function Eq. 5.20 is modified to include the change of energy in the system. When the system is losing energy, a negative reward is given. When the system is gaining energy, a positive reward is given

$$g(s_t, u_t, u_{t+1}) = N_{tasking} C_{coverage} CommLink + C_{crash} N_{crashed} + C_{move} N_{move} + E_{diff} C_{diff} \tag{5.26}$$

where

$$E_{diff}(t) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} b_{c_i,j,t+1} - \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} b_{c_i,j,t} \tag{5.27}$$

The overall state transition logic is given in Algorithm 5.8, and it is similar to Algorithm 5.4. Changes are highlighted. Lines 18-20, basically notes the difference from Algorithm 5.4. Line 18 finds the index and the level of the best battery and assigns to V_{max} and I_{max} respectively. The battery of the agent a_i is swapped with the the best battery in lines 19 and 20.

5.3.3 Simulation Results

Below are several results obtained through algorithm simulations. The simulations were done in an in-house MDP solver framework that supported many MDP solvers, function approximators and different domains.

There are several metrics defined to measure the system performance. The simulation started from an initial state, and fed the possible state trajectories to the MDP solver TBVI defined in Section 5.1.5 Using ϵ-greedy exploration approach, random actions were selected to explore unseen states. The simulation simulated 3-agent state transitions to learn the optimal policy.

Algorithm 5.8 State Transition Routine

function SAMPLENEXTSTATE($currentState$, $action$)
  for $i \in \{0, .., N_{charger} - 1\}$ **do** ▷ For all chargers
    for $j \in \{0, .., N_{battery}\}$ **do** ▷ For each battery in charger c_i
      $b_{c_i,j,t+1} \leftarrow b_{c_i,j,t} + getRate(b_{c_i,j,t})$ ▷ increase battery potential
    end for
  end for
  for $i \in \{0, .., N_{agent}\}$ **do**
    $L_{a_i,t+1} \leftarrow AdvanceLocation(F_{a_i,t}, L_{a_i,t}, u_i)$ ▷ Set the location
    if $random() \leq p_{fuel}$ **then**
      $F_{a_i,t+1} \leftarrow F_{a_i,t} - 1$
    else
      $F_{a_i,t+1} \leftarrow F_{a_i,t} - 2$
    end if
    $H_{actuator,a_i,t+1} \leftarrow AdvanceActuatorHealth(H_{actuator,a_i,t}, p_{actuatorfail}, L_{a_i,t+1})$
    $H_{sensor,a_i,t+1} \leftarrow AdvanceSensorHealth(H_{sensor,a_i,t}, p_{sensorfail}, L_{a_i,t+1})$
    if $L_{a_i,t+1} = Base$ **then**
      if $u_i = Recharge_x$ and $F_{a_i,t+1} \neq 0$ **then**
        $V_{max}, I_{max} \leftarrow max(b_{c_x,0,t}, ..., b_{c_x,m-1,t})$
        $b_{c_x,I_{max},t+1} \leftarrow F_{a_i,t+1}$
        $F_{a_i,t+1} \leftarrow V_{max}$
      end if
    end if
    $F_{a_i,t+1} \leftarrow saturate(F_{a_i,t+1}, 0, F_{max})$
  end for
end function

Algorithm 5.9 Location Transition Routine

function ADVANCELOCATION($fuel$, $location$, $action$)
  if $location = Base$ and $action = Advance$ **then**
    return $Communication$
  else if $location = Communication$ and $action = Advance$ **then**
    return $Tasking$
  else if $location = Communication$ and $action = Recharge_x$ **then**
    return $Base$
  else if $location = Tasking$ and $action = Retreat$ **then**
    return $Base$
  else if $action = Loiter$ **then**
    return $location$
  end if
end function

- Cumulative reward for the mission
- Average battery potential increase in system each time step
- Average battery voltage before being sent to recharge

The policy obtained through the execution of TBVI is compared against the heuristic policy defined as in Table 5.1 and Table 5.2. Basically, based on which agent (indicated by different rows) is querying policy and based on its location (indicated by columns), the policy will execute the if block in the corresponding cell. In a single agent scenario in Table 5.1, since it is not possible to satisfy communication requirement and have as many vehicles as possible in the tasking area at the same time, the communication constrained is removed. If the vehicle is in *Base* region, and has positive fuel, it is commanded to transition into *Communication* area. In communication area, if the fuel level is < 4, there is no need to be present in the communication area. If fuel level is ≤ 5, transitioning into $Tasking$ region is dangerous, since by the time it will go to *Base* for recharge, it will have $F_{a_i} \leq 2$, so it is sent to $Recharge$. Similarly, the agent is commanded to incrementally go to *Base* for recharge if $F_{a_i} < 5$.

Figure 5-5 shows the cumulative reward obtained through the execution of 1) policy obtained through TBVI and 2) Heuristic policy. After about 23000 steps of TBVI, which took about 15 minutes, TBVI produces better policy than the heuristic. This performance is really good given the state space of that size. When the policy produced is examined, it is seen that heuristic is relatively conservative since it always tries to stay above fuel level 2 for each battery. The TBVI policy allows the agent to go to below fuel level 2 if that still results in positive overall voltage change in the system thus stays persistent. By allowing to stay longer in the tasking area, the overall coverage in the tasking area is increased. Figure 5-6 shows average battery level when vehicles are called back from Communication area. The heuristic policy always calls the vehicle back when the fuel level is 3, so that the next time step when it is at Base, the fuel level is still ≥ 2. The policy produced by the TBVI will go below 2, as long as the total energy difference between consecutive timesteps is greater than

Table 5.1: Single Agent Policy

Base	Communication	Tasking
$fuel <= 0 \rightarrow Loiter$ $fuel > 0 \rightarrow Advance$ $otherwise \rightarrow Loiter$	$fuel \leq 5 \rightarrow Advance$ $fuel < 4 \rightarrow Recharge_{NR}$ $otherwise \rightarrow Loiter$	$fuel \geq 5 \rightarrow Loiter$ $fuel < 5 \rightarrow Retreat$ $otherwise \rightarrow Loiter$

Table 5.2: Multi Agent Heuristic Policy. r represents all agents except 0th. The basic philosophy behind this heuristic is that the vehicle should not reach fuel levels ≤ 2 at any point in the execution.

A	Base	Communication	Tasking
0	$fuel <= 0 \rightarrow Loiter$ $fuel > 0 \rightarrow Advance$ $otherwise \rightarrow Loiter$	$fuel \leq 5 \rightarrow Loiter$ $fuel < 4 \rightarrow Rechg_{NR}$ $otherwise \rightarrow Loiter$	$Retreat$
r	$fuel <= 0 \rightarrow Loiter$ $fuel > 0 \rightarrow Advance$ $otherwise \rightarrow Loiter$	$fuel \leq 5 \rightarrow Advance$ $fuel < 4 \rightarrow Rechg_{NR}$ $otherwise \rightarrow Loiter$	$fuel \geq 5 \rightarrow Loiter$ $fuel < 5 \rightarrow Retreat$ $otherwise \rightarrow Loiter$

0, as shown in Figure 5-7.

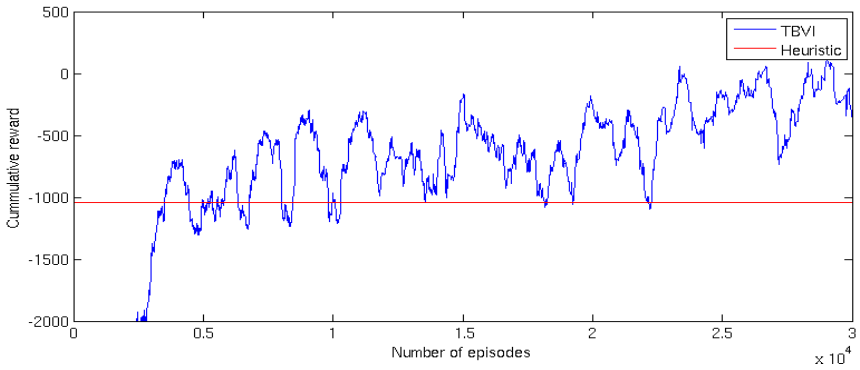


Figure 5-5: Red line represents the cumulative reward obtained using the heuristic policy defined in Table 5.2. The policy obtained through execution of TBVI exceeds the score obtained by the heuristic.

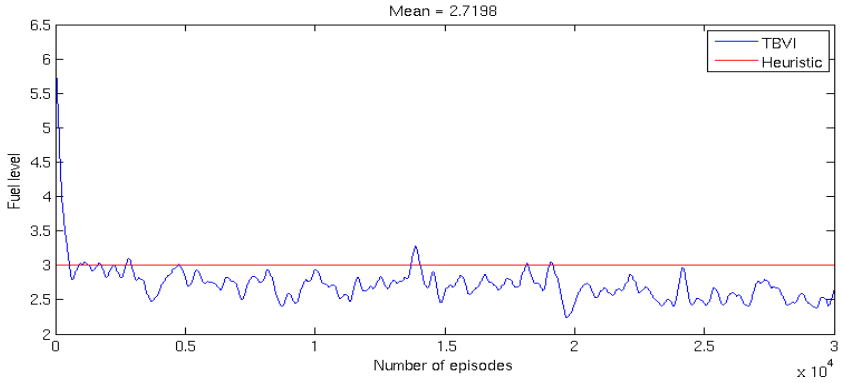


Figure 5-6: TBVI policy increases the cumulative reward by calling the quadrotor for recharge when its fuel level is well below 3. The heuristic policy, on the other hand, calls the quadrotor when the fuel level is 3. Calling it later means that that battery needs to be charged with smaller current to protect it. By sacrificing from the energy put into system, it increases the overall performance.

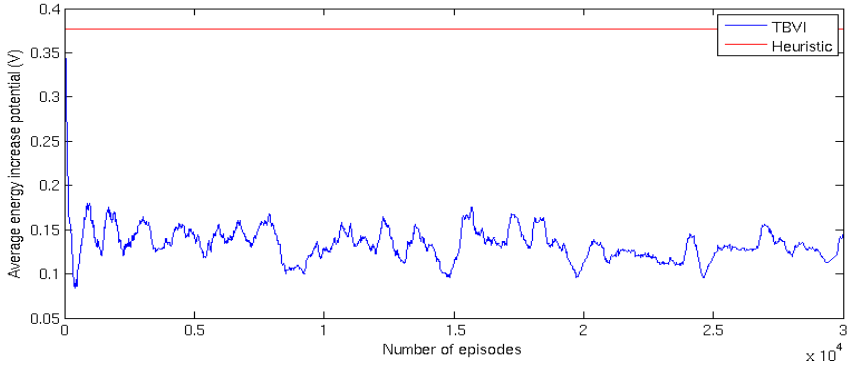


Figure 5-7: The TBVI policy increases the cumulative reward by calling the quadrotor later than fuel level drops below 3. By sacrificing from the overall voltage increase per timestep in the system, it increases the overall performance as it is seen in Figure 5-5.

Chapter 6

Conclusion and Future Work

This thesis consisted of two focus areas. The first was the design and implementation of a mobile battery charge platform. The motivation for such a platform was that, in many hardware experiments, the performance needs to be evaluated in the long run. Due to very short battery life, this is not possible without having a stack of batteries and charging them.

In order to extend the mission durations, this thesis introduced a platform that has a buffer of 7-batteries. The platform enabled very rapid battery swapping through its rail structure. Using off-the-shelf commercial chargers, the mission length is extended indefinitely, making it possible to test planning algorithms in the long run.

To prove relevancy of the designed platform, multiple hardware experiments have been designed and implemented. The first of these experiments was a simple take-off and land mission. This experiment was useful in that it isolated the platform from other aspects of missions, and concentrated just on the capability. Multiple runs of this experiment, one lasted for $5 - hours$ and one lasted for $3hours$, have been done, and data regarding to battery voltages on the UAV and the platform has been collected. The second mission was a car chase mission in which the quadrotor tried to learn the car's behavior so that it could guess where the car might be in the future. This mission was executed for about $4 - hours$. Third experiment was the persistent search and track mission. This was the only experiment that used 3 battery replacement platforms. In this experiment, over 120 swaps have been realized

in about $3.5 hours$. This experiment was particularly useful for robustness test. The latest of the missions was another car chase mission with 2 quadrotors that simply followed a ground vehicle. This experiment was conducted in Boeing facilities, and it lasted for about $75 minutes$

The battery management platform brought the necessity to to manage the use of this resource. In many missions, there are multiple UAVs and multiple battery management platforms, and proper scheduling of this resource is of crucial importance. The second part of the thesis proposes a modification to the well-studied persistent search and track mission scenario to more realistically model the battery swapping, and to increase the performance of the mission. Increasing the problem complexity also brought the necessity to implement faster MDP solvers. Using an approximate MDP solver, iFDD, the problem became tractable but still very slow. By introducing caching mechanism into iFDD, the solver is made orders of magnitude faster.

6.1 Future Work

Although the introduced battery management platform introduces persistency into system, there is still room for improvement. In its current state, the platform is capable of serving one vehicle indefinitely. However, if balanced charging were used, the charging rate could be increased more than $1C$, and it would have been possible to serve more than 1 vehicles indefinitely. Balanced charging is also useful for keeping the battery healthy for longer periods. In addition, the current design is relatively complex with its drum structure. It makes it hard to build additional devices and add additional components such as balancers. Simplifications in the design are possible. One idea would be to use multiple "empty bay - central landing area - full bay" structures one next to each other. This would eliminate rotational actuation in the drums, removing one more point of failure. It would also make adding balancers easier as it would not need additional thick wires that connect to the charger.

Currently, the platform assumes all the batteries show similar discharge performance. This may not be true in real world scenarios since some of the batteries

may be cycled more than others, or discharged more than their safe level. Assuming all batteries are the same may result in relatively poor performance in algorithms that take into account that information. It is possible to learn individual battery performances using [47, 48]. If this information is embedded into system during trajectories fed into TBVI, it would be possible to improve system performance. To identify individual batteries, each battery could be tagged with an RFID chip, and an RFID reader could be embedded into landing platform to read its ID. This way, each battery performance could be logged with timestamps and IDs, and analysis on that data could be performed.

Additional modifications to IFDD are also possible. Currently the cache grows indefinitely. This increases the memory consumption, and slows down the performance to some extent. Using smarter caching structures, like caches that only keeps most frequently used feature mappings, it would be possible to put an upper bound on the memory used by the cache. Moreover, current implementation is not optimal. Every time a new feature is discovered, multiple copies of the θ vector is created in order to add the new feature into its appropriate slot. This doesn't need to be the case, and copy operation could be removed completely by allocating larger feature vector than there is available.

Appendix A

Recharge Station Manual

A.1 Recharge Station Communication Protocol

The recharge station operation is controlled through messages sent using UDP protocol. The software currently supports two different messaging protocols, one described in [24–26] and another used by Boeing VSTL lab.

In summary, these two protocols can be summarized as follows:

- Raven Protocol

  Raven protocol is an ASCII protocol in the form of space delimited values. $<$ *Sender ID* $>$ $<$ *Destination ID* $>$ $<$ *Command ID* $>$ $<$ 8 *Data Items* $>$;

Field	Data Type	Description
Sender ID	Integer	Unique identifier of the sender. This is assigned by the Vicon Broadcaster
Destination ID	Integer	Unique identifier of the receiver. This is assigned by Vicon Broadcaster.
Command ID	Integer	Unique command identifier specifying the content of the message or command. Usually given different ranges for different command groups.
Data Items	Integer	Space delimited values describing the parameters of the specified commands.

- Boeing Protocol

  Boeing protocol is a binary protocol given in following table

MsgType	Destination	Orig	Command	Size	Payload	Count	CRC

Field	Data Type	Size	Description
Msg Type	Integer	4B	Needs to be 0
Destination ID	Integer	4B	Unique identifier of the receiver. This is assigned by Vicon Broadcaster.
Origin ID	Integer	4B	Unique identifier of the sender. This is assigned by the Vicon Broadcaster
Command ID	Integer	4B	Unique command identifier specifying the content of the message or command. Usually given different ranges for different command groups.
Size	Integer	4B	Size of the message in bytes. This doesn't include CRC
Payload	Float[8]	32B	Space delimited values describing the parameters of the specified commands.
Count	Integer	4B	Needs to be 8.
CRC	Integer	4B	Checksum

A.1.1 Operational Messages

Command Messages

These messages are the messages to control recharge station operations. They follow the same messaging format.

25015: Load/Unload bays

This message commands the recharge station to load or unload bays. This command makes use of battery voltages of each bay to determine if a bay is full or empty, so that it doesn't spin to the bay when commanded to be full and is already full.

Data Field	Name	Description
1	Bay Status 0	1: If needs to be full, 0: if needs to be empty.
2	Bay Status 1	1: If needs to be full, 0: if needs to be empty.
...		
x	Bay Status x	1: If needs to be full, 0: if needs to be empty.

25017: Load/Unload bays

This message commands the recharge station to start or to stop a particular bay charger.

Data Field	Name	Description
1	Operation	0: Stop. 1: Start
2	Bay Number	The charger number

25020: Start Swap

This message commands the recharge station to start swapping operation.

Data Field	Name	Description
1	Mode	Type of swapping to implement. AutoSwap = 1: Picks the swap source based on the battery levels. SwapUsingBays = 2: Swap source and destination is provided by user. RotateToBaysOnly = 3: Do not realize swap, but just rotate to bays given.
2	Source	Specifies the swapping source (i.e. the battery to push into quadrotor). Disregarded in Auto mode.
3	Destination	Specifies the swapping destination (i.e. the slot to have the spent battery).
4	Final Bay 1	The bay to rotate to after swapping is completed.
5	Final Bay 2	The bay to rotate to after swapping is completed.

Status Messages

These messages are the messages sent from recharge station to notify the mission manager and other vehicles about the state of battery bays or the state of an operation.

25016: Operation status This message informs the mission manager and other vehicles about the status of any operation.

Data Field	Name	Description
1	Status	1: Operation Started. 2: Operation Finished

25021: Bay battery voltages

This message informs the mission manager and other vehicles about the voltages of individual battery bays. The information is broadcasted by the station at $1Hz$.

Data Field	Name	Description
1	Voltage	The battery voltage of first bay
...		
x	Voltage	The battery voltage of x'th bay

A.2 Sample Recharge Station Configuration File

Recharge station configuration files are self explanatory. One recharge station controller can manage multiple recharge stations. Even though configuration file may contain more recharge station configuration than there are in Vicon stream, only the ones in Vicon stream will be initialized.

```
1 [
2   {
3     "communication": {
4       "bluetooth": {
5         //MAC address of the bluetooth device
6         "deviceAddress": "00:11:11:16:00:22",
7         //In case no address is provided, this is going to be used
              for lookup
```

```
8         "deviceName":"CH01",
9         //Specifies what dongle to use.
10        "dongle": 1
11      },
12      //Specifies how to communicate. Supported values are bluetooth
            and serialport
13      "method": "bluetooth"
14    },
15    //How many battery bays a drum has?
16    "drumCapacity": 4,
17    //Do we have charging capability - ie are chargers connected?
18    "hasChargingCapability": true,
19    //When do we say a bay is empty? This is useful because when a
         battery is pulled, the voltage decaysslowly.
20    "emptinessThreshold":7,
21    //Vicon name of the thing
22    "name": "CH01",
23    //Do we have center motor reverse?
24    "centerMotorReverse":true
25  },
26  {
27    "communication": {
28      "serialport": {
29        "uri": "/dev/ttyUSB0",
30        "baudrate":38400,
31      },
32      "method": "serialport"
33    },
34    "drumCapacity": 4,
35    "hasChargingCapability": true,
36    "emptinessThreshold":7,
37    "name": "CH02",
38    "centerMotorReverse":false
39  },
40 ]
```

Listing A.1: My Javascript Example

A.3 Recharge Station PCBs and Schematics

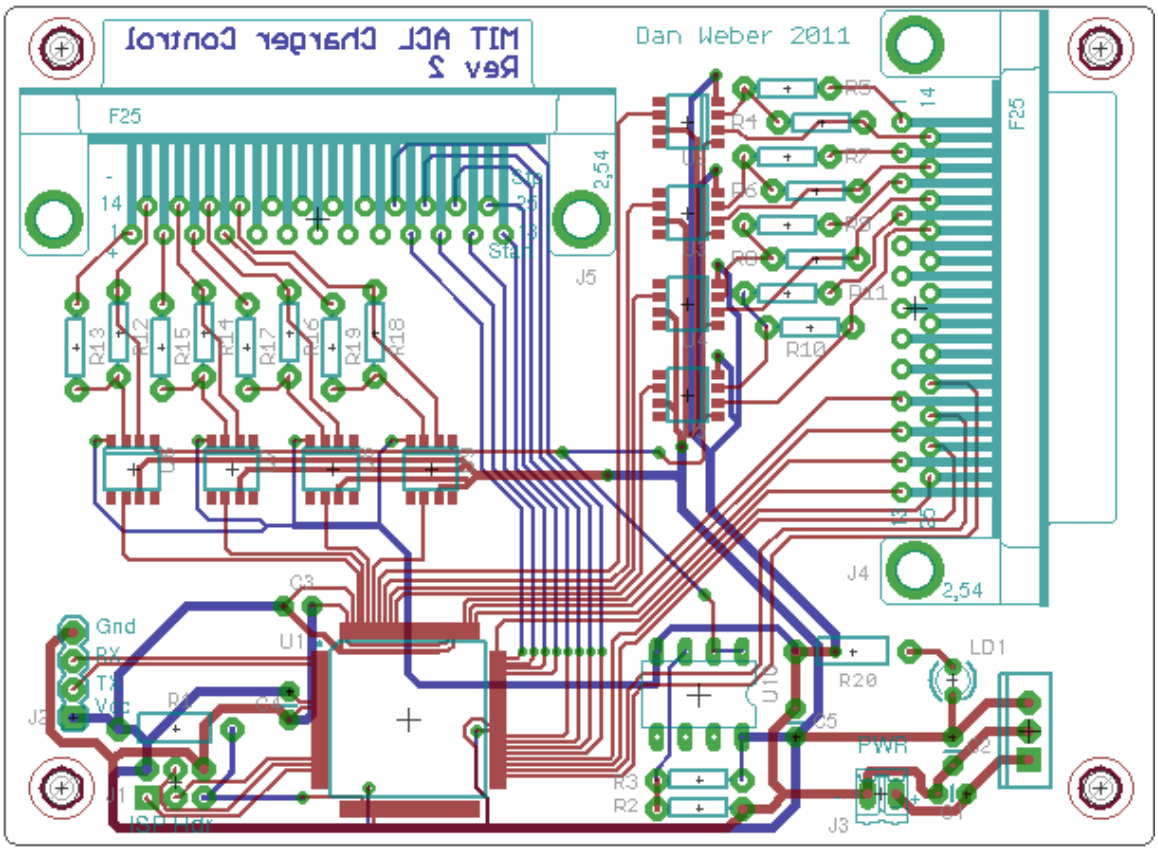


Figure A-1: Charger PCB

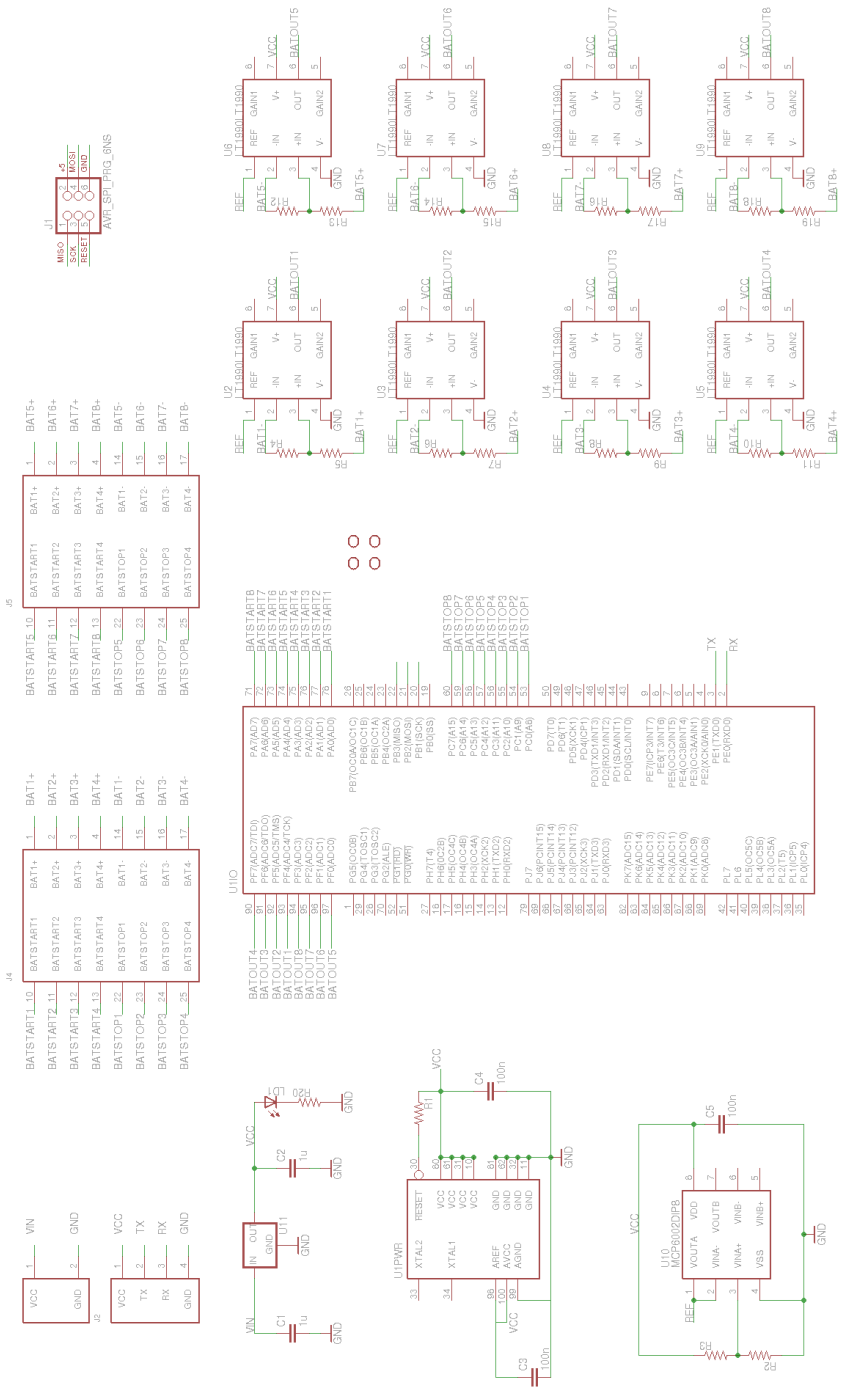
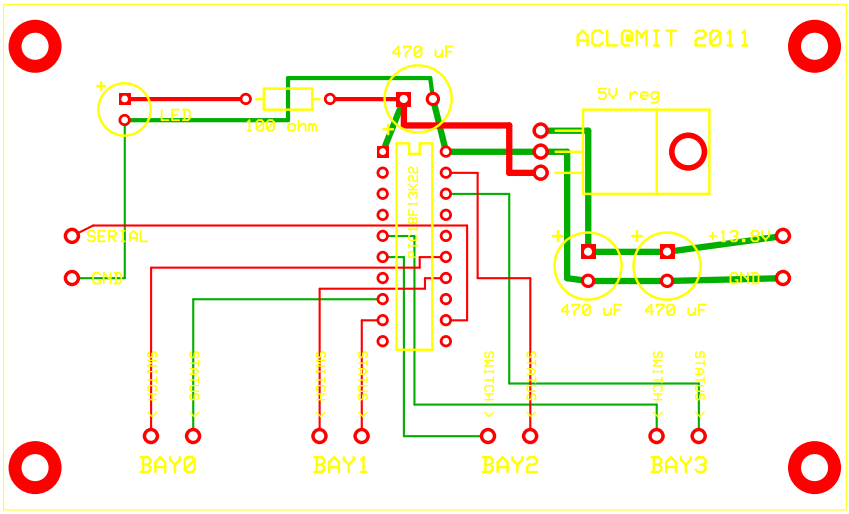


Figure A-2: Charger Schematic



f

Figure A-3: Drum PCB

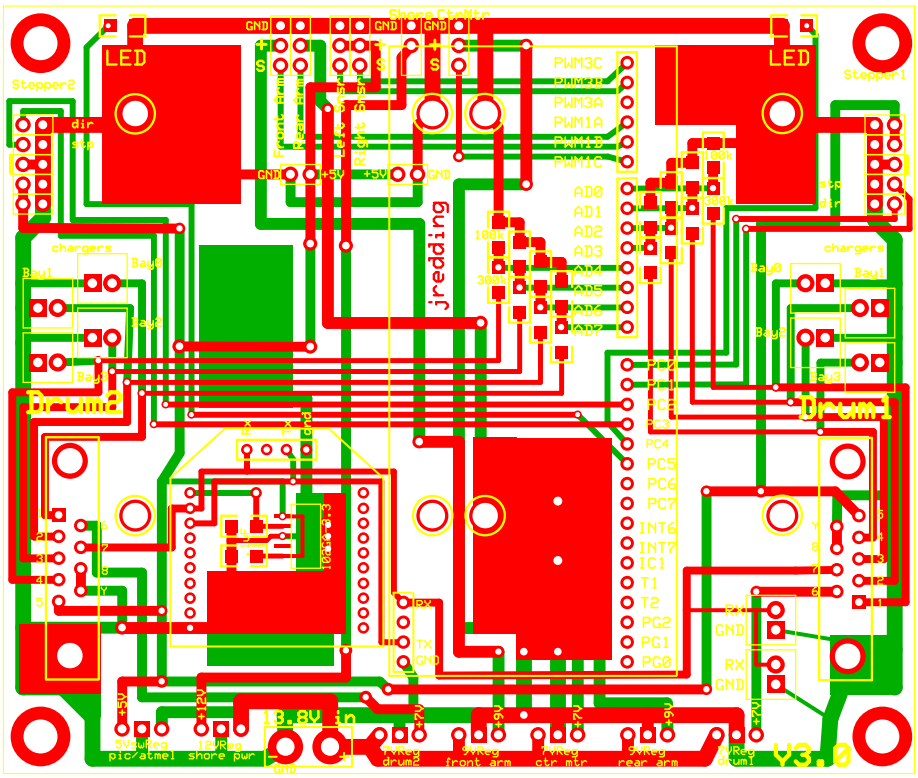


Figure A-4: Central PCB

Bibliography

[1] M. Valenti, D. Dale, J. How, and J. Vian, "Mission health management for 24/7 persistent surveillance operations," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, (Myrtle Beach, SC), August 2007.

[2] e. a. Vian, John L., "Autonomous vehicle rapid development testbed systems and methods," 10 2010.

[3] L. G. Weiss, "Autonomous robots in the fog of war," *IEEE Spectrum*, vol. 48, no. 8 (NA), p. 30, August, 2011.

[4] Y. Hada and S. Yuta, "A first-stage experiment of long term activity of autonomous mobile robot - result of repetitive base-docking over a week," in *Experimental Robotics VII*, ISER '00, (London, UK), pp. 229–238, Springer-Verlag, 2001.

[5] Y. Hada and S. Yuta, "A First-Stage Experiment of Long Term Activity of Autonomous Mobile Robot - Result of Respective Base-Docking Over a Week," *Lecture Notes in Control and Information Sciences: Experimental Robotics VII*, vol. 271, pp. 229–238, 2001.

[6] D. Austin, L. Fletcher, and A. Zelinsky, "Mobile Robotics in the Long Term - Exploring the Fourth Dimension," *Procedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[7] K. Kouzoubov and D. Austin, "Autonomous recharging for mobile robotics," in *Australian Conference on Robotics and Automation, Auckland*, pp. 27–29, 2002.

[8] M. C. Silverman, B. Jung, D. Nies, and G. S. Sukhatme, "Staying alive longer: Autonomous robot recharging put to the test," Tech. Rep. CRES-03-015, Center for Robotics and Embedded Systems (CRES), University of Southern California, 2003, 2003.

[9] V. Vladimerouy, A. Stubbs, J. Rubel, A. Fulford, J. Strick, and G. Dullerud, "A hovercraft testbed for decentralized and cooperative control," in *American Control Conference (ACC)*, (Boston, MA), pp. 5332–5337, July 2004.

[10] R. Cassinis, F. T. P. Bartolini, and R. Fedrigotti, "Docking and charging system for autonomous mobile robots," 2005. (http://www.ing.unibs.it/∼arl/docs/papers/05_008.pdf).

[11] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T. Chiueh, "MiNT-m: An Autonomous Mobile Wireless Experimentation Platform," *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, 2006.

[12] D. R. Dale, "Automated ground maintenance and health management for autonomous unmanned aerial vehicles," Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge MA, June 2007.

[13] M. J. Valenti, *Approximate Dynamic Programming with Applications in Multi-Agent Systems*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge MA, May 2007.

[14] K. Swieringa, C. Hanson, J. Richardson, J. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3335–3340, May 2010.

[15] A. Chernov, "Self-charging stations for flying vehicles," 2008. Semester Project.

[16] B. Bethke, J. P. How, and J. Vian, "Multi-UAV Persistent Surveillance With Communication Constraints and Health Management," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2009. (AIAA-2009-5654).

[17] e. a. David A. Cohen, "Autonomous robot auto-docking and energy management systems and methods," 02 2008.

[18] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *Spectrum, IEEE*, vol. 45, pp. 26 –34, july 2008.

[19] A. Stubbs, V. Vladimerou, A. Vaughn, and C. Dullerud, "Development of a vehicle network control testbed," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 4, pp. 3028 – 3033 vol.4, 2002.

[20] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, J. Strick, and G. Dullerud, "A hovercraft testbed for decentralized and cooperative control," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6, pp. 5332 –5337 vol.6, 30 2004-july 2 2004.

[21] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, pp. 396 –407, jun 1996.

[22] A. Hu, C. Liu, and H. L. Li, "A novel contactless battery charging system for soccer playing robot," in *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*, pp. 646 –650, dec. 2008.

[23] S. Mukhopadhyay, G. Gupta, and B. Lake, "Design of a contactless battery charger for micro-robots," in *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*, pp. 985 –990, may 2008.

[24] M. Valenti, B. Bethke, G. Fiore, J. P. How, and E. Feron, "Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, (Keystone, CO), August 2006 (AIAA-2006-6200).

[25] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, pp. 51–64, April 2008.

[26] J. P. How, C. Fraser, K. C. Kulling, L. F. Bertuccelli, O. Toupet, L. Brunet, A. Bachrach, and N. Roy, "Increasing autonomy of UAVs," *Robotics and Automation Magazine, IEEE*, vol. 16, pp. 43–51, June 2009.

[27] T. Nugent and J. Kare, "Laser Power for UAVs," 2008. http://lasermotive.com/wp-content/uploads/2010/04/Wireless-Power-for-UAVs-March2010.pdf.

[28] K. Suzuki, P. Kemper Filho, and J. Morrison, "Automatic battery replacement system for uavs: Analysis and design," *Journal of Intelligent & Robotic Systems*, pp. 1–24, 2011.

[29] F. P. Kemper, K. A. Suzuki, and J. R. Morrison, "Uav consumable replenishment: Design concepts for automated service stations," *J. Intell. Robotics Syst.*, vol. 61, pp. 369–397, January 2011.

[30] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, pp. 56 –65, sept. 2010.

[31] E. Altug, J. Ostrowski, and C. Taylor, "Quadrotor control using dual camera visual feedback," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, pp. 4294 – 4299 vol.3, sept. 2003.

[32] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii*, 2008.

[33] G. M. Hoffmann, H. Huang, S. L. Wasl, and E. C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *In Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2007.

[34] J. D. Redding, *Approximate Multi-Agent Planning in Dynamic and Uncertain Environments*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, February 2012.

[35] Mikrokopter , "Electronic speed controllers."

[36] http://sparkfun.com/products/3970.

[37] M. Cutler, "Design and Control of an Autonomous Variable-Pitch Quadrotor Helicopter," Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, August 2012.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[39] A. Geramifard, *Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2012.

[40] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[41] A. Geramifard, F. Doshi, J. Redding, N. Roy, and J. How, "Online discovery of feature dependencies," in *International Conference on Machine Learning (ICML)* (L. Getoor and T. Scheffer, eds.), pp. 881–888, ACM, June 2011.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[43] L. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, pp. 385–393, 1982. 10.1007/BF02579435.

[44] J. E. Smith and J. R. Goodman, "A study of instruction cache organizations and replacement policies," *SIGARCH Comput. Archit. News*, vol. 11, pp. 132–137, June 1983.

[45] J. E. Smith and J. R. Goodman, "A study of instruction cache organizations and replacement policies," in *Proceedings of the 10th annual international symposium on Computer architecture*, ISCA '83, (New York, NY, USA), pp. 132–137, ACM, 1983.

[46] P. McCullagh and J. Nelder, *Generalized Linear Models, Second Edition*. Monographs on Statistics and Applied Probability, Taylor & Francis, 1989.

[47] N. K. Ure, A. Geramifard, G. Chowdhary, and J. P. How, "Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery," in *European Conference on Machine Learning (ECML)*, 2012.

[48] N. K. Ure, G. Chowdhary, J. Redding, T. Toksoz, J. How, M. Vavrina, and J. Vian, "Experimental demonstration of efficient multi-agent learning and planning for persistent missions in uncertain environments," in *Conference on Guidance Navigation and Control*, (Minneapolis, MN), AIAA, August 2012.