Design and Control of Miniature Air-and-Ground Vehicles

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUN 21 2017
LIBRARIES
ARCHIVES

by

Minoru Brandon Araki

B.S., Yale University (2014)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.

Author ... Signature redacted .....
Department of Mechanical Engineering
May 12, 2017

Certified by... Signature redacted .....
Daniela Rus
Professor and Director of CSAIL
Thesis Supervisor

Certified by..... Signature redacted .....
Sangbae Kim
Assistant Professor
Thesis Supervisor

Accepted by ..... Signature redacted .....
Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses

Design and Control of Miniature Air-and-Ground Vehicles

by

Minoru Brandon Araki

Submitted to the Department of Mechanical Engineering
on May 12, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

The ability to both fly and drive is a superpower that few robots have. This thesis describes the design and control of two miniature air-and-ground vehicles, the "Flying Monkey" and the "Flying Car." The Flying Monkey was developed to demonstrate the viability and utility of miniature air-and-ground vehicles. The final design weighs 30g yet is capable of crawling, grasping, and flying. It features a novel crawling and grasping mechanism that consists of 66 linkages yet weighs only 5.1 grams. Although the crawler is capable of only forward and backward motion, we designed a controller that uses the yaw torque of the propellers to give the Flying Monkey two degrees of freedom on the ground. In experiments we demonstrated that the Flying Monkey is able to grasp small objects, fly over obstacles, and crawl through narrow pipes. The Flying Car was designed as a swarm vehicle to test multi-robot path planning. We therefore made the Flying Car as simple and robust as possible, built a small swarm of them, and tested them in a miniature town. We present two of the first algorithms for multi-robot path planning for air-and-ground vehicles, one based on priority planning and the other based on multi-commodity network flow. Thus, by designing and testing robots, controllers, and algorithms for miniature air-and-ground vehicles, this thesis hopes to serve as a starting point for future research in this promising area of study.

Thesis Supervisor: Daniela Rus
Title: Professor and Director of CSAIL

Thesis Supervisor: Sangbae Kim
Title: Assistant Professor

Acknowledgments

This thesis was made possible by many people besides myself. I would like to thank my research advisor, Daniela Rus, for taking me into her lab and giving me the opportunity to learn and do so many amazing things. I would also like to thank my many collaborators, particularly Jesung Koh, who helped with the mechanical design of the Flying Monkey; Daniel Aukes, who taught me how to use popupCAD and was a constant source of advice; Mike Tolley, who initially came up with the idea for the Flying Monkey; and Robert Wood, for allowing me to work in his lab and giving me a start in the world of research. I also need to thank Yash Mulgaonkar, who designed the Dragonfly quadrotor and who carried out the tests of the Flying Monkey; Luis Guerrero-Bonilla, who designed the controller for the Flying Monkey; Anurag Makineni, who helped write the code and run the system to control the Flying Monkey; and Vijay Kumar for supporting their work. Next I would like to thank my amazing UROP students Jack Strang, who wrote much of the code for the SIPP algorithm and the Flying Car simulator; Sarah Pohorecky, who was responsible for much of the Flying Car control, and Celine Qiu, who helped with the pure pursuit path following controller. I would like to thank Tobi Naegeli for his constant supply of quadcopter wisdom. I am also grateful to Robert Katzschmann and Cenk Baykal for their advice and help on the Flying Car project. Next, I would like to thank all of my labmates, past and present, for their help and friendship throughout my Master's studies. And last but far from least, I am grateful and thankful to my family, as well as to Kristi, for their constant love and support throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nearly all animals that can fly can also walk. Birds, bats, and insects all come equipped with legs and feet despite the energetic burden of carrying them. Even though aerial locomotion is faster, more agile, and, in the case of animals, more efficient than walking, the ability to maneuver on solid surfaces safely and efficiently is extremely valuable. It allows animals to perform tasks that require dexterity and precision, such as feeding their young, gathering food, and interacting with others. So why is it that so few of today's flying robots, particularly small ones such as quadrotors, have any form of ground locomotion? There are countless scenarios in which the ability to maneuver on the ground would greatly benefit a flying robot. For example, a quadrotor with wheels could drive through a narrow tunnel, orient itself on a platform to take a picture, or station itself to collect data. Meanwhile, it is also easy to imagine scenarios in which a ground robot would benefit from the ability to fly. For example, a search-and-rescue robot could fly over a pile of debris instead of needing to climb over it, or a flying car could whisk passengers over buildings and traffic to reach its destination quickly.

In this thesis, I explore the idea of multimodal locomotion by describing the design and control of two miniature air-and-ground vehicles. But first, why air-and-ground vehicles? And why miniature?

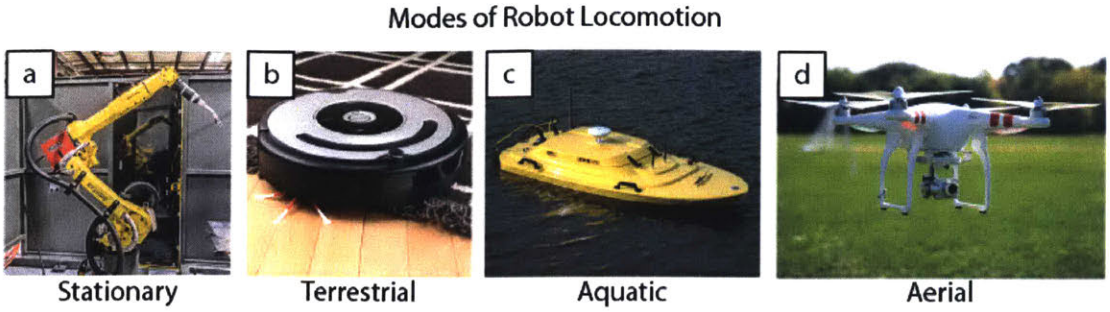


Figure 1-1: Robots have at least four general modes of locomotion. a) Stationary robots such as this Fanuc Arcmate 100IC/7L arm do not move. b) Terrestrial robots like this iRobot Roomba 560 are confined to the ground. c) Aquatic robots including this DiveLog Automated Survey Boat by Shark Marine Technologies stick to the water. d) Aerial robots, for example this DJI Phantom 2 Vision+ quadrotor, navigate through the air.

1.1 Modes of Locomotion

Robots can be divided into four general classes based on mobility, shown in Fig. 1-1. The most widely used are stationary robots – the robotic arms on factory floors, which are secured tightly to the ground. Second are terrestrial vehicles, such as autonomous cars and Roombas. Third are aquatic robots, such as autonomous submarines and boats. And fourth are aerial vehicles, such as quadrotors and other UAVs. Since stationary robots are not mobile and aquatic robots are usually highly specialized, I narrowed my focus to considering only ground and aerial locomotion for this thesis. My goal was to combine aerial and terrestrial locomotion in a way that could enhance the mobility of a robot.

In summary (see Table 1.1), ground robots are slow and highly constrained compared to aerial robots, but they also have a higher payload capacity and lower energy needs for a given payload. By contrast, aerial vehicles, particularly quadrotors, are fast and can easily navigate around most obstacles, but they have high energy needs and a low payload capacity.

In nature, flying is actually a more efficient form of locomotion than walking or crawling [1]. Yet, despite the energetic cost of carrying a set of legs to walk on, almost all animals that can fly also have some form of terrestrial locomotion. Amongst

	Pros	Cons
Aerial	• Fast • Agile • Unconstrained by ground obstacles	• High energy needs • Low payload capacity
Terrestrial	• Low energy needs • High payload capacity	• Slow • Constrained by ground obstacles
Aerial + Terrestrial	• Fast • Agile • Unconstrained by ground obstacles • Low energy needs on ground	• High energy needs in air • Low payload capacity

Table 1.1: Comparison of aerial, terrestrial, and aerial + terrestrial locomotion

birds, for example, only highly specialized birds such as hummingbirds and loons have difficulty walking. So why is it that almost all robots have been designed for either ground or aerial locomotion, but not both? One of the main reasons, which will be elaborated upon in Chapter 2, is that the weight of the ground mechanism would be too great for the aerial vehicle to carry or would reduce the robot's flight time to be impractically short. Another is that adding a second mode of locomotion increases the cost of the vehicle. However, I believe that these reasons are a poor excuse to not study robots with combined aerial and ground locomotion. First of all, battery energy density will inevitably improve in the future, easing the payload and energy constraints on aerial vehicles. Second of all, one can work around the payload and energy constraints by making the driving mechanism lighter or by automating the battery recharging process so that a swarm of flying-and-driving vehicles can stay in the air at all times. And third of all, combining aerial and ground locomotion in a single vehicle results in significant improvements in mobility.

By combining aerial and ground locomotion, as shown in Table. 1.1, it is possible to have all of the benefits of aerial locomotion plus the option to switch to low energy travel on the ground. This allows an air-and-ground vehicle to fly over ground

obstacles, drive through narrow passages, and prioritize between speed and energy efficiency while traveling. The only costs incurred by having air-and-ground capabilities is that the weight of the terrestrial locomotion mechanism slightly increases the energy needs and reduces the payload capacity of the robot. However, these limitations can be mitigated by making the terrestrial locomotion mechanism as light as possible, as discussed in Chapter 3.

I expect that in the future, robots with both aerial and ground locomotion will be used in a variety of tasks. For example, a surveillance robot could fly into an air shaft, crawl through the air ducts, and park itself near a vent to gather information. Once its mission is complete, it could then crawl back out and transmit its data. Or, a search-and-rescue robot could enter an earthquake-damaged building and fly over piles of rubble and crawl through narrow openings in order to map out the building for rescuers. As materials science and manufacturing techniques enable batteries with higher energy density and actuators with higher power density, I expect that it will one day be taken for granted that aerial robots have some form of ground locomotion.

1.2 Miniature Robots

In this thesis, I specifically study miniature air-and-ground robots. Why miniature? Some of the reasons are listed in Table 1.2. Both the pros and cons of small robots make them an interesting subject of study. First of all, small robots are easier to work with since they are safer, cheaper, and stronger in relation to their size. Second of all, the constraints on small robots – a shorter battery life and lower payload capacity, along with their small size – elevate the challenge of constructing a lightweight ground mechanism into a research problem. For example, attaching landing and taxiing gear to the General Atomics MQ-9 Reaper drone, which has a wingspan of 84 feet, is not a research challenge. I therefore focused on designing ground mechanisms for small quadrotors, which have extreme payload and energy constraints, since that is where the open research challenges are.

	Pros	Cons
Small	• Stronger • Safer • Cheaper	• Shorter battery life • Low payload capacity
Big	• High payload capacity • Longer battery life	• Weaker • Safety hazard • More expensive

Table 1.2: Comparison of large and small robots

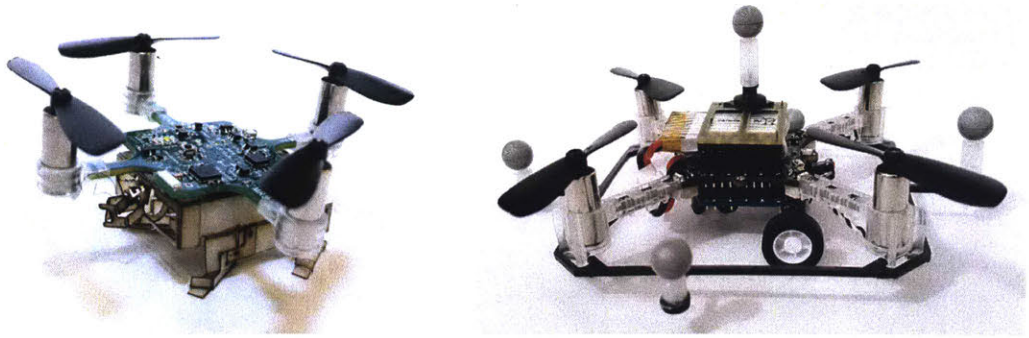
1.3 Path Planning for Air-and-Ground Vehicles

Given a robot that can both fly and drive, how does one choose paths for it? Aerial locomotion is fast but energy-expensive, whereas ground locomotion is slow but efficient. The value placed on speed versus efficiency forms a natural basis for choosing when and where to fly or drive. As will be discussed in Chapter 2, air-and-ground robots have been so little studied that there is almost no literature on path planning for robots with multimodal locomotion. Therefore, another major thrust of this thesis is proposing and testing algorithms that solve the path planning problem for multiple air-and-ground vehicles.

1.4 Contributions

I make three main contributions to the existing literature in this thesis:

1. A novel crawling and grasping mechanism fabricated using smart composite microstructures manufacturing.
2. A hardware platform for a miniature wheel-based robot that can fly and a system architecture for a swarm of such robots.
3. Two modified algorithms for multi-robot path planning with multimodal locomotion.



(a) The Flying Monkey [see Section 3.1] (b) The Flying Car [see Section 3.2]

Figure 1-2: The two air-and-ground vehicles explored in this thesis

1.5 Organization of Thesis

In this thesis, I describe the design and control of two miniature air-and-ground vehicles, the Flying Monkey (Fig. 1-2a) and the Flying Car (Fig. 1-2b). In Chapter 2, I discuss the background of air-and-ground vehicles, foldable robots, and multi-robot path planning. Chapter 3 describes the design of the Flying Monkey and the Flying Car. Chapter 4 explains the control of each vehicle. Chapter 5 covers multi-robot path planning for the Flying Car. Chapter 6 describes the experiments done with the Flying Monkey and the Flying Car that validate the effort put into their design, control, and path planning.

Chapter 2

Related Work

2.1 Miniature Air-and-Ground Vehicles

The field of miniature air-and-ground vehicles is small but slowly growing. Researchers have come up with diverse strategies for coping with the low payload of flying vehicles, but of the six designs surveyed here, four of them save weight by integrating the aerial and terrestrial locomotion mechanisms. For example, the DALER robot, shown in Fig. 2-1c, moves on the ground by rotating its wings. This clever use of the wings eliminates the need for a separate ground mechanism. Meanwhile, the Quadroller (Fig. 2-1a), HyTAQ (Fig. 2-1e), and Flymobile (Fig. 2-1f) are quadrotors on passive wheels that employ various clever mechanisms that allow their propellers to propel them when they are on the ground.

The other major category of air-and-ground vehicle treats the aerial and terrestrial locomotion mechanisms as independent subsystems. A unit of the Distributed Flight Array (Fig. 2-1b) uses a singe large propeller to fly (they must be attached to other units to fly stably) and has three independent omniwheels for ground locomotion. MALV (Fig. 2-1d) uses flexible wings and a propeller to fly and crawls on the ground using 'whegs,' which are leg-like wheels. The Distributed Flight Array can overcome the payload limitation of aerial vehicles because a single unit has a proportionally huge propeller compared to the size of the omniwheels - therefore, when at least three units are joined, the vehicle is not miniature but quite large, meaning that the

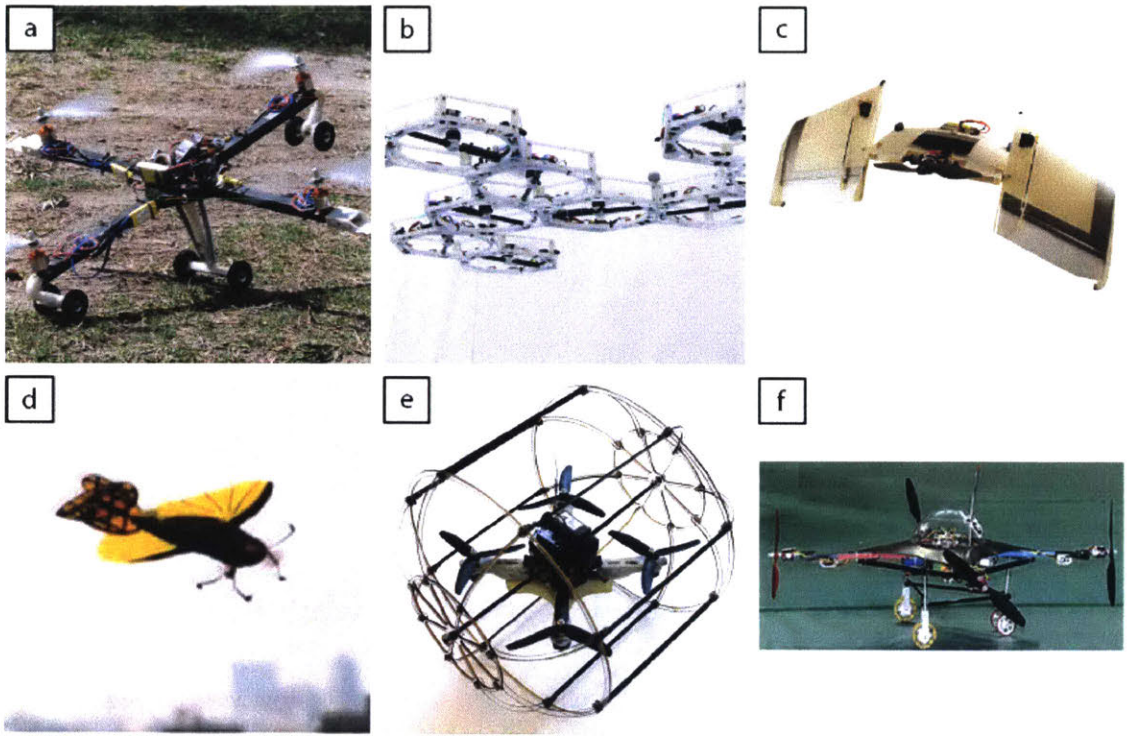


Figure 2-1: Researchers have come up with many ingenious designs for miniature air-and-ground vehicles. a) The Quadroller tilts to move forward on land [2]. b) Each unit of the Distributed Flight Array consists of a propeller and three omniwheels [3]. c) DALER can rotate its wings to move on the ground [4]. d) MALV has flexible wings and a propeller for aerial locomotion and 'whegs' for terrestrial locomotion [5]. e) HyTAQ moves on the ground by spinning in a cage [6]. f) The Flymobile tilts its rotors by 90° to propel itself on the ground [7]. All photos used with the permission of the authors.

omniwheels are proportionally very small and light. MALV gets around the weight limit by using clever materials and manufacturing techniques - the flexible wings and whegs are very lightweight.

The Flying Monkey's crawling mechanism has a single motor for forward and backward motion. It therefore must rely on its propellers for steering while on the ground. Moreover, smart composite microstructures techniques were used to supply the Flying Monkey with an extremely lightweight and robust crawling and grasping mechanism. The Flying Monkey is therefore a hybrid of the strategies discussed above. By contrast, the driving mechanism of the Flying Car is innovative in its simplicity - it is a three-wheeled differentially steered mechanism that is independent from the

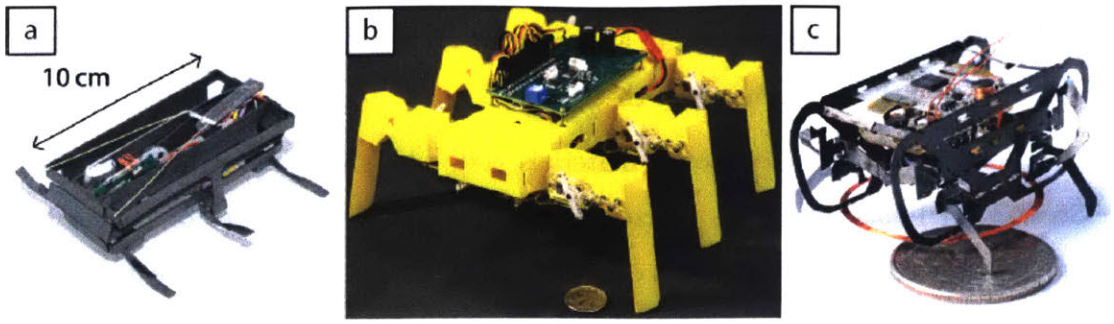


Figure 2-2: (a) The DASH robot served as the inspiration for the Flying Monkey [8]. (b) The MIT FireAnt robot is a foldable 12-dof hexapod [9]. (c) Harvard's HAMR weighs only 2.1g [10].

flying mechanism. The simplicity of the Flying Car's design allowed us to make and test many of them in a multi-robot system, as will be discussion in Chapter 5. The design of the Flying Monkey and the Flying Car will be discussed in Chapter 3.

2.2 Foldable Robots

In recent years, the idea of creating 3D mechanisms by first manufacturing a 2D design and then folding it has been gaining traction. The field of foldable robots was inspired by the art of origami, in which intricate 3D shapes can be folded from a sheet of paper. Since 2D manufacturing typically involves planar sheets of material and a laser cutter, production is fast and cheap. Researchers have devised many manufacturing techniques to develop a vast array of designs. One method of making foldable robots is to cut and score a single sheet of plastic, which can then be folded into a final shape. [9] (see Fig. 2-2b) describes the design of a 6-legged, 12-DOF ant-like robot. [11] made a crawling robot and a gripper; the two 2D designs were then combined by connecting them with a bridge, enabling the manufacture of a crawling robot with a gripper. [12] describes a fold pattern that enables the creation of a wide range of cylindrical robots that can move using a worm-like peristaltic motion. [13] presents a Python package that automatically generates cut patterns for quadrotor frames that can be cut and folded from a sheet of plastic. Although the technique of folding a robot or structure from a single sheet of plastic has been successfully applied

to a wide variety of robots, the use of a single type of material (typically a thin sheet of plastic) limits the performance of these robots.

The concept of smart composite materials [14], which will be further discussed in Section 3.1.1, addresses the shortcomings of using a single sheet of plastic by introducing the concept of laminates composed of both rigid and flexible layers. Cuts in the rigid and flexible layers allow the creation of flexures and rigid links. Since the rigid layer (typically fiber glass or carbon fiber) can be selected to be far stiffer than the flexible layer (typically thin plastic), far higher performance can be gained from SCM robots. Examples of SCM robots include the Robobee, a 60mg robot that can fly [15], and the Harvard Ambulatory Microrobot (HAMR), a 2.1g robot quadrupedal robot that can travel up to 4 body lengths/sec [10] (see Fig. 2-2c). Other examples of SCM robots include a water strider robot that takes of advantage of the surface tension of its legs on water to jump as efficiently as an actual water strider [16] and a myriapod robot that weighs 2.2 grams yet has 20 legs and can run at over 1 body length/sec [17]. The DASH robot (see Fig. 2-2a), which served as the primary inspiration for the Flying Monkey (as will be discussed in Section 3.1.2) was also constructed using laminates. Due to the high performance, intricate design, and light weight made possible by SCM techniques, the Flying Monkey was also designed and manufactured as an SCM robot.

Several design programs have been made to either automatically generate foldable robots or make the design of foldable robots easier. [18] presents a system for automatically generating foldable designs, wiring instructions, and code based off of high-level user specifications. Its system has a library of parametrically defined robot parts such as legs, wheels, and bodies, which can be modified and stitched together to satisfy the user's requirements. The Interactive Robogami system presented in [19] also has a library of expertly designed parts that can be composed into a single robot. However, it provides an interactive 3D environment in which users can design their own robot themselves by dragging and dropping and modifying components. The software used in this thesis is called popupCAD [20]. Unlike the previously mentioned systems, popupCAD does not take user specifications as input. Rather it

allows low-level operations on planar geometries (such as union and difference functions) that allow the creation of complicated laminate structures. popupCAD greatly speeds up the design process by providing a placement operation that enables subdesigns (such as a hinge) to be composed with another design (such as a hinged robot). popupCAD also provides operations such as support generation that enables the generation of manufacturable vector files for use with a mill or laser cutter. Prior to the development of popupCAD, all of these operations involved in SCM laminate design had to be done by hand in a CAD environment such as Solidworks. popupCAD was invaluable to the design of the Flying Monkey (as described in Section 3.1.4).

2.3 Path Planning

The problem of path planning for air-and-ground vehicles has received little attention – I have been able to find only one paper that addresses the subject, 3D Path Planning for Flying/Crawling Robots, by Rao and Arkin, published in 1989 [21]. Rao and Arkin proposed using A* to find a lowest-cost path on a discrete 3D grid on which flying and crawling costs have been computed. This thesis picks up where Rao and Arkin left off by extending the path planning problem to the multi-robot domain.

However, the multi-robot path planning problem (MPP) for robots with a single mode of locomotion has been the subject of extensive research. MPP is a challenging problem because the configuration space grows exponentially with the number of robots [22]. It can be made tractable through decentralized solutions such as [23, 24, 25] that delegate to each robot the responsibility for avoiding other robots. Centralized algorithms exist that can find feasible collision-free paths for multiple robots in $O(n^3)$ time or faster [26]. Priority planning is a suboptimal but fast centralized solution to MPP in which priorities are assigned to robots and paths are then found for each robot in order of its priority [27, 28]. A number of tricks exist for making priority planning better. For example, [29] suggests limiting the planning horizon, rotating priorities amongst robots, and using a true distance heuristic to get better paths from priority planning. [28] proposes the idea of well-formed infrastructures in

order to generate graphs and scenarios with guaranteed solutions. Meanwhile, approaches such as [30, 31] provide optimal solutions but become intractable when the graph size or number of robots goes past a certain limit. However, these problems can be made tractable by allowing for suboptimal solutions by for example splitting the problem into multiple smaller problems in the time domain. I contributed to this literature by extending MPP to include robots with multiple modes of locomotion. Chapter 5 describes two algorithms, one a fast priority planner and the other an optimal ILP, to show that the MPP can be extended to include the modified objective function of flying cars.

Chapter 3

Design

3.1 Design of the Flying Monkey

The Flying Monkey is a 30g robot with the ability to fly, crawl, and grasp. My principle contribution to the design of the Flying Monkey was its gripper and crawling base, shown in Fig. 3-1. The two mechanisms and their actuators had to weigh less than the quadrotor's 10g payload capacity. I created the Flying Monkey's base using the techniques of smart composite microstructure fabrication, since they are uniquely suited for designing and manufacturing lightweight yet complex mechanisms.

3.1.1 Overview of Smart Composite Microstructures

Smart composite microstructures (SCM) technology uses laminates consisting of stiff and flexible layers to create complex mechanisms [14]. Diagrams of a typical SCM laminate are shown in Fig. 3-2. Flexures can be easily made by cutting gaps in the rigid layer. The areas where the rigid layers are present define links, while gaps in the rigid layers that expose the flexible layer define hinges. These laminates enable the design of 2D patterns which, once manufactured, can be folded into functional 3D mechanisms.

Recent advances in techniques for analyzing laminate geometries, determining manufacturability, and automating the creation of laminate device manufacturing files

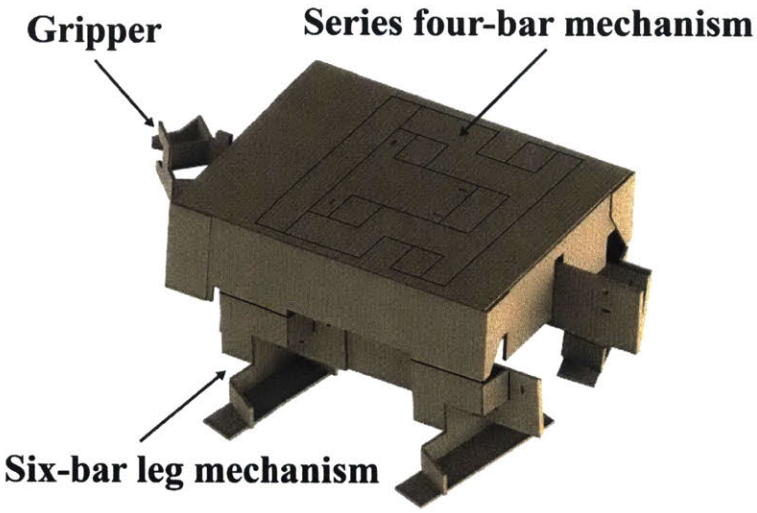
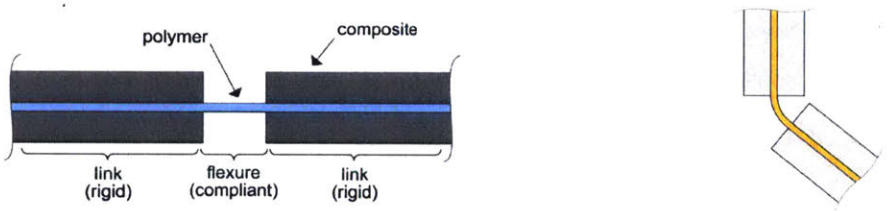


Figure 3-1: A folded model of the Flying Monkey's base.



(a) (b)

Figure 3-2: (a) Links and flexures are created by cutting gaps in the rigid layer. (b) The gap in the rigid layer allows the flexible layer to bend. Diagrams from [14].

have made designing and manufacturing SCM devices into a fast and simple process [32]. Furthermore, functional components such as hinges and four-bar linkages can be saved and reused in an object-oriented fashion using a purpose-built software tool called popupCAD [20], a design suite that stores and operates upon layered sets of planar geometries. popupCAD also provides operations for converting the laminate design into the vector files needed for manufacturing.

3.1.2 Kinematic Design

The kinematic design of the crawling mechanism was based off of UC Berkeley's DASH robot [8]. DASH's kinematic design calls for a 1-DOF 6-bar leg mechanism

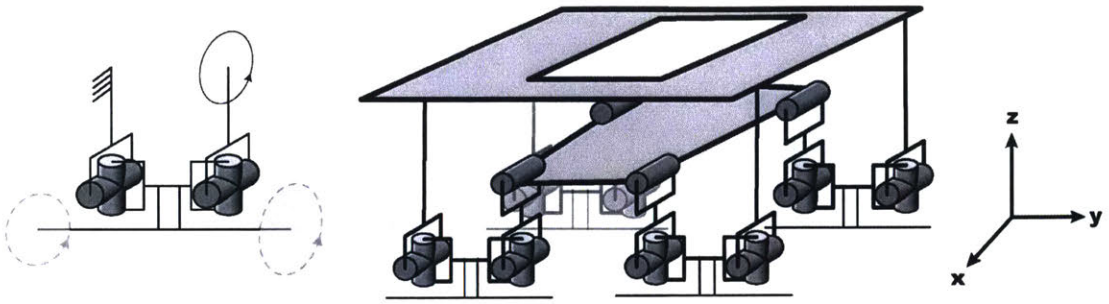


Figure 3-3: A kinematic diagram of the Flying Monkey's crawling mechanism. The diagram on the left shows how the actuator input results in output motion of the feet. The diagram on the right shows the overall structure of the crawler.

that actuates six legs. I modified this design to be the one shown in Fig. 3-3. In this design, every 'foot' is attached to a 4-bar linkage, or 'hip', that allows fore-aft motion. Each foot consists of an inner foot and an outer foot. Each hip is connected to a central 1-DOF 6-bar mechanism that is constrained to be able to move in a circular pattern in the x-z plane. A motor with a crank is attached to the 6-bar mechanism by a shaft. As the crank turns, it causes the 6-bar mechanism to trace out a circle. As the 6-bar mechanism moves up and down, it causes the hips to pivot inwards and outwards, respectively, causing the outer feet to move down and up and the inner feet to move up and down. As the 6-bar mechanism moves forward and backward, it causes the hips to rotate as well, causing the outer feet to move backward and forward and the inner feet to move forward and backward. In summary, the rotary motion of the crank and the 6-bar mechanism causes the outer and inner feet to trace out alternating circular strokes. When the four outer feet are touching the ground, the four inner feet are not; and when the four inner feet are touching the ground, the outer feet are in the air. This allows the leg mechanism to move forward in a stable configuration.

3.1.3 Laminate Pattern Design

The final laminate design of the Flying Monkey is shown in Fig. 3-4. It consists of 66 rigid elements. Each bounded area represents a linkage. Dotted lines represent hinges

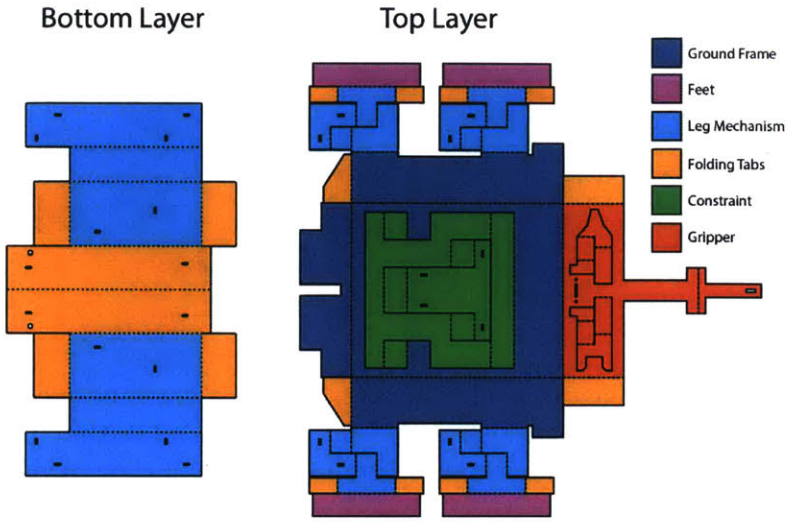
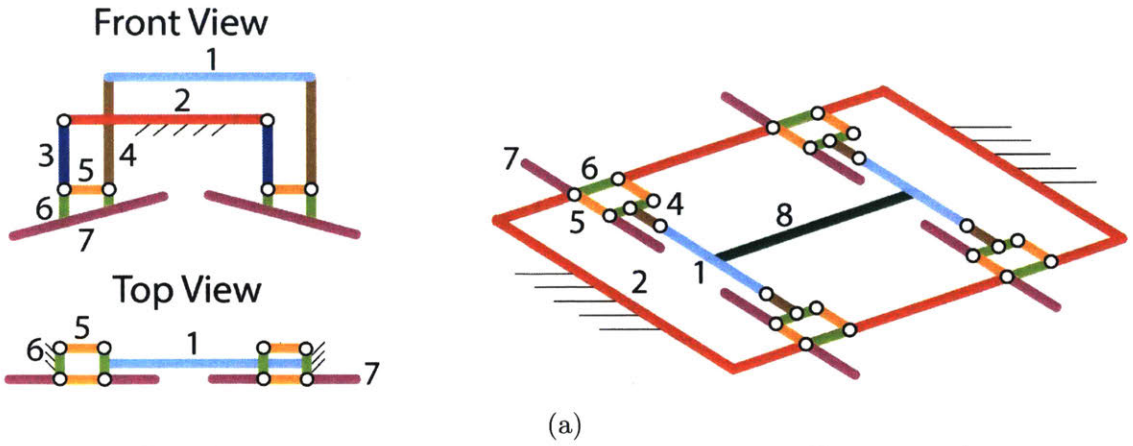


Figure 3-4: The laminate design of the Flying Monkey. The links are color-coded to correspond to functional regions.

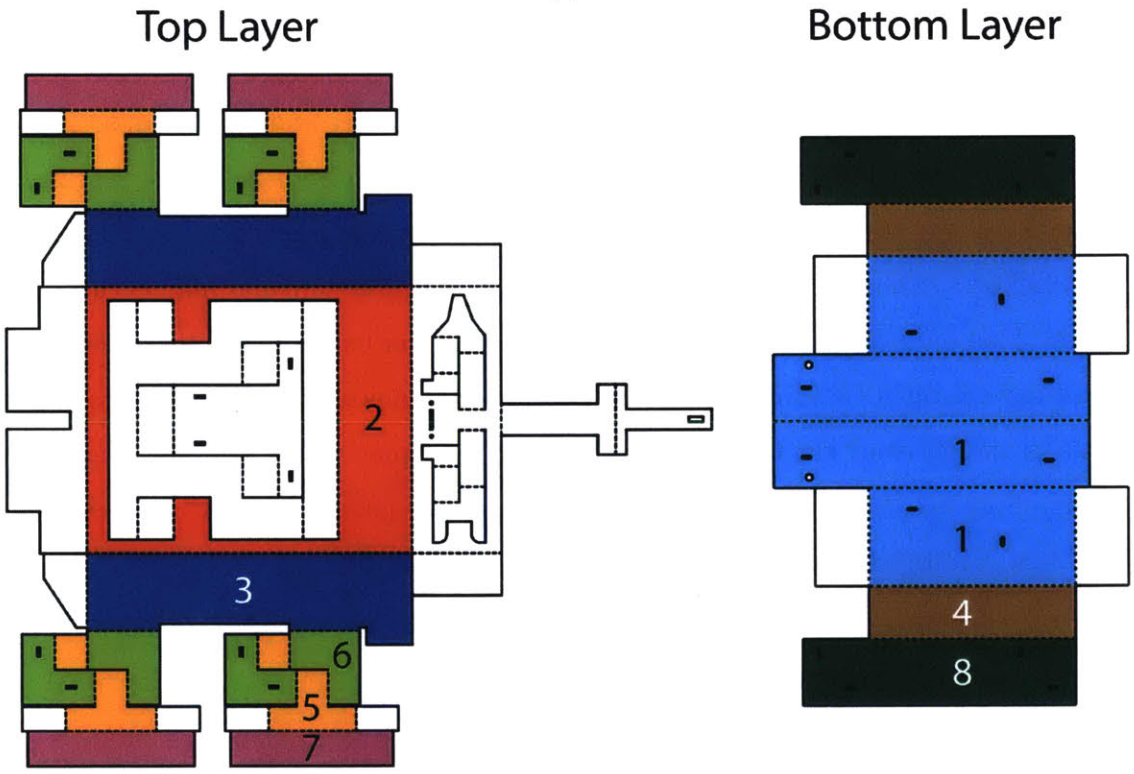
and solid lines represent cuts. The design consists of two sublaminates, a top layer and a bottom layer, which themselves are glued together to form a single composite laminate. Folding tabs, shown in orange, constitute an important part of the design beyond the kinematic plan, because by being folded and glued into place, they hold the folded design together and help to rigidize it.

The correspondence of each link in the abstract kinematic design to a link in the laminate pattern design is shown in Fig. 3-5. For example, the Flying Monkey has four 'hips', each of which is a 4-bar linkage. In Fig. 3-5a, the links of the hips are colored as orange and green. The corresponding links in Fig. 3-5b are also colored orange and green. Some of the links have also been numbered to help clarify the relationship between the individual links. For example, one of the hips in Fig. 3-5a has been labeled with the numbers 5 and 6. The corresponding links in Fig. 3-5 have also been labeled 5 and 6.

Two other major components of the laminate design are shown in Fig. 3-6. On the left of each subfigure is the abstract design and on the right is the laminate design. As in Fig. 3-5, the linkages are color-coded to show the correspondence between the abstract and laminate designs. In addition, linkage lengths are labeled. Fig. 3-6a shows the series 4-bar mechanism. Since the crawling mechanism is one DOF, the 6-



(a)

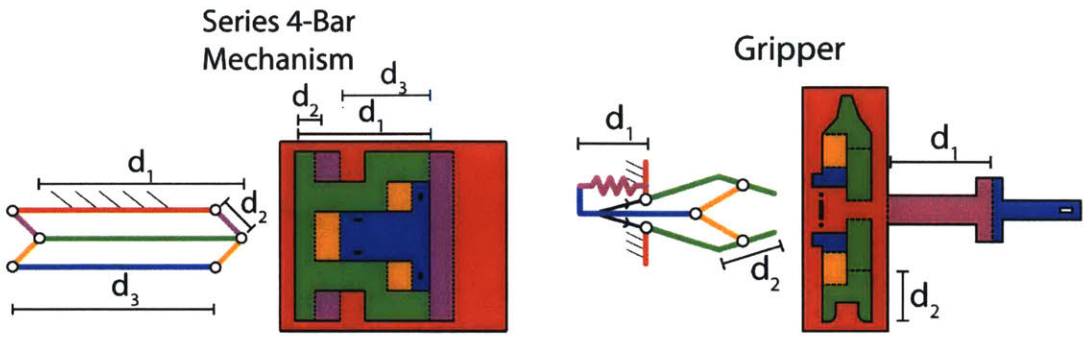


(b)

Figure 3-5: This figure shows the relationship between the abstract kinematic design (a) and the laminate pattern design (b). Linkages are numbered and color-coded to relate the kinematic linkages to the corresponding linkages of the laminate.

bar leg mechanism must be constrained to a single DOF. The series 4-bar mechanism serves this purpose by constraining the leg mechanism to a circular motion in the x-z plane.

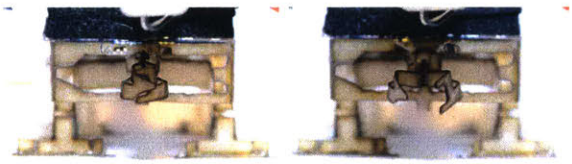
The gripping mechanism is shown in Fig. 3-6b. The gripper consists of two 4-



(a) (b)

Figure 3-6: (a) The series four bar constraint mechanism. (b) The gripping mechanism.

bar mechanisms with extensions that can be pulled together and pushed apart. It is unique among laminate pop-up designs in that it uses a built-in passive spring, colored in purple, to pull the gripper open. This is achieved by bending the blue and purple linkages so that the blue linkage slots into the small rectangular hole in the red linkage. This built-in spring pulls the gripper mechanism back so that the gripper's default state is open. A thin shape-memory alloy coil (shown as two black lines with arrows in the abstract kinematic diagram) pulls the gripper closed for short periods of time. Fig. 3-7 shows a closeup of the gripper in its open (Fig. 3-7b) and closed (Fig. 3-7a) positions. On the Flying Monkey, the onboard micro-controller controls the SMA actuator through one of the digital outputs and a high power MOSFET.



(a) Gripper closed (b) Gripper open

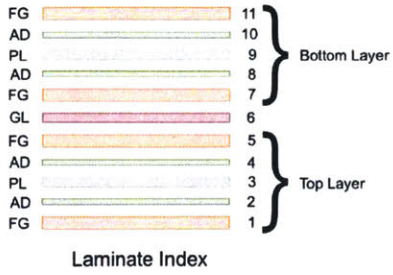
Figure 3-7: Images of the gripper open and closed

3.1.4 Fabrication

As discussed in the previous section, the laminate is composed of two five-layer sub-laminates bound together by an additional adhesive layer. The order of the eleven

Abbreviation	Material	Description
FG	Fiberglass	Flame-Retardant Garolite (G-10/FR4), 0.005" Thickness Precision Film
AD	Adhesive	GBC Octiva Hot Mount Adhesive
PL	Plastic	Grafix Clear 0.001" Dura-Lar Film Roll
GL	Hot Glue	

(a)



(b)

Figure 3-8: (a) The materials used to create the laminate design. (b) The ordering of the layers.

layers is shown in Fig. 3-8b, and the materials used are listed in Fig. 3-8a. The fiberglass serves as a rigid element that keeps the linkages stiff. The plastic layer servers as the flexible element in every hinge. The adhesive layers, meanwhile, bind the rigid and flexible layers together.

The design process is outlined in Fig. 3-9. First, the 2D laminate pattern is sketched in Solidworks. Although the pattern could be made in any vector graphics program, one benefit of using Solidworks is that the design can be tested by making the linkages into an assembly, adding appropriate constraints between them, and folding them into the final design. Next, the Solidworks sketch is exported to popupCAD. I have created custom zero-radius hinge designs in popupCAD which are added to the design. popupCAD then outputs vector files for the laser cutter for all of the layers of the laminate (see Figs. A-1 through A-12). A laser cutter and laminator are then used to cut and laminate the layers of material (as described in the next paragraph), and the finished laminate is hand folded into the final shape.

Fabrication of pop-up laminate devices - in other words, the process of turning flat sheets of multiple materials into a popup device - is described in [33]. The general process is illustrated in Fig. 3-10. I will describe my particular process here:

1. Preparation: A laser cutter is used to cut the layers of the laminate.
2. Layer Addition: The layers are aligned and stacked on top of each other.
3. Lamination: The layers are then passed through a laminator.
4. Release Cut: A laser cutter is used again to cut the final two laminates and

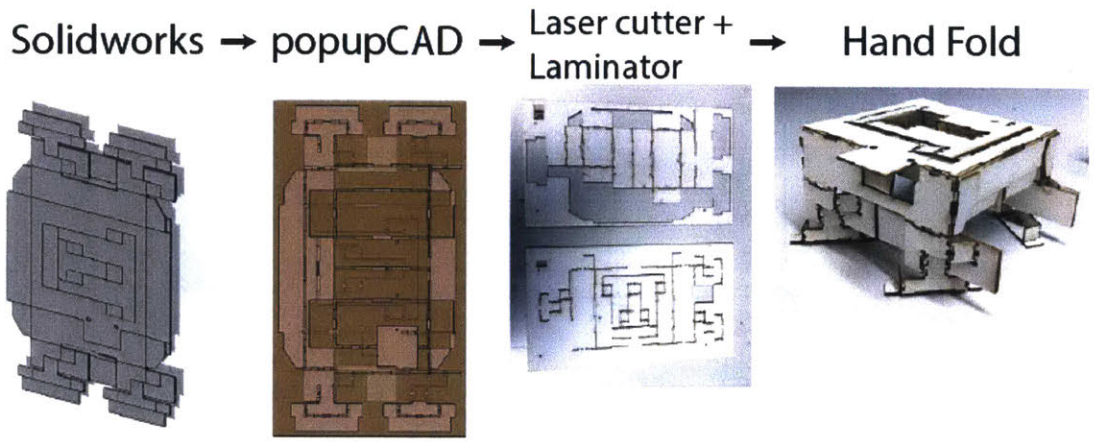


Figure 3-9: The design process.

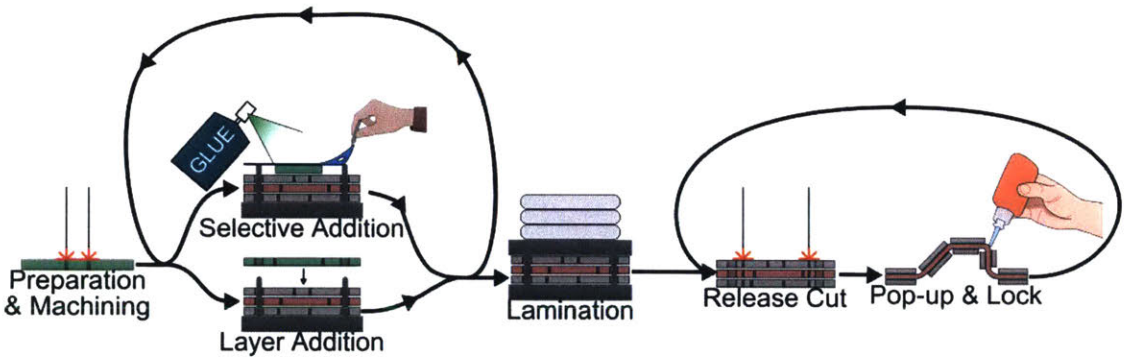


Figure 3-10: The pop-up fabrication process, as described in [33].

release them

5. Pop-up & Lock: The two finished laminates are glued together to form the final composite laminate

3.1.5 Dragonfly Quadrotor

The Dragonfly quadrotor used as the flying chassis for the Flying Monkey was designed by Yash Mulgaonkar of the University of Pennsylvania [35, 34]. It weighs 24g and is constructed from a 0.047" thick double layer fiber-glass PCB. It is fast and very agile, capable of reaching speeds of up to 6m/s and coming to a full stop within a 4m × 4m flight space.

The autopilot was designed from the ground-up in order to make it as efficient

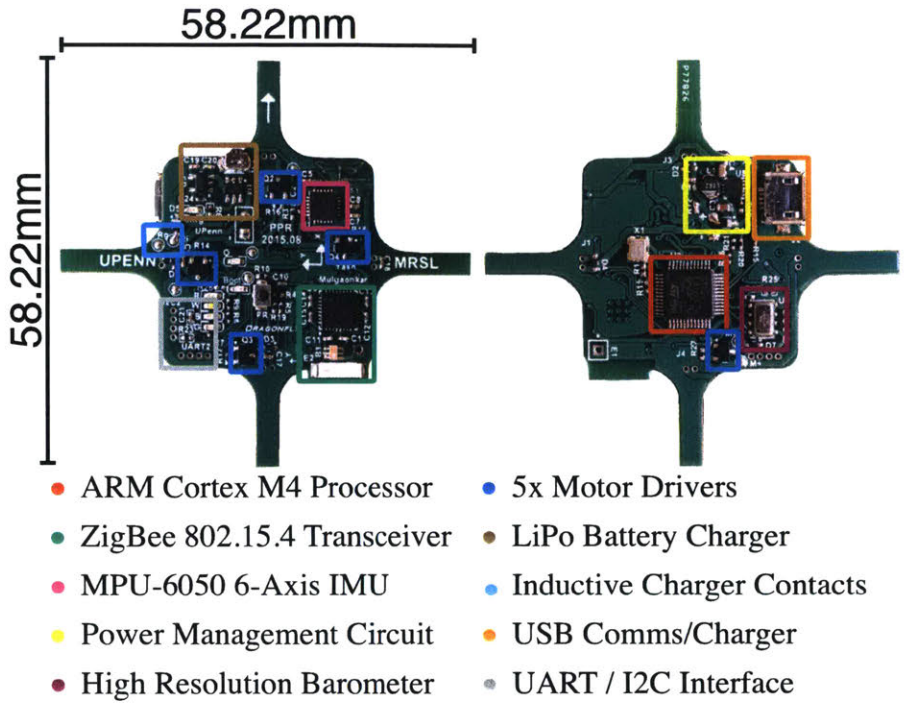


Figure 3-11: Components of the Dragonfly quadrotor, as described in [34].

and lightweight as possible. Almost all commercially available autopilots, such as the PX4 Pixhawk [36], though feature-rich, are too bulky for a micro quadrotor, weighing close to 36g, with a footprint averaging about $40cm^2$. In contrast, the Dragonfly's autopilot, shown in Fig. 3-11 spans only $3cm^2$ and weighs 4.8g. The Dragonfly is equipped with an ARM Cortex M4 STM32F373 microprocessor, which interfaces with Atmel's AT86RF212 900MHz 802.15.4 wireless transceiver chip. An InvenSense MPU-6050 6-axis MEMS gyroscope & accelerometer and a Measurement Specialties MS5611 high precision barometer allow for accurate attitude and altitude measurement, while a 3.3v Buck/Boost switching regulator powers all the subsystems while maintaining a consistent logic level throughout the circuit. Five 4A DC brushed motor drivers power the motors and an integrated Lithium Polymer ($LiPo$) battery charging circuit allows for in-system charging of the on-board battery. A micro USB port and two multipurpose I2C and UART ports allow for interfacing with a wide range of external sensors.

This 0.047" thick, double layered autopilot also serves as the main structural

component of the Dragonfly, eliminating the need for an additional load bearing frame. 3D printed snap-on motor mounts are used to attach the motors to the autopilot. Finally, a single cell 3.7V, 240mAh Li-Po battery powers the Dragonfly, giving it a six minute flight time.

3.1.6 Mass Distribution

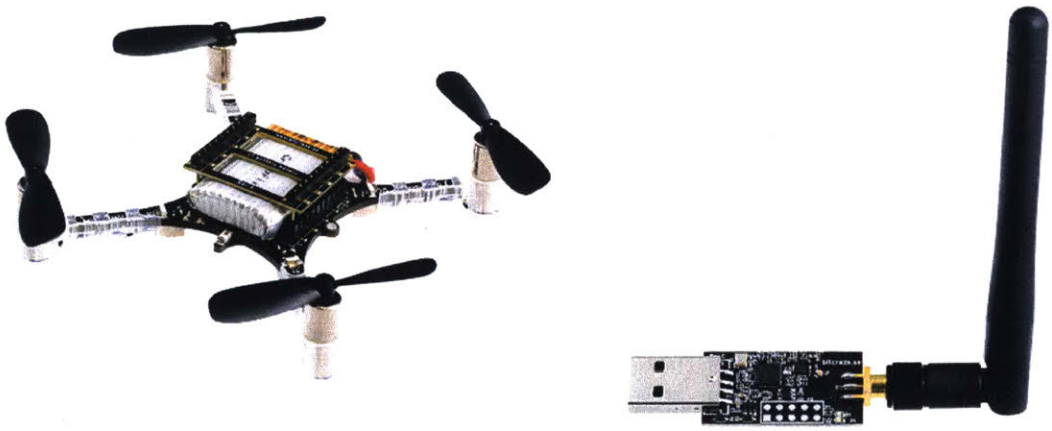
Table 3.1 shows the distribution of mass in the Flying Monkey. Its total mass is 30g. Given that the Dragonfly's maximum payload capacity is 10g, the crawler, which weighs only 5.1g, comes in well under the limit. Another 0.9g are used to up to attach the crawling base to the Dragonfly, meaning that the Flying Monkey could theoretically carry a payload of up to 4g.

Mass Table		
Item	Mass (g)	Mass Percent
Crawler	5.1 g	17%
Battery	8.1 g	27%
Dragonfly	15.9 g	53%
Misc.	0.9 g	3%
Total	30.0 g	100%

Table 3.1: Masses of the elements of the Flying Monkey

3.2 Design of the Flying Car

The Flying Car was designed to be the successor to the Flying Monkey and was intended to be used in a swarm. Therefore the goal of the design of the Flying Car was to make as small, reliable, robust, and easy-to-manufacture an air-and-ground vehicle as possible. As described in the following sections, this was accomplished by replacing the custom-made Dragonfly quadrotor with the commercially available Crazyflie 2.0 and by replacing the intricate crawling mechanism of the Flying Monkey with an extremely simple and robust two-motor differentially steered mechanism.



(a) The Crazyflie 2.0 (b) The Crazyradio PA Dongle

Figure 3-12: The Crazyflie 2.0 and Crazyradio PA were two important components of the Flying Car made by `bitcraze.io`.

3.2.1 Crazyflie Quadrotor

The Crazyflie 2.0 is a commercially available, highly versatile 27g quadrotor (see Fig. 3-12a). It has a 240 mAh Li-Po battery that provides 5 minutes of flight. Given that it has a payload capacity of 15g (56% of its body weight) and is fully open-source, it makes an ideal platform for research involving micro aerial vehicles. Moreover, it can communicate with a computer via a 2.4 GHz radio dongle called the Crazyradio PA, shown in Fig. 3-12b.

3.2.2 Wheel Base

The Crazyflie 2.0 is designed to accommodate expansion boards called 'decks' that can be used to add new functions to the Crazyflie. Decks interface with the Crazyflie via certain pins as shown in Fig. 3-13. I designed a wheel deck for the Crazyflie that adds a motor driver and a pair of motors to the bottom of the Crazyflie, as in Fig. 3-14b. I designed the deck to be as lightweight and robust as possible. It consists of a 1/32" thick PCB, shown in Fig. 3-14a with a TI DRV8835 dual motor driver mounted on top, and a 6mm carbon fiber tube glued with epoxy to the PCB that serves as the fixture for two 1.2g 136:1 micro planetary gear motors. The wheel mechanism can be seen in Fig. 3-15. The customized firmware that runs the wheel deck can be found at

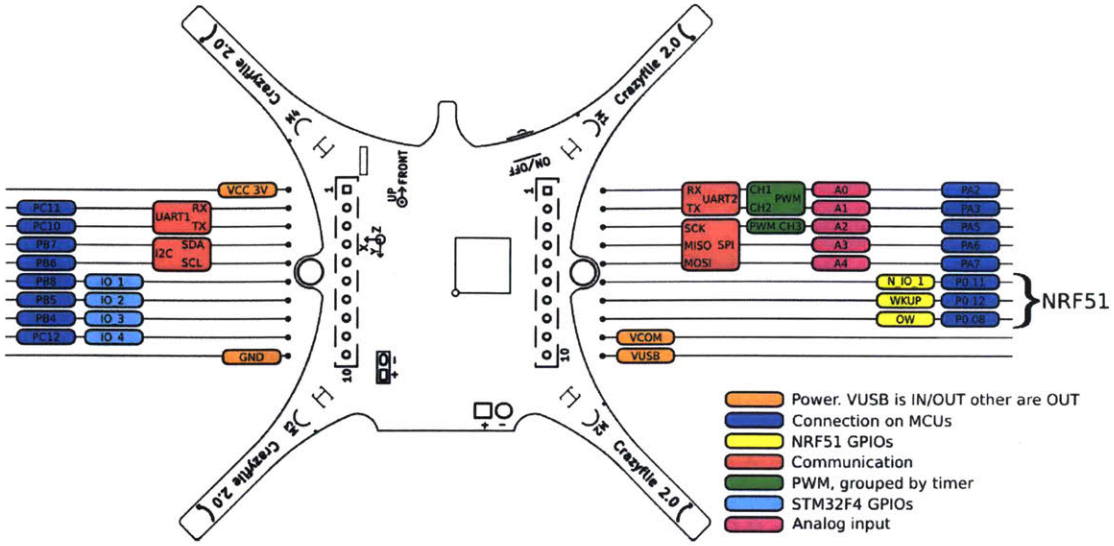
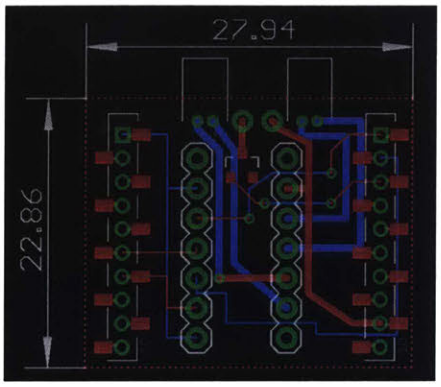
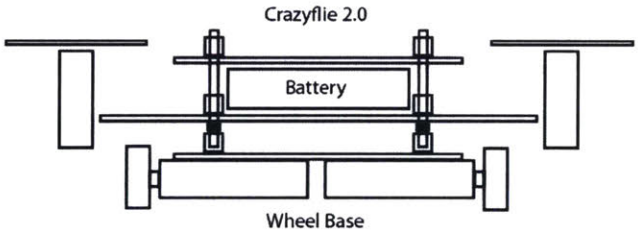


Figure 3-13: Pinout for Crazyflie 2.0 decks [37].



(a) Wheel deck PCB, units in mm



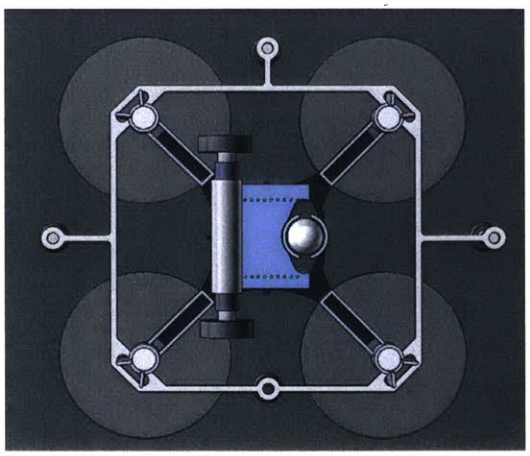
(b) The deck configuration of the Crazyflie 2.0

Figure 3-14

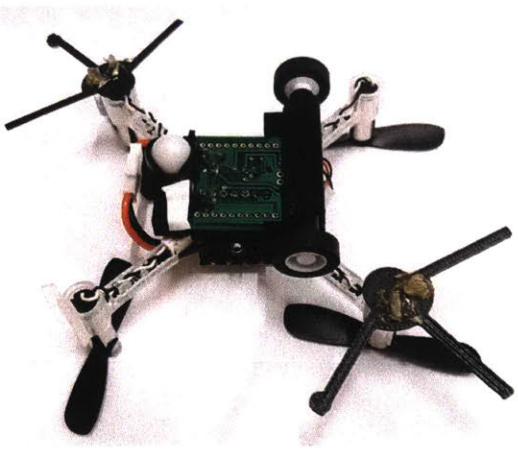
`https://github.com/braraki/crazyflie-firmware`.

3.2.3 Optical Tracking

A Vicon motion capture system was used to track the Flying Cars. Thin, lightweight frames made from Delrin based off of the design in [38] were made to hold the optical markers. Since a total of seven flying cars were meant to be controlled at once, seven unique optical marker configurations were made so that the Vicon system could distinguish between them.



(a) A model of the underside of the Flying Car



(b) The underside of the Flying Car

Figure 3-15: Two views of the driving mechanism of the Flying Car

3.2.4 Mass Distribution

The mass distribution of the Flying Car is shown in Table 3.2. The driving mechanism, at 8.3g, weighs more than the Flying Monkey's 5g crawling base. However, it is still well under the Crazyflie's 15g payload capacity.

Mass Table		
Item	Mass (g)	Mass Percent
Wheel Base	8.3 g	20.2%
Motion Capture Markers	4.2 g	10.2%
Crazyflie	28.5 g	69.5%
Total	41.0 g	100%

Table 3.2: Masses of the elements of the Flying Car

Chapter 4

Control

4.1 Control of the Flying Monkey

4.1.1 Ground Locomotion

The crawling mechanism of the Flying Monkey has only 1 DOF – forward and backward motion along the x-axis of the Flying Monkey's body frame (see Fig. 4-1). We gave the Flying Monkey a second DOF on the ground by using the quadrotor's propellers to control the yaw angle. We use a Dubins car model to study the behavior of the crawler with the quadrotor attached to it while crawling:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\phi(t)) \\ v(t)\sin(\phi(t)) \\ u(t) \end{bmatrix} \tag{4.1}$$

where $x(t)$ and $y(t)$ are the cartesian position of the robot in the plane, $\phi(t)$ is the yaw angle, and $v(t)$ and $u(t)$ are the control inputs for the linear velocity and yaw velocity respectively. Let us define $e_\phi = \phi - \phi_d$, where ϕ_d is the desired yaw angle, and assume that $|e_{\phi_{max}}| \leq \pi$. The control law for the yaw angle is selected as follows

$$u = -k_\phi \sin(e_\phi) + \dot{\phi}_d \tag{4.2}$$

where k_ϕ is a positive constant. For the linear velocity control law we use a controller

similar to [39]. Let $\mathbf{x}$ be the position vector in the plane and $\mathbf{x}_d$ the desired position vector. Defining $\mathbf{e}_x = \mathbf{x} - \mathbf{x}_d$, the control law for the linear velocity is selected as follows:

$$v = \left[-k_x\left(\mathbf{e}_x\right) + \dot{\mathbf{x}}_d\right]^T \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix} \tag{4.3}$$

where k_x is a positive constant. Substituting eqns.(4.2) and 4.3 into eqn.(4.1), it can be shown that

$$\dot{\mathbf{x}} = -k_x\left(\mathbf{e}_x\right) + \dot{\mathbf{x}}_d + \|-k_x\left(\mathbf{e}_x\right) + \dot{\mathbf{x}}_d\| \left|\sin\left(e_\phi\right)\right| \tag{4.4}$$

$$\dot{\phi} = -k_\phi \sin\left(e_\phi\right) + \dot{\phi}_d \tag{4.5}$$

Substituting eqn.(4.2) into eqn. (4.1) and rearranging terms, we find that

$$\dot{e}_\phi = k_\phi \sin\left(e_\phi\right) = 0 \tag{4.6}$$

Within $|e_\phi| \leq \pi$, the yaw angle has only one stable equilibrium point at $|\phi - \phi_d| = 0$ so that e_ϕ converges asymptotically to 0 in this region. Consider now the Lyapunov function candidate

$$V = \frac{1}{2}\mathbf{e}_x^T\mathbf{e}_x + \frac{1}{2}e_\phi^2 \tag{4.7}$$

It can be shown that its time derivative is negative definite as long as

$$k_x k_\phi > \frac{\|\dot{\mathbf{x}}_d\|_{max}^2}{4\left(1 - \left|\sin\left(e_{\phi_{max}}\right)\right|\right)} \tag{4.8}$$

where $\|\dot{\mathbf{x}}_d\|_max$ is the maximum value of the norm of $\dot{\mathbf{x}}_d$.

While this last constraint on the product of the gains k_x and k_ϕ might seem discouraging, it is important to notice that since e_ϕ converges asymptotically to 0

independent of the position error $\mathbf{e}_x$, there is no need to use high gains if we allow some time for the robot to get to the right orientation.

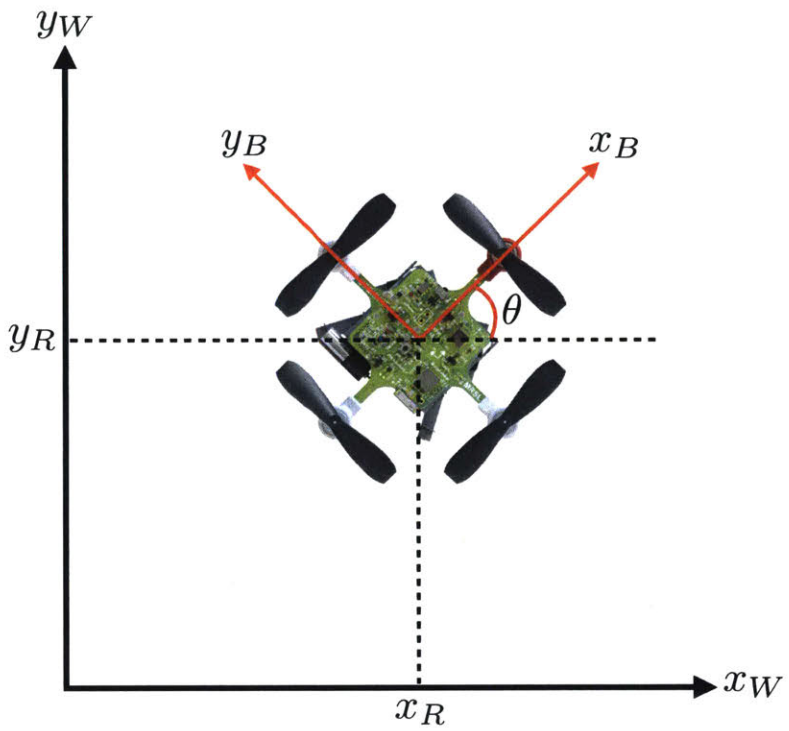


Figure 4-1: Flying Monkey coordinate system.

4.1.2 Aerial Locomotion

The controller described in [35] and [40] was used to control the aerial trajectory of the Flying Monkey (see Fig. 4-2). In short, an EKF was used to estimate the state of the Flying Monkey. Meanwhile, a PD controller was used to bring the quadrotor to a given setpoint.

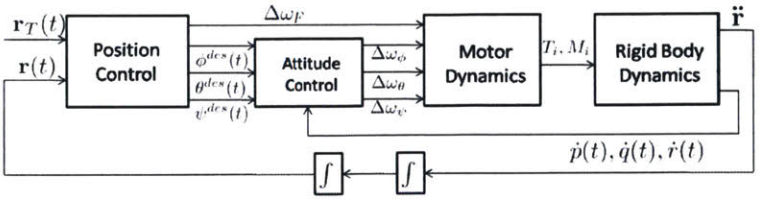


Figure 4-2: Position and attitude control of the Dragonfly quadrotor, from [40].

4.2 Control of the Flying Car

4.2.1 Ground Locomotion

On the ground, we use a pure pursuit path follower with PID wheel control for a differential steering mechanism. Let us define the state of the Flying Car on the ground as

$$\mathbf{x} = [x, y, \theta, v_x, v_y, t] \in \mathbb{R}^6 \tag{4.9}$$

A trajectory π is a zero-indexed array of goals. Every goal g contains states g_x, g_y, and g_t corresponding to the goal's x, y, and time values, respectively. For ground control, let us assume that the robot has already reached i goals. Then we consider both the goal g^1, which is defined as being $\pi[i]$, and the 'next goal' g^2, which is defined as being $\pi[i+1]$. g^1 is used to calculate speed, while g^2 is used to calculate θ. Goal speed g_{speed} is therefore calculated as

$$g_{speed} = \frac{\sqrt{(g_x^1 - x)^2 + (g_y^1 - y)^2}}{g_t^1 - t} \tag{4.10}$$

In other words, we want the Flying Car to maintain a speed that allows it to reach g^1 by the goal time.

The speed error e_{speed} is

$$speed = \sqrt{v_x^2 + v_y^2} \tag{4.11}$$

$$e_{speed} = g_{speed} - speed \tag{4.12}$$

The x and y errors are defined as

$$e_x = g_x^2 - x \tag{4.13}$$

$$e_y = g_y^2 - y \tag{4.14}$$

Next, we calculate the desired heading θ_d between the robot and g^2 as

$$\theta_d = \tan^{-1} \frac{e_y}{e_x} \tag{4.15}$$

and the angular error e_θ as

$$e_\theta = \theta_d - \theta \tag{4.16}$$

Now, given e_{speed} and e_θ, we can calculate the PWM signals to send to the wheels. We first define a feedforward command pwm_f that is a rough estimate of the PWM value necessary to achieve g_{speed}:

$$pwm_f = \frac{(g_{speed} + c_{b1})}{c_{b2}} \tag{4.17}$$

where c_{b1} and c_{b2} are experimentally determined constants.

Next we calculate a 'speed offset' offset_{speed}

$$\text{offset}_{speed} = k_{speed} e_{speed} \tag{4.18}$$

and a 'theta offset'

$$\text{offset}_\theta = k_\theta e_\theta \tag{4.19}$$

where gains k_{speed} and k_{theta} were experimentally determined. Next we define a PWM value

$$pwm = \max\left(0, \min\left(pwm_f + \text{offset}_{speed}\right), 255\right)) \tag{4.20}$$

Since the Flying Car uses a differential steering mechanism, we take this baseline PWM value and add/subtract a theta offset from it in order to generate PWM signals for the left and right wheels.

$$pwm_{right} = \max\left(0, \min\left(pwm + \text{offset}_\theta\right), 255\right) \tag{4.21}$$

$$pwm_{left} = \max\left(0, \min\left(pwm - \text{offset}_\theta\right), 255\right) \tag{4.22}$$

When x and y are within a certain tolerance of g_x and g_y, g^1 and g^2 are set to be $\pi[i+1]$ and $\pi[i+2]$.

4.2.2 Aerial Locomotion

We used a Kalman Filter to estimate the state of the Crazyflie, and we used the `crazyflie_ros` package to control it [41].[1] The `crazyflie_ros` package uses a simple PID controller to guide the Crazyflie to setpoints and to perform trajectory following.

[1] available at `https://github.com/whoenig/crazyflie_ros`

Chapter 5

Planning

There has been very little work on path planning for air-and-ground vehicles. Therefore, no path planner was made for the Flying Monkey because that went beyond the scope of the contribution. The Flying Car project was started with the desire to address the challenge of path planning for multiple air-and-ground vehicles.

5.1 Problem Statement

Assumptions We assume that we have a swarm of uniform robots that can drive and fly. They have a limited energy budget and therefore energy consumption is a major factor in the path planning.

Environment We model the environment by a 3D graph that consists of a 2D road system and an aerial region, either of which may contain disconnected regions as long as the overall graph is connected. The map can accommodate static obstacles, elevated "helipads", and "no-fly zones".

Problem Given n robots with unique start and goal positions, find collision-free paths for each robot.

Solution We developed and implemented two algorithms:

1. Priority planning with Safe Interval Path Planning: a relatively fast algorithm that gives guaranteed solutions under certain conditions; however, it is not optimal.

2. Multi-commodity network flow ILP: an NP-hard problem that can provide optimal solutions in a reasonable amount of time when the number of robots and graph vertices is not too high.

5.2 Problem Formulation

Let $\mathcal{W}$ denote a workspace with $\mathcal{W} \subset \mathbb{R}^3$. This space is approximated by a connected, directed graph $G = (V, E)$. There are n robots $R = \{r_1, \ldots, r_n\}$ that can be approximated by cylinders of radius r and height h. Each robot r_i is assigned a *task* to move from its start position s_i to a goal position g_i. A *scenario* is defined to be a collection of tasks $\{(s_1, g_1), \ldots (s_n, g_n)\}$.

A path is a map $p_i : \mathbb{N} \rightarrow V$. A path is *satisfying* if $p_i(0) = s_i$, $p_i(t_f) = g_i$ for some time index $t_f \in \mathbb{N}$, and for any $0 \leq t < t_f$, $\{p_i(t), p_i(t+1)\} \in E$ or $p_i(t) = p_i(t+1)$. A trajectory $\pi_i : [0, \infty) \rightarrow \mathcal{W}$ maps a time to coordinates in $\mathcal{W}$; the transformation from p_i to π_i bridges the gap between the abstract graph G and the continuous workspace $\mathcal{W}$.

The trajectories π_i, π_j of two robots are *conflict-free* if and only if the bodies of the robots i, j never intersect when they follow the trajectories π_i and π_j.

Problem 1 (Multi-Robot Path Planning Problem). *Given a workspace $\mathcal{W}$, a graph G, and a scenario S specifying the start and end points for n robots, find trajectories $\pi_1, \ldots, \pi_n$ that are satisfying and conflict-free.*

5.3 Graph Data Structure

Graph G is defined in Section 5.2. Each edge e_i is assigned a length d_i that is the Euclidean distance between its two endpoints. A cost function $c(r_i, e_j, t)$, described in the next section, assigns a cost to each edge for each robot; this function also accepts self-referencing edges (v_i, v_i) and returns the cost of waiting at that node. The graph contains 3 unique types of nodes: parking nodes, land nodes, and air nodes. Parking nodes exist only on the sides of roads, where, as will be explained in the next section,

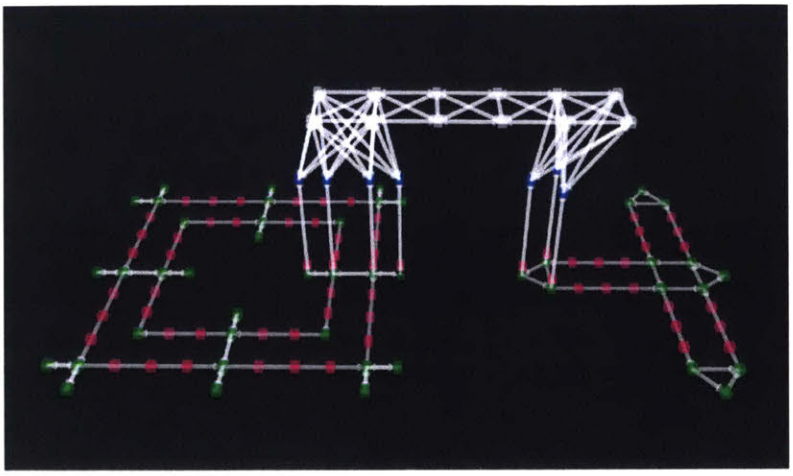


Figure 5-1: Graph structure of the environment. Land nodes are green; land waypoint nodes are purple; air nodes are blue or white and connected by white edges.

they are critical for guaranteeing feasible solutions. Land nodes represent roads, and air nodes represent chunks of air where only a single flying car can be present. There is a layer of air nodes above the ground, as well as "interface" air nodes that connect the ground and air layers. The 3 node types create 4 types of edges: takeoff edges, air edges, landing edges, and ground edges. Each type of edge has a unique velocity associated with it. Fig. 5-1 shows the graph structure of an example map. Green nodes and edges on the bottom represent the road system; blue nodes represent air nodes that interface between the ground and the white air layer.

5.4 Objective Function

We assume two modes of locomotion: flying and driving, each with an associated power consumption, P_f and P_d, and velocity, v_f and v_d. For the purposes of this study we assume that power consumption and velocity remain constant for each mode of locomotion. We calculate the energy cost of each edge connecting two nodes separated by a distance d; in the case of flight, we also consider the gain in potential energy during takeoff caused by lifting the mass m of the vehicle from z_1 to z_2. The work associated with flying is

$$W_f = \frac{P_f d}{v_f} + mg \cdot max(z_2 - z_1, 0) \tag{1}$$

and with driving is

$$W_d = \frac{P_d d}{v_d} \tag{2}$$

We can then calculate the cost of edge e_i with the function

$$c(e_i) = \mu \frac{W}{W_{max}} + (1 - \mu) \frac{t}{t_{max}} \tag{3}$$

where $0 \leq \mu \leq 1$ and W_{max} and t_{max} are the maximum possible energy and time of any edge in the graph. μ is a tuning parameter that allows the planner to weight energy and time in the cost function. When μ is 0, the function only weights time and the planner will simply minimize time. When μ is 1, the function only weights energy and the planner will minimize energy consumption. The ability to tune μ allows users to choose between time and energy. For example, if implemented for a flying taxi, a customer who values time over energy can pay more to make μ lower, so that the planner will value time more than energy.

5.5 Battery Life

We added a battery life constraint to the planning problems by estimating the current consumption over each edge. Given battery capacity C in Ah, battery voltage $V(r_i)$ as measured for robot i, average power draw $P(e_j)$, distance $d(e_j)$, and velocity $v(e_j)$, an approximate current $I(r_i, e_j)$ and current consumption $I_c(r_i, e_j)$ can be calculated by

$$I(r_i, e_j) = \frac{P(e_j)}{V(r_i)} \tag{4}$$

$$I_c(r_i, e_j) = \frac{I(r_i, e_j) d(e_j)}{v(e_j)} \tag{5}$$

In our planners, battery capacity is not allowed to fall below a specified level C_{low}.

5.6 Priority Planning with SIPP

Priority planning assigns a priority number to each robot under consideration and plans collision-free paths for each robot in order of its priority; as it iterates through the robots, it saves the robots' paths and the nodes within a radius r of each path to the obstacle space. Therefore priority planning is non-optimal and in a naive implementation is not guaranteed to find a solution. However, if the graph on which the robots are traveling is structured as a well-formed infrastructure in which the robots start and finish on endpoints that do not obstruct the paths of other robots, then every robot is guaranteed to have a feasible path [28]. The notion of a well-formed infrastructure easily extends to 3D space so that a path is guaranteed as long as each flying car starts at a parking node, has an empty parking node as its endpoint, and does not travel through other parking nodes on its way to its goal.

Our Safe Interval Path Planning algorithm extends the one in [42] and finds collision-free paths for robots with multimodal locomotion by extending A* to the time domain, as outlined in Algorithm 1. Unlike an approach such as cooperative A* that discretizes the time domain [29], SIPP allows search in continuous time. It achieves this by compressing time for each node into "safe intervals" in which no obstacle passes through the given node. For example, a node through which no vehicles pass has a safe interval $(0, \infty)$. If a vehicle passes through the node from times 4.5 to 5.5, the node now has two safe intervals $(0, 4.5)$ and $(5.5, \infty)$. We define a state s to be a node-interval pair, in which a node is associated with one of its safe intervals. $T(s)$ stores the earliest-time arrival, $C(s)$ stores the lowest cost-to-come, and $h(s)$ represents the estimated cost-to-go. Arrivals in states are stored in the set $OPEN$ as tuples, so that the state s_i, the cost-to-come c, the sum of the cost-to-come and estimated cost-to-go e, and the arrival time t are associated with that arrival. $cost(s)$ returns the cost associated with state s. $state(v, i)$ returns the state associated with node v and time i, while $node(s)$ returns the node associated with state s.

Algorithm 1: Safe Interval Path Planning

```
1  Algorithm SIPP
2      Input: G, set of safe intervals, robot index i, initial state s_0, goal state s_g
3      Output: collision-free path, updated safe intervals
4      OPEN = ∅;
5      insert (s_0, c = 0, e = h(s_0), t = start time) into OPEN;
6      C(s_0) = 0;
7      T(s_0) = starting time;
8      while s_g not expanded do
9          remove (s, c, e, t) with the smallest e-value from OPEN;
10         successors = getSuccessors(s, t);
11         foreach s' in successors do
12             if s' was not visited before then
13                 C(s') = T(s') = ∞;
14             end
15             if C(s') > c + cost(s') or T(s') > time(s') then
16                 C(s') = min(c + cost(s'), C(s'));
17                 T(s') = min(T(s'), time(s'));
18                 insert (s', c + cost(s'), c + cost(s') + h(s'), time(s')) into OPEN;
19                 remove all tuples from OPEN with s = s' and c >= C(s') and
                     t >= T(s')
20             end
21         end
22     end
23     recover p_i by backtracking from g_i;
24     return p_i;
```

Algorithm 2: Successor Function

```
1  Function getSucccessors(s, t) : successors is
2      successors = ∅;
3      foreach v' in Neighbors(node(s)) do
4          L = interval list of v';
5          dt = time to traverse (node(s), v');
6          foreach safe interval i in L do
7              if end_time(i) > t + dt and start_time(i) < end_time(interval(s))
                then
8                  t_arr = earliest arrival time in L;
9                  cost = c(r_i, (node(s), v'), t_arr − t);
10                 if v' is the goal or v' is not parking then
11                     s' = state(v',i);
12                     associate s' with time t_arr and cost;
13                     insert s' into successors;
14                 end
15             end
16         end
17     end
18     return successors;
19 end
```

The successor function, outlined in Algorithm 2, takes a state and the robot's arrival time to that state. It then finds all overlapping safe-intervals for all neighboring nodes, storing each node-safe interval pair as a successor state, along with the earliest arrival time and the cost. In [42], SIPP guarantees time-cost optimality by storing only the earliest arrival to a state, as any later arrival would simply have a subset of the earlier arrival's options.

In order to optimize energy as well as time, the choice of successor becomes more complicated. One cannot consider only the earliest arrival, as that may not have the lowest cost. Nor can we ignore all but the lowest cost arrival, as earlier arrivals could have more available safe intervals and thus a lower cost path overall. To maintain optimality, one would have to consider all arrivals to a state that have a lower cost than all previous arrivals, but this causes the number of successors considered to expand exponentially. We therefore sacrifice optimality (we already sacrificed multi-robot optimality by using priority planning) to speed up computation times by only

considering the earliest arrival and lowest cost arrival to a state.

Once a path is found, the safe-intervals are updated. Nodes are considered unsafe for the time that the robot is within a certain radius of them. Following that, the next robot plans its path, using the updated set of safe-intervals.

Algorithm 3: MPP-ILP-Optimization

```
1  Algorithm Model Construction
       Input: G, T_max
       Output: ILP model
2      construct a time-expanded version of G by making a copy of every node for
         every time step and linking nodes at time t_i to nodes that are reachable at
         time t_{i+1};
3      add arc capacity constraints;
4      add flow conservation constraints;
5      add scenario completion constraints;
6      add meet collision constraints;
7      add head-on collision constraints;
8      add battery-life constraints;
9      add objective function;
10     return ILP model
```

5.7 Multi-commodity flow with ILP

Our algorithm is based off the algorithm found in [30]. In [30], the MPP is reformulated as a multi-commodity network flow problem. This is achieved by discretizing the time domain and connecting nodes in the time domain to each other with arcs, forming a new graph G'. Multi-commodity flow can be solved by adding a binary variable x_{ij} representing the presence or absence of each robot i to each arc j, and then adding constraints that prevent robots from colliding with each other, as outlined in Algorithm 3. The specifics of the graph construction and constraints can be found in [30]; however, we added an additional battery life constraint and a new objective. Given binary variables x_{ij} and $|G'|$ arcs in G', the battery constraint for each robot i can be expressed as

$$\sum_{0 \leq j < |G'|} I_c(r_i, e_j) \cdot x_{ij} \leq C_{low} \tag{6}$$

and the objective function as

$$\min \sum_{0 \leq j < |G'|, 1 \leq i \leq n} c(r_i, e_j) \cdot x_{ij} \tag{7}$$

In [30], the distance between nodes is assumed to be 1, and all of the robots move with the same velocity. In order to accommodate multiple velocities we designed a new graph structure. Given a basic graph of the road system and desired air, takeoff, landing, and ground velocities, we add "waypoint" nodes between the existing ground nodes so that the time it takes to move between nodes takes about 1 timestep. We also space out the aerial nodes so that the time it takes to travel between nodes takes about 1 timestep. In the planning step we then assume that it takes exactly 1 timestep to travel between any two adjacent nodes; this way, the velocity a robot must travel between adjacent nodes remains close to, but not exactly, the original desired velocity. Our formulation differs from the original formulation in that our objective function incorporates both time and energy; our graph contains arcs of different lengths that are calculated based off of the robots' velocities; and we include a battery voltage constraint.

Chapter 6

Experiments & Results

6.1 Flying Monkey Experiments & Results

6.1.1 Software Architecture

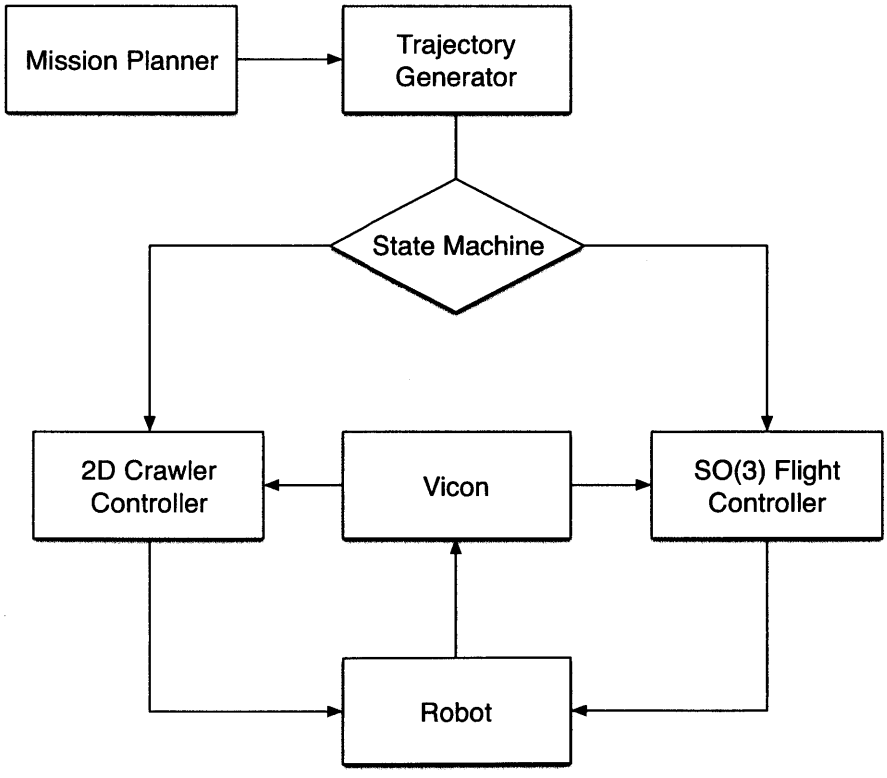


Figure 6-1: Software architecture for controlling the Flying Monkey [34].

The control stack for the robot is written in C++ using the ROS [43] (Robot Operating System) framework. The incorporation of ROS greatly simplifies the transition between computation on the base station and onboard the robot.

As seen in the architecture diagram in Fig. 6-1, a high level mission planner reads in user input in the form of waypoints or time parameterized trajectories. The trajectory generator then sends calculated desired position commands to a state machine which analyzes the position commands and governs the mode of locomotion of the robot, delegating the control to either the 2D crawler controller for terrestrial, planar locomotion, or to the SO(3) flight controller for the current phase of the mission.

The integration of the finite state machine into ROS and C++ allows us to run closed loop controllers by using pose and position estimates from the Vicon motion capture system and the attitude state estimation on-board the MAVs.

The selected controller receives the robot's current pose and position from the motion capture system and the desired position from the trajectory generator. Using this information, the controller computes a desired attitude and thrust setpoint and transmits them to the robot through a 900MHz wireless uplink at 100Hz. With these desired attitude and thrust measurements and its own onboard pose estimates, the robot computes and executes the appropriate motor commands to attain the desired setpoints. This low-level control loop runs at a rate of 1kHz onboard the robot.

Pictures of the Flying Monkey in action can be seen in Fig. 6-2.

6.1.2 Energetics at hover

To obtain the energetics of the Flying Monkey, we measured the battery voltage of the robot using an onboard battery monitor and designed a custom power board consisting of a MAX4172 Current-Sense Amplifier to measure in-flight current draw. Fig. 6-3 shows the power draw of the standalone Dragonfly quadrotor and the Flying Monkey at hover. We empirically determined the power draw of the Dragonfly and the Flying Monkey to be 9.7450W and 10.5928W respectively.

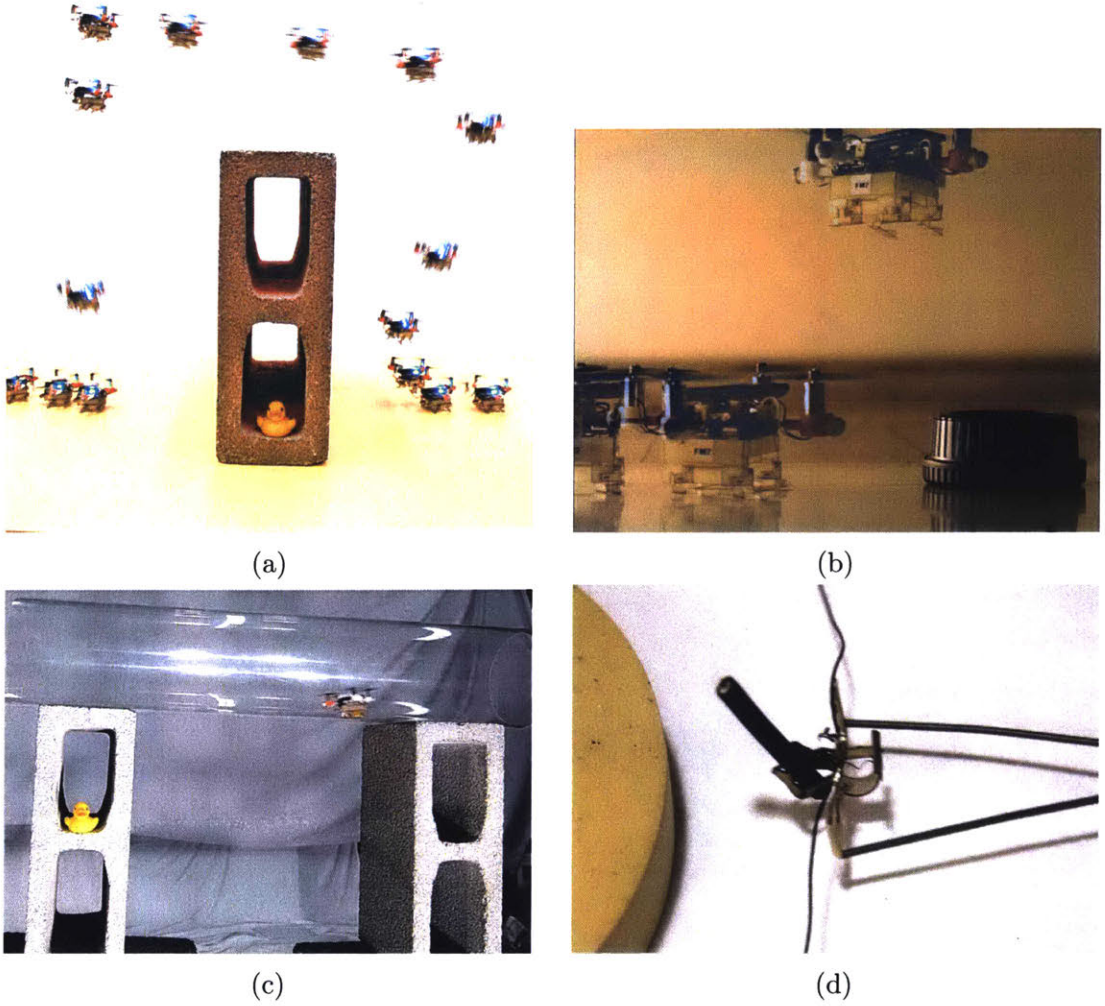


Figure 6-2: (a) The Flying Monkey traversing a cinder block. (b) A close-up of the Flying Monkey taking off. (c) The Flying Monkey crawling through a pipe. (d) The gripper picking up a wire. More footage can be seen at `http://mrsl.grasp.upenn.edu/yashm/2017ICRA2016.mov`

6.1.3 Energetics during crawling

Next, to determine the energetics during terrestrial locomotion, we recorded the voltage and current drawn by the Flying Monkey while crawling at its maximum speed of 0.16 m/s on a flat surface. We found that the power drawn while crawling was 0.6435W – over 93% lower than the power consumption during flight. This is shown in Fig. 6-4. The figure shows a 45 minute data log, over which the battery voltage only dropped by a few millivolts, confirming the lower power draw for a ground robot.

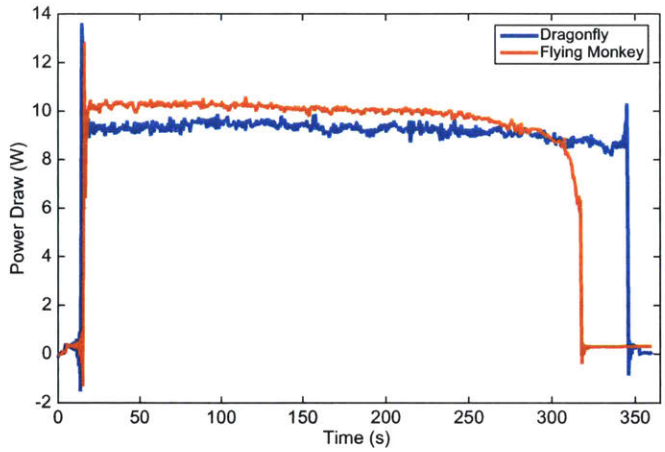


Figure 6-3: Hover power draw of the Dragonfly quadrotor $P_{avg} = 9.7450W$ and the Flying Monkey $P_{avg} = 10.5928W$.

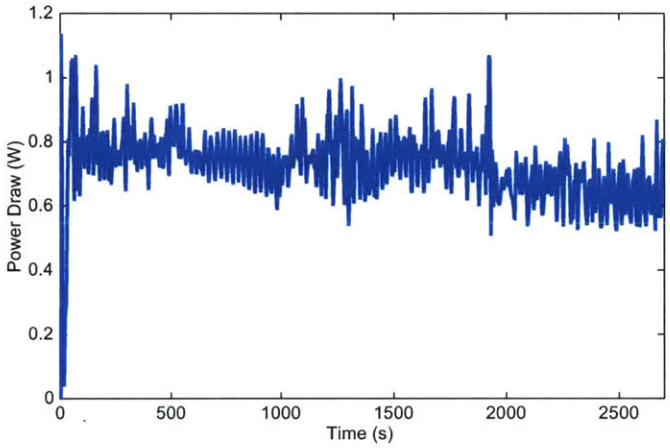


Figure 6-4: Power draw of the Flying Monkey at 0.1m/s . $P_{avg} = 0.6435$ W.

6.1.4 Cost of transportation

The *Cost of transportation* (COT) measures the energy efficiency of a mode of transport. Since it is a dimensionless quantity, it allows the comparison of different modes of locomotion. It can be calculated as

$$COT = \frac{P}{mgv} \tag{6.1}$$

To calculate the COTs, we assume that the power P consumed by the Flying Monkey while flying at a velocity v of 1m/s was equal to the power drawn at hover.

The results can be seen in Table 6.1.

	Energetics				
Vehicle	Speed (m/s)	Run Time (min)	Max Dist (m)	Ave Pow (W)	COT
Dragonfly	1.0	5.75	345	9.75	41.4
Flying Monkey	1.0	5.3	317	10.6	36.0
Crawler	0.16	45	432	0.64	13.7

Table 6.1: Comparison of the energetics of the Dragonfly, Flying Monkey, and crawler

This analysis builds a strong case for ground robots, showing that a purely aerial robot has a significantly higher cost of transportation compared to a ground robot. However, with some compromise and by combining the two locomotion modalities, the Flying Monkey can harness the potential of aerial locomotion while keeping the COT low.

6.1.5 Gripper

The maximum gripping load was measured by testing what weights the gripper could support before grasp failure. Weights were suspended from a segment of a drinking straw, and the gripper was clamped around the straw. The weights started at 1.4g, then 2g, then increased in 1g increments until the straw slipped from the gripper. A test was considered a failure if the straw slipped out of the gripper and a success if it did not. The results are shown in Fig 6-5. The value of the maximum gripping load can be increased by a surface treatment for a higher friction coefficient.

6.1.6 Regulation and Time Parameterized Trajectory Tracking

Fig. 6-6a shows the performance of the robot at different speeds while trying to crawl from an initial position to a fixed destination: the solid lines in red show its performance without a controller, while the dotted lines show its performance using

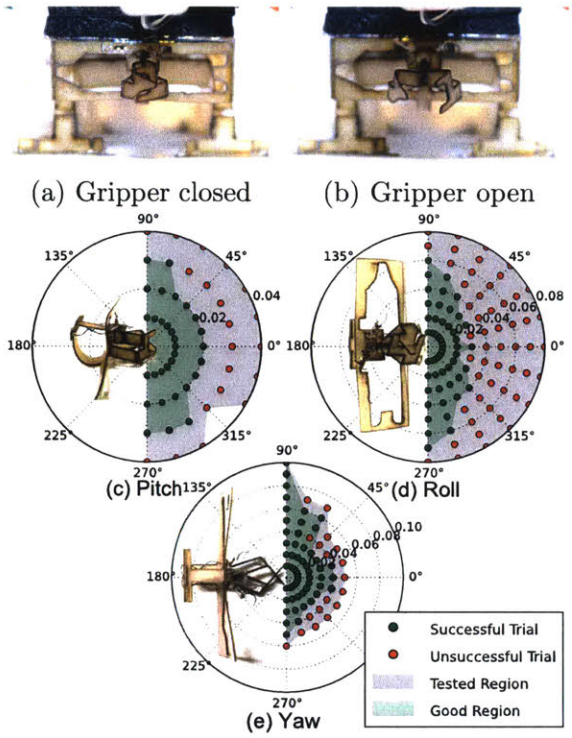
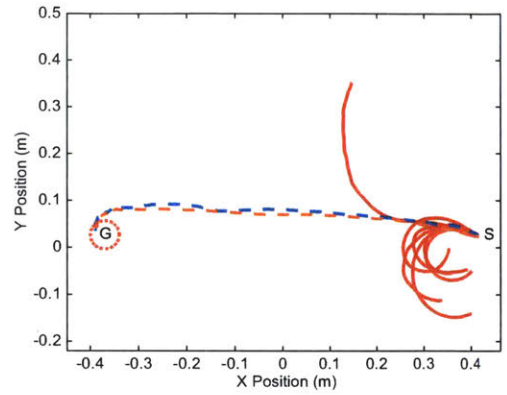
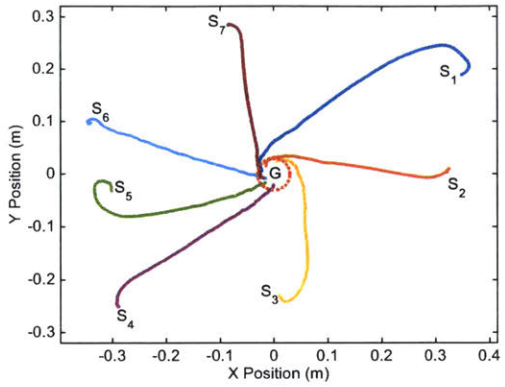


Figure 6-5: The gripper mechanism(a,b), Gripper pull-out force data in (c) pitch, (d) roll and (e) yaw. Radial axes are displayed in .02N segments, and rotational segments are in 15-degree increments. Trials with successful grasps are shown in green, and failures in red.

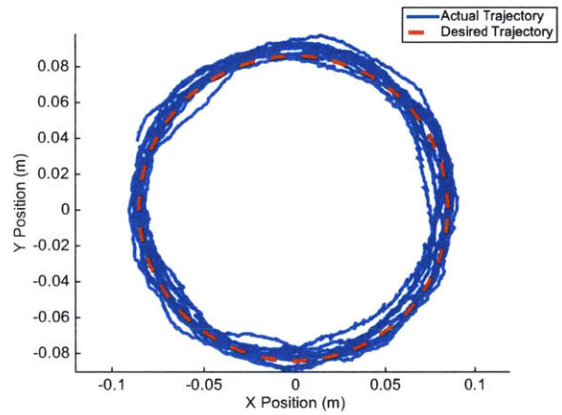
the controller described above. Fig. 6-6b shows the performance of the robot under feedback control crawling to a constant position from different initial positions and orientations. Fig. 6-6c shows the performance of the robot tracking a reference moving in a circular trajectory of radius $8cm$ centered at the origin at approximately $-0.21rad/s$. Fig. 6-6d shows the performance tracking the Lissajous curve described by $x(t) = 0.2cos(-0.01t)$ $y(t) = 0.2sin(-0.02t)$.



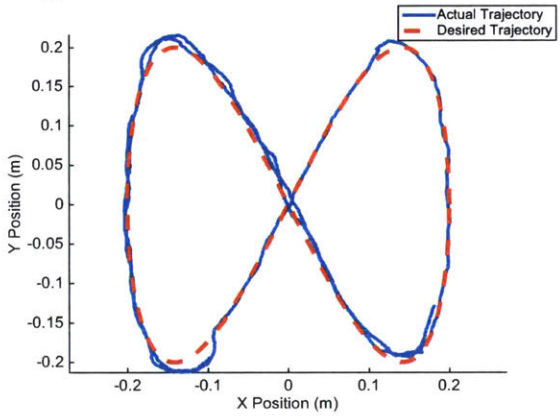
(a) Crawler performance with and without active controller.



(b) Position regulation starting from different initial positions ($S_1 - S_7$) and orientations to the goal G.



(c) Trajectory tracking performance along a circle.



(d) Trajectory tracking performance along Lissajous curve.

Figure 6-6: Controller performance

6.2 Flying Car Experiments & Results

6.2.1 System Architecture

This project was written in the Robot Operating System (ROS) environment, and the overall architecture of the system can be seen in Figure 6-7.[1] The process starts with a map configuration file, in which the road layout, as well as any no-fly zones and helipads, is specified. A "scenario specification" file, which specifies the parking

[1]All code for the Flying Car project can be accessed at `https://github.com/braraki/flyingcar_vizplan` and `https://github.com/braraki/flyingcar_ros`.

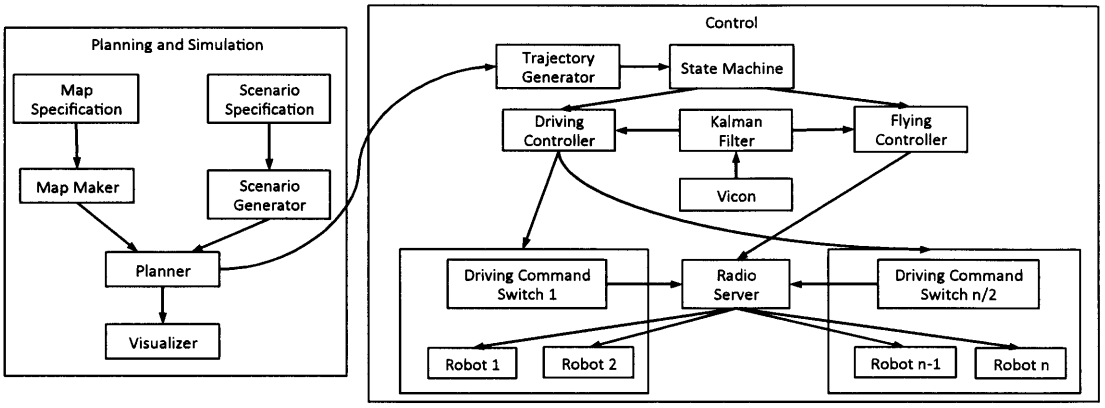


Figure 6-7: System architecture

nodes at which each robot will begin and end its path, can also be supplied. If not, a random scenario will be generated. The generated 3D environment and the scenario are supplied to the planner, which calculates a set of collision-free paths for the robots. The map and the paths are sent to a visualizer (such as RViz) or to a trajectory generator. The trajectory generator smooths the path into closely spaced waypoints and send them to a state machine that determines based off of the node type whether to send commands to the driving or flying controller, and if it chooses the flying controller, whether to takeoff, land, or fly. The flying controller sends flight commands directly to a radio server. We found that a single Crazyradio can support communication with at most 2 Crazyflies reliably. The driving controller uses a slower communication protocol than the flying controller, so we found that we had to serialize the driving commands being sent to a single radio. This was accomplished using a "driving command switch" that reads the driving commands for two robots and sends the commands in serial to the radio server.

6.2.2 Energetics

Experiments were conducted to determine the power consumption and battery life of the robots using different modes of locomotion. The results can be seen in Table 6.2. Adding wheels to the base of the Crazyflie reduced its flying time by ~12%. However, the battery life of the Flying Car using wheels was ~8.4 times longer than that of

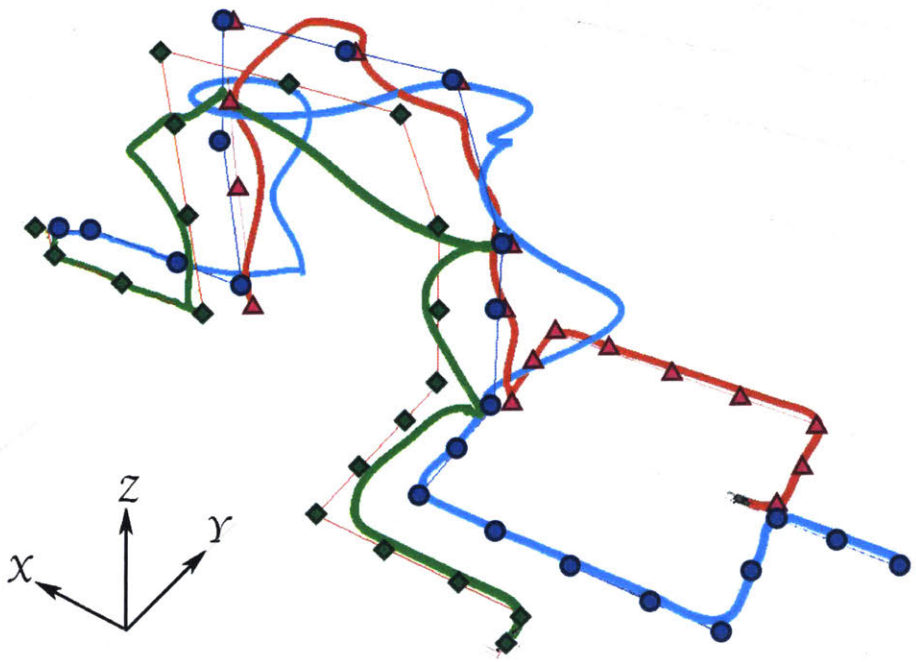
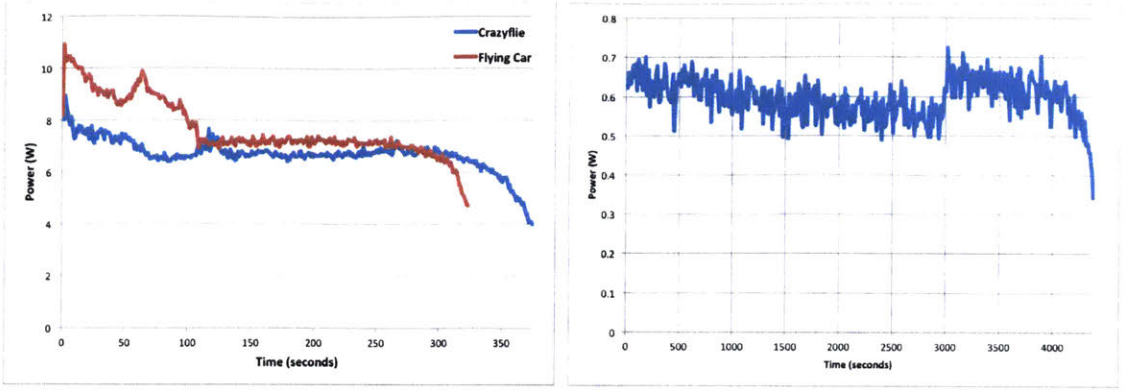


Figure 6-8: Driving and flight paths of three flying cars. Bold lines are recorded trajectories and dots indicate commanded waypoints of the path with the corresponding color.

	Energetics				
Vehicle	Speed (m/s)	Run Time (min)	Max Dist (m)	Ave Pow (W)	COT
Crazyflie	0.3	5.7	103	6.86	81.8
Flying Car	0.3	5.0	90	7.83	64.9
Wheels	0.1	42	252	0.60	14.9

Table 6.2: Comparison of the energetics of the Crazyflie, Flying Car, and wheel base

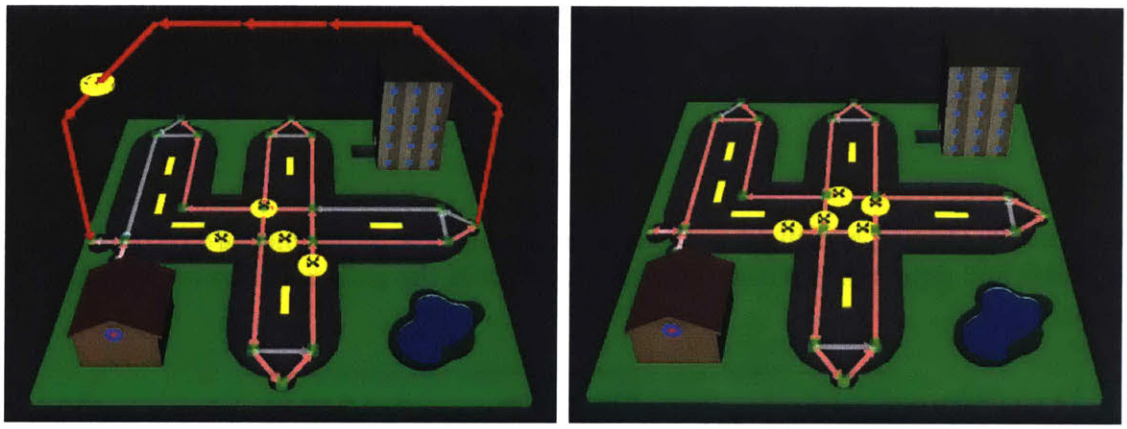
the Flying Car when it flew, and about ~7.4 times longer than that of the Crazyflie. This implies that at the speeds used in the simulations, the flying car could travel 90 m by flying and 252 m by driving. The cost of transport of driving is 87% lower than that of flying, corroborating the results found in [34] and supporting our argument that flying can serve as a high-cost, high-speed transport option while driving serves as a low-cost, low-speed option.



(a) Power draw while flying

(b) Power draw of the wheel base

Figure 6-9: Power draw comparison of the Crazyflie, Flying Car, and wheel base



(a) In priority planning, the lowest priority car flies over traffic

(b) The optimal planner coordinates all robots simultaneously so none have to fly

Figure 6-10: Simulation experiments

6.2.3 Simulation

Simulations were run to test the abilities of the system. Simulations were performed on a Dell Precision M4800 notebook with 16 GB of RAM and a 2.90 GHz Intel Core i7-4910MQ CPU. Selected simulations are shown in the attached video. We ran simulations on over 20 maps; the largest map, shown in Fig. 6-12, had 934 nodes. We could plan for up to 80 vehicles in the large map; Fig. 6-12 shows paths for 20 vehicles. Fig. 6-10a shows an example of how a flying car could fly over traffic instead of waiting at an intersection for other cars to pass. Fig. 6-10 is also a comparison of the priority planner and the optimal planner; in Fig. 6-10a, the priority planner has

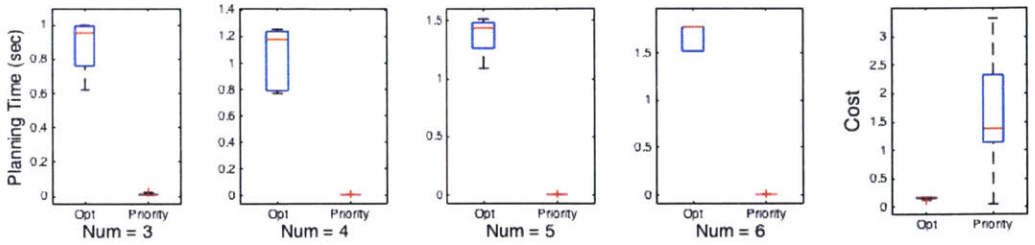


Figure 6-11: Planning times vs. number of vehicles and average cost of paths

the lowest priority car fly over the other cars in a high-cost path; in Fig. 6-10b, the optimal planner finds a way to coordinate the cars in order to main low-cost driving paths for all of them. Fig. 6-11 shows a comparison of the run times and the path cost returned by the optimal and priority-based planners. 20 trials were run for both planners for 3-6 vehicles in the 125-node map in Fig. 6-10. Note that planning time did not vary significantly with the number of vehicles. The plot on the right of the figure shows the average cost, which is a dimensionless value, of all 80 trials for both planners. The mean cost of the optimal planner was 0.155, or 91% lower than the 1.76 mean cost of the paths returned by the priority-based planner. Meanwhile, the mean planning time of the priority-based planner was 0.106 seconds, 0.85% of the value of the 1.25-second mean planning time of the optimal planner. Therefore the two planners offer a clear trade-off in planning time versus cost optimality. The optimal planner is slow but returns low-cost paths, and the priority-based planner is fast but returns relatively high-cost paths.

6.2.4 Physical Experiment

Finally, a physical experiment using 8 flying cars was conducted. We provided a scenario in which each robot had to travel from a start position to a goal position. The full experiment can be viewed in the video attachment. A 4.2m x 2.1m map was constructed using velvet mats. Fig. 6-13a shows the experiment in progress; Fig. 6-13b shows the same moment in time in simulation. The map of the demonstration was constructed so that the road system consists of two disconnected regions; in order

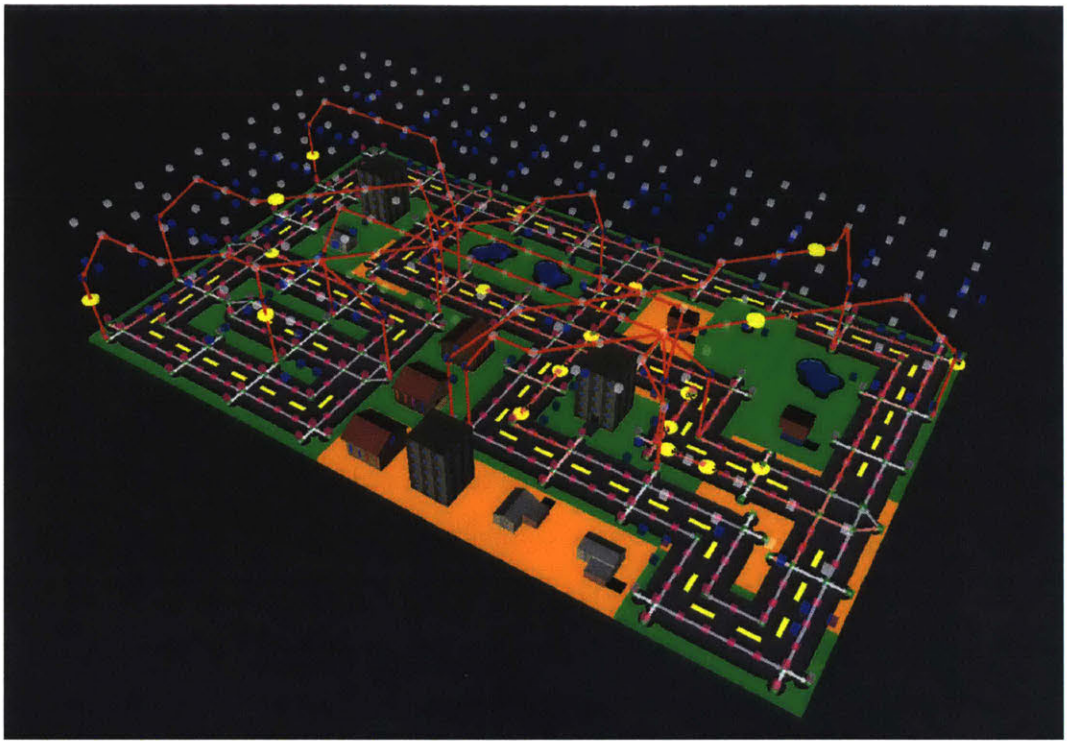
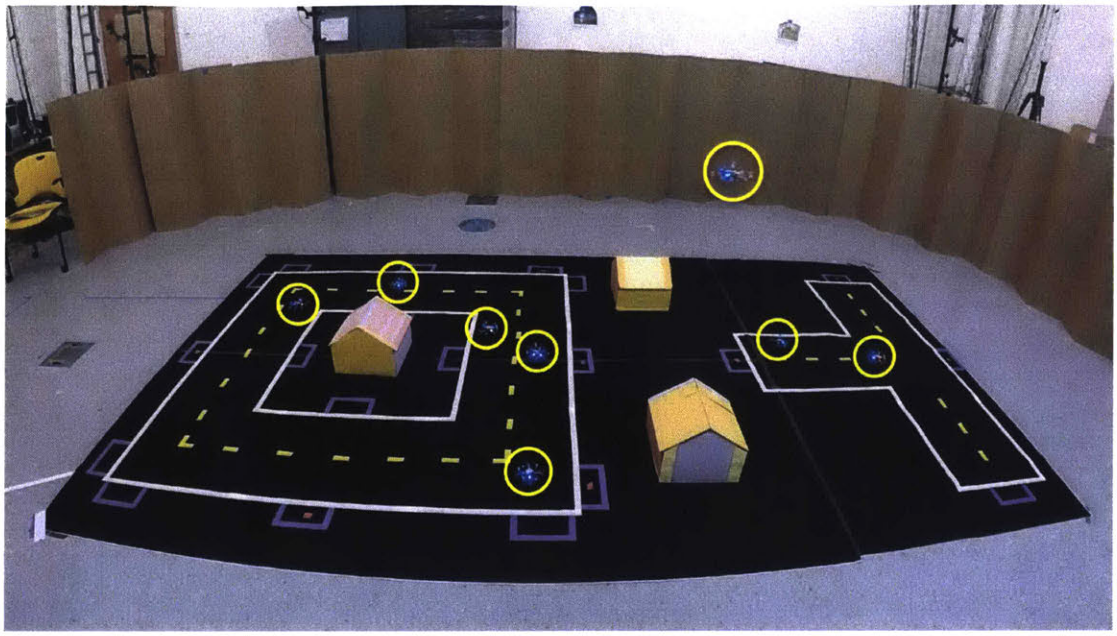
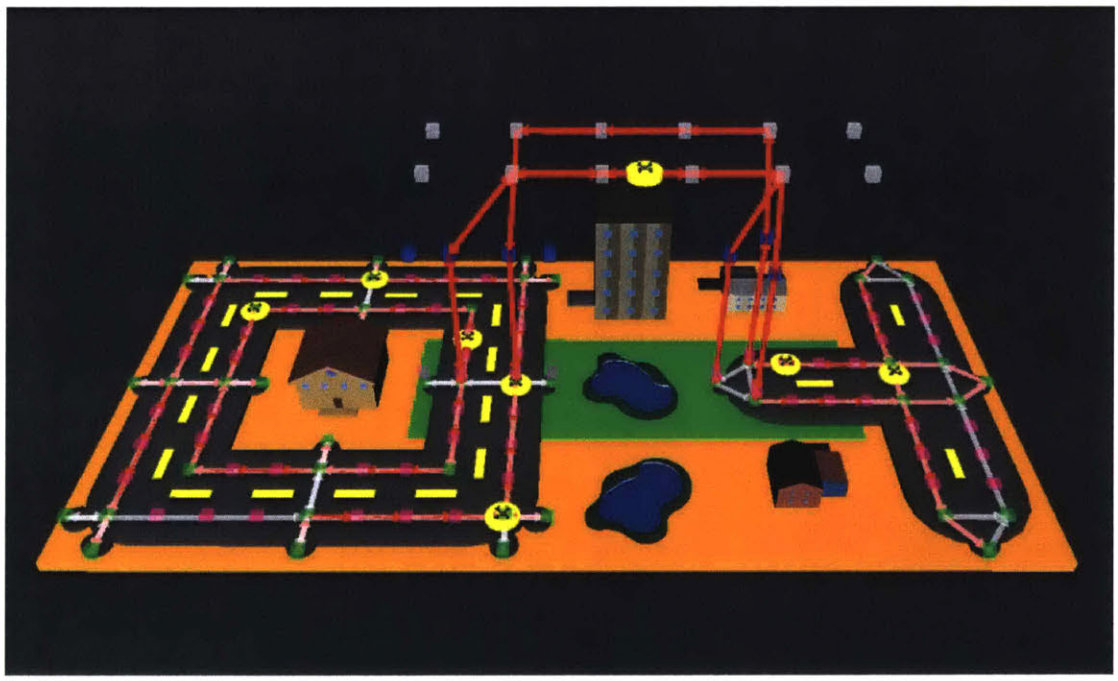


Figure 6-12: Paths for 20 vehicles with flying and driving.

to travel from one region to another, the flying cars must fly above the gap between the two. This situation is common in the real world; for example, the gap between the road sections could be a wall, a lake, a fallen tree, or any number of other obstacles. Moreover, only a narrow corridor of land, colored in green in Fig. 6-13b, has available airspace; the rest of the land, colored in orange, is a no-fly zone. The narrow corridor forces the cars to coordinate their motion, and such corridors of restricted flight are also a possibility in the future; for example, delivery drones or personal flying cars may not be allowed to fly over residential areas at night. The recorded flying and driving trajectories of three of the flying cars is shown in Fig. 6-12. The robots successfully completed the objective of the experiment by traveling from their start positions to their goal positions in collision-free paths.



(a) Flying cars traversing a town



(b) The same moment in simulation

Figure 6-13: Experimental and simulated flying cars

Chapter 7

Future Work & Conclusion

7.1 Future Work

There remains much to explore in the field of air-and-ground vehicles, particularly in path planning. As batteries become lighter and more energy dense, the payload capacity and battery life of air-and-ground vehicles should increase, making them more useful for tasks such as package delivery, surveillance, and exploration. This thesis covers only the problem of routing multiple air-and-ground vehicles from a set of start points to a set of end points. An interesting extension to the work in Chapter 5 would be to add recharging stations to the map. This could be implemented by creating self loops on recharging station nodes that increase the battery level of a vehicle on each time step. Package delivery and taxi services are other possible extensions of this work. Our graph structure and algorithms would have to be extended to enable pick up and drop off of packages/passengers based on priority (such as how much a passenger is willing to pay for speed).

Two of the most compelling use cases for air-and-ground vehicles are surveillance and mapping of unknown environments. Both of these tasks require sensors, which would be difficult to integrate into a miniature air-and-ground vehicle the size of the Flying Car due to limited payload capacity. Mid-sized commercially available quadrotors are a promising platform for building a surveillance/exploration air-and-ground vehicle. New algorithms would have to be made or modified to accommodate the

two locomotion options. For example, although aerial locomotion is more dangerous and energetically expensive than ground locomotion, it could allow the robot to fly over rubble or to reach the tops of walls and high platforms. Meanwhile, ground locomotion could be used to move stealthily, to closely inspect an area of interest, or to drive through narrow openings.

Another interesting extension to path planning for robots with multimodal locomotion is to extend the problem to path planning for heterogeneous robots with arbitrary dynamics and capabilities. For example, imagine this team of robots: Robot 1 is a heavy ground vehicle with a high-resolution laser scanner; Robot 2 is a quadrotor with low battery life and a low-resolution camera; and Robots 3 and 4 are air-and-ground robots with radiation measuring devices. Given an unknown environment, what is the best way to explore it and position Robots 3 and 4 to measure radiation levels? Such a team of robots would need to closely cooperate in order to make the most of their complementary capabilities.

7.2 Conclusion

Two air-and-ground vehicles were designed and tested in this thesis. The Flying Monkey is a 30g robot that is capable of grasping, flying, and crawling. The crawler is a sophisticated 1-DOF 6-bar mechanism with 4 four-bar hip joints that control 8 feet; it also has a series four-bar element that constrains the 6-bar mechanism to rotational motion in a single plane. Furthermore, the design contains an SMA coil-actuated gripper with a built-in passive spring. The crawler consists of 66 linkages and is strong enough to carry the 24g Dragonfly quadrotor yet weighs only 5.1 grams, well under the 10g payload limit of the Dragonfly.

The crawler was designed to have 1 DOF in order to minimize the number of actuators required to operate it; we compensated for the lack of a second DOF by designing a nonlinear controller that uses the yaw torque of the Dragonfly's propellers to steer the crawler. We then performed a number of experiments using the Flying Monkey that showed that the controller worked as expected.

The goal of the Flying Monkey was to demonstrate the many capabilities of an air-and-ground vehicle. We therefore performed a number of experiments demonstrating that the Flying Monkey is capable of everything we promised - it was able to fly over a cinder block that it could never have crawled over; it was able to crawl through a pipe and then fly out the opening; and the gripper was able to pick up and hold a piece of wire. The success of the Flying Monkey shows that the dream of air-and-ground vehicles outperforming ground- or air-only vehicles is not just a dream, but that it has already happened.

We next presented the first two algorithms for multi-robot path planning for air-and-ground vehicles. One algorithm is based off of Safe Interval Path Planning, a form of priority planning that enables planning in continuous time. The second algorithm is a variation of multi-commodity network flow solved using integer linear programming that finds optimal paths for the robots.

The Flying Car was designed as a testbed for the multi-robot path planning algorithms. We therefore made its design as simple and robust as possible, presenting the first example that we know of of a differentially steered driving mechanism on a quadrotor. We then designed a software architecture that allows for the specification of map configurations and start-and-end scenarios for the path planning problem. We also overcame the challenge of connecting to as many as eight robots at once using a single computer by developing a radio switching node. And finally, we verified the functionality of our mechanical design, controllers, and algorithms by running them on 8 robots in a miniature town. This work on multi-robot path planning algorithms for air-and-ground vehicles represents the first attempt at generating collision-free paths for multiple vehicles in an environment that involves both flying and driving.

Thus, the Flying Monkey demonstrates that many useful functions can be packaged into a small air-and-ground vehicle, and the algorithms controlling the Flying Cars show that multi-robot path planning for air-and-ground vehicles is easily achievable. With further development, it is possible to see a future of flying cars and versatile robot teams not far beyond the horizon.

Appendix A

Figures

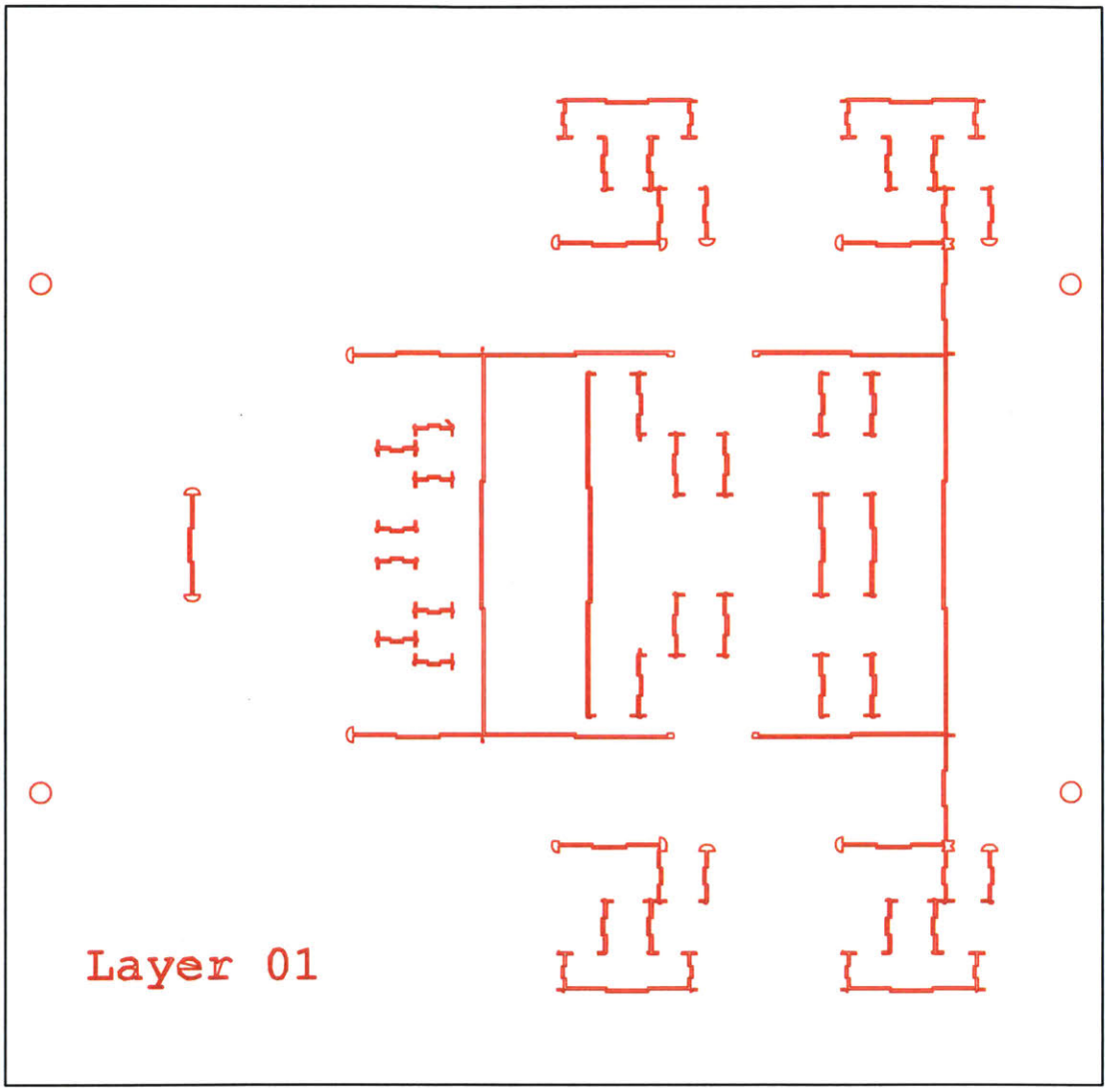


Figure A-1: Layer 1 of the Flying Monkey

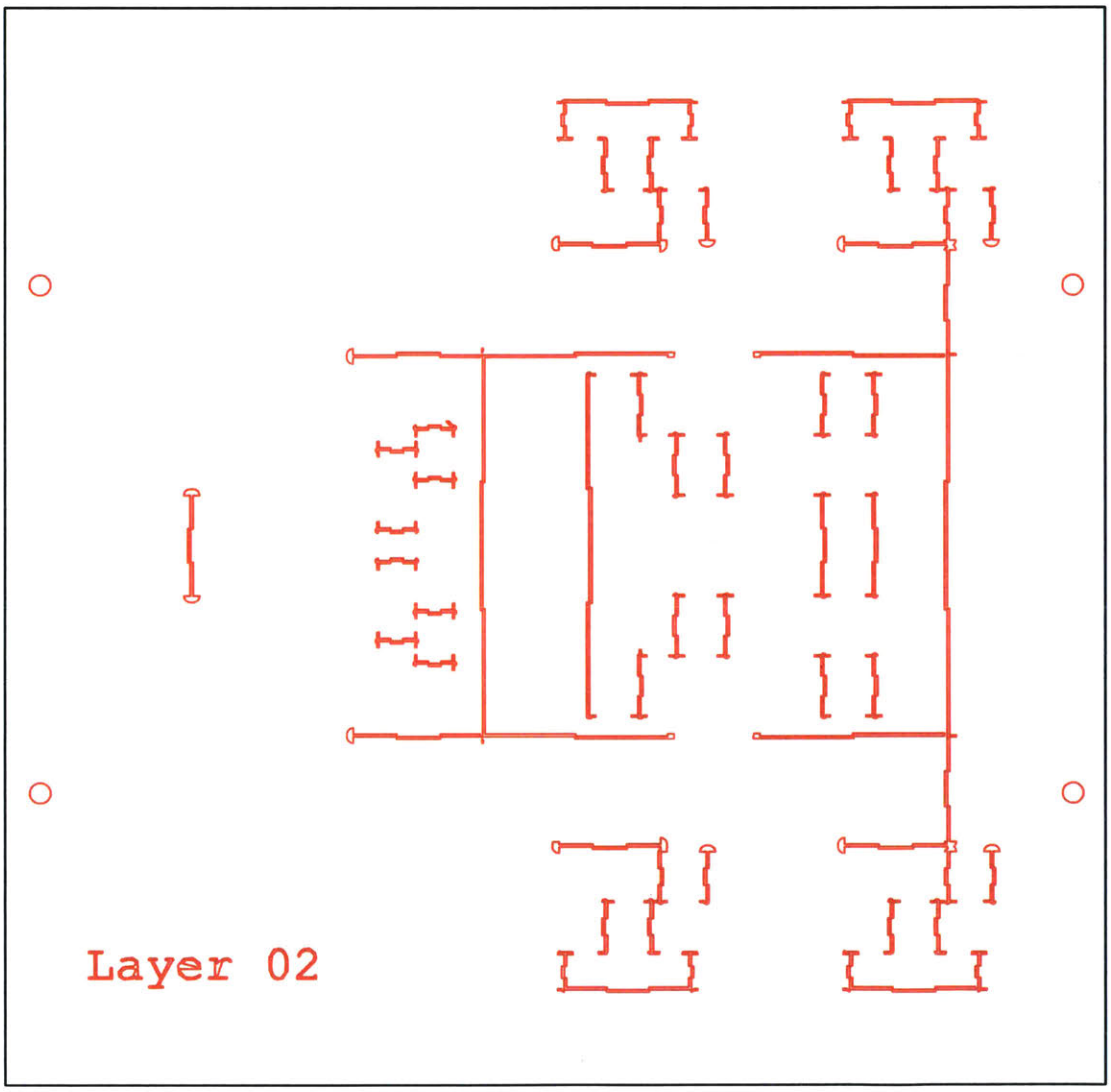


Figure A-2: Layer 2 of the Flying Monkey

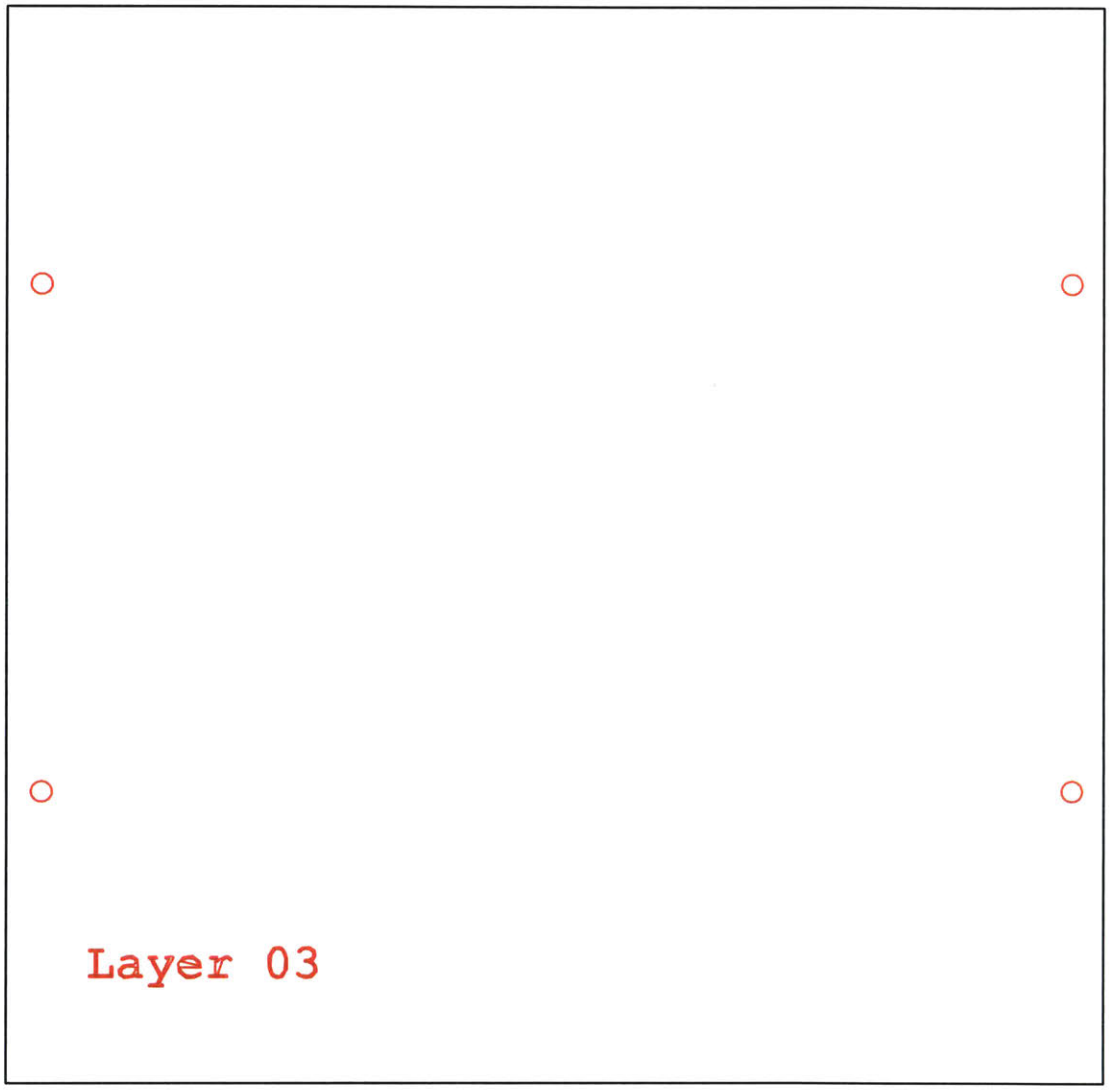


Figure A-3: Layer 3 of the Flying Monkey

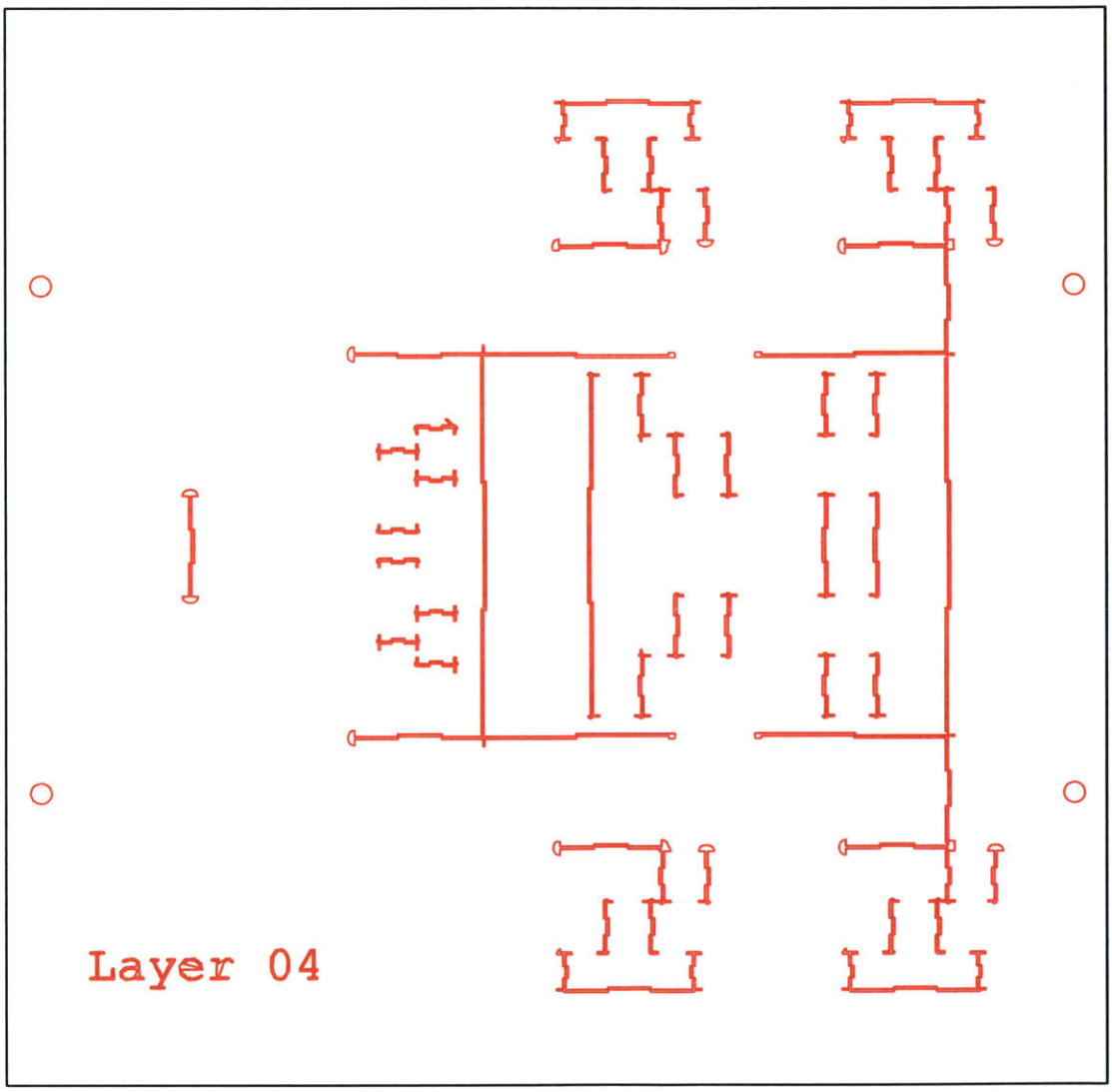


Figure A-4: Layer 4 of the Flying Monkey

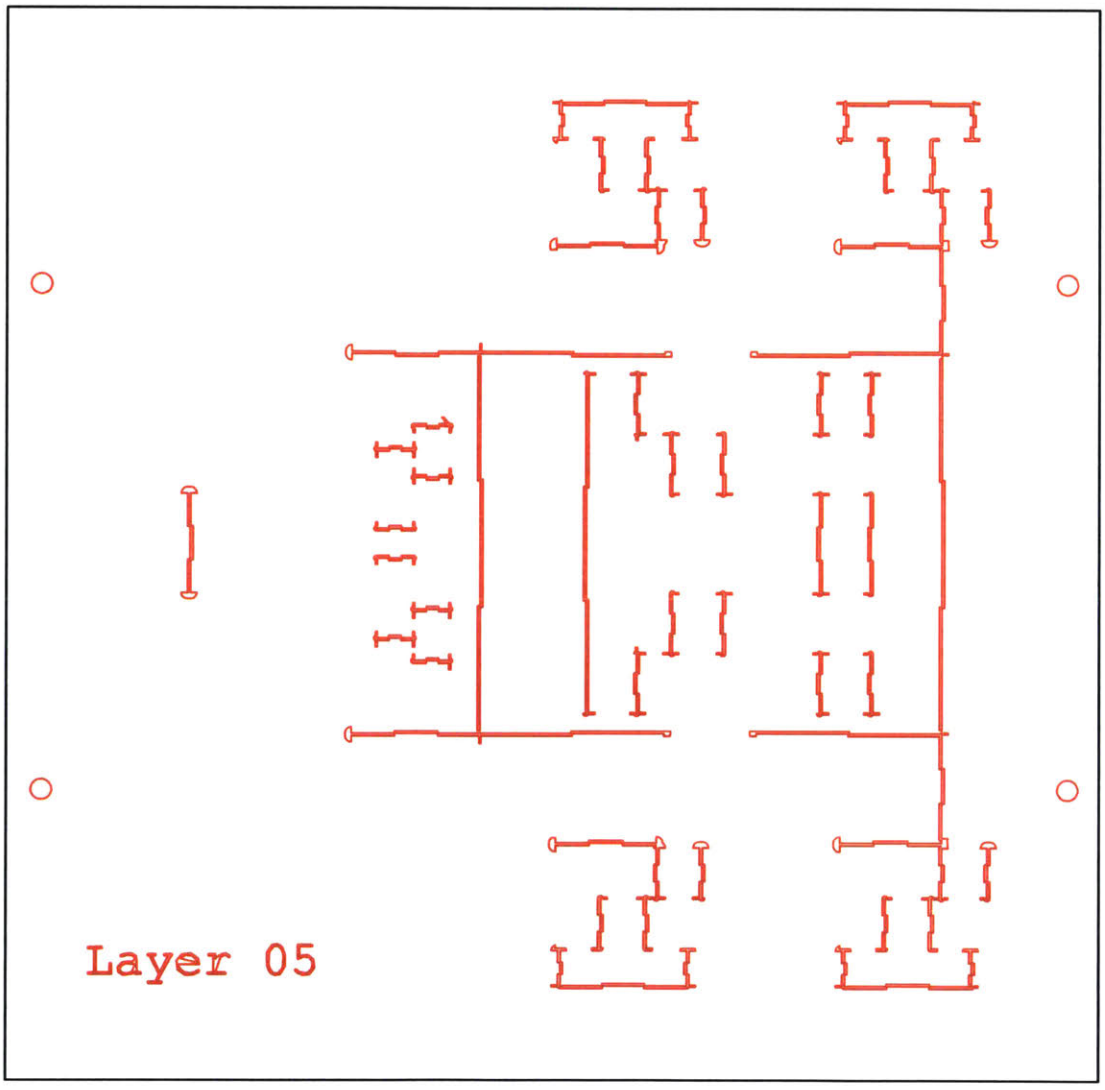


Figure A-5: Layer 5 of the Flying Monkey

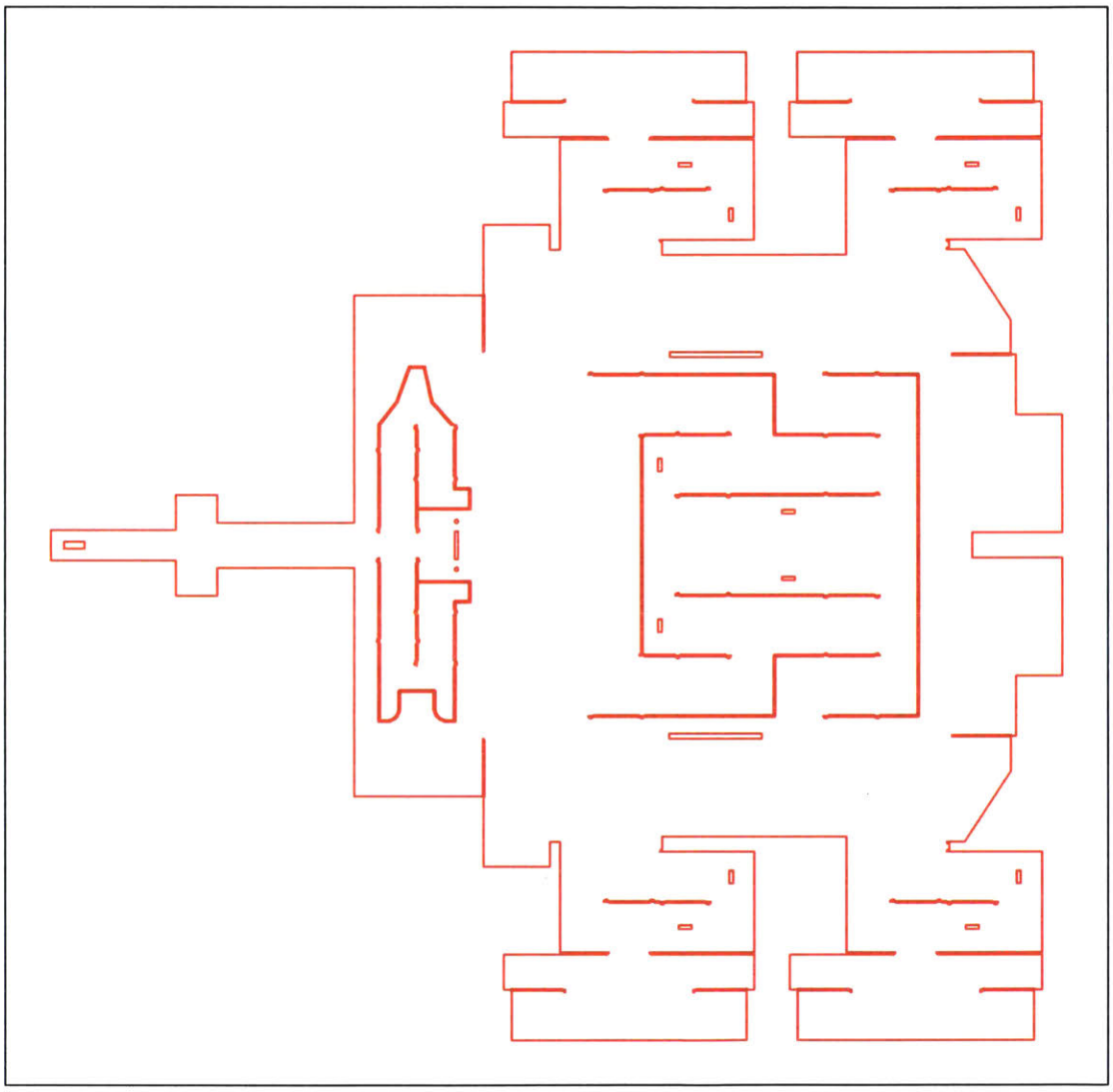


Figure A-6: Final Cut of Top Laminate of the Flying Monkey

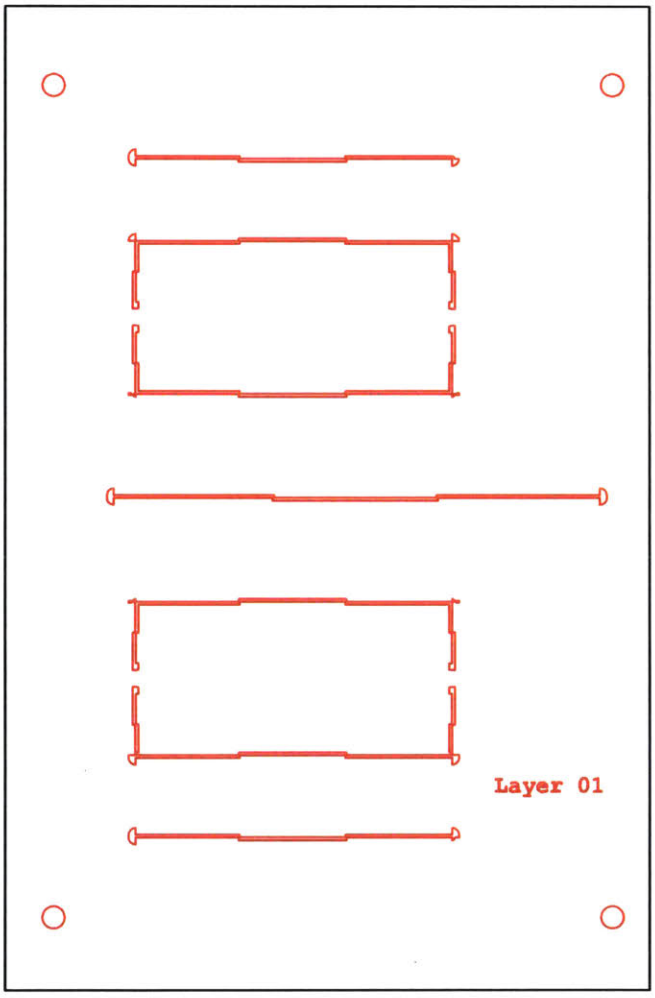


Figure A-7: Layer 7 of the Flying Monkey

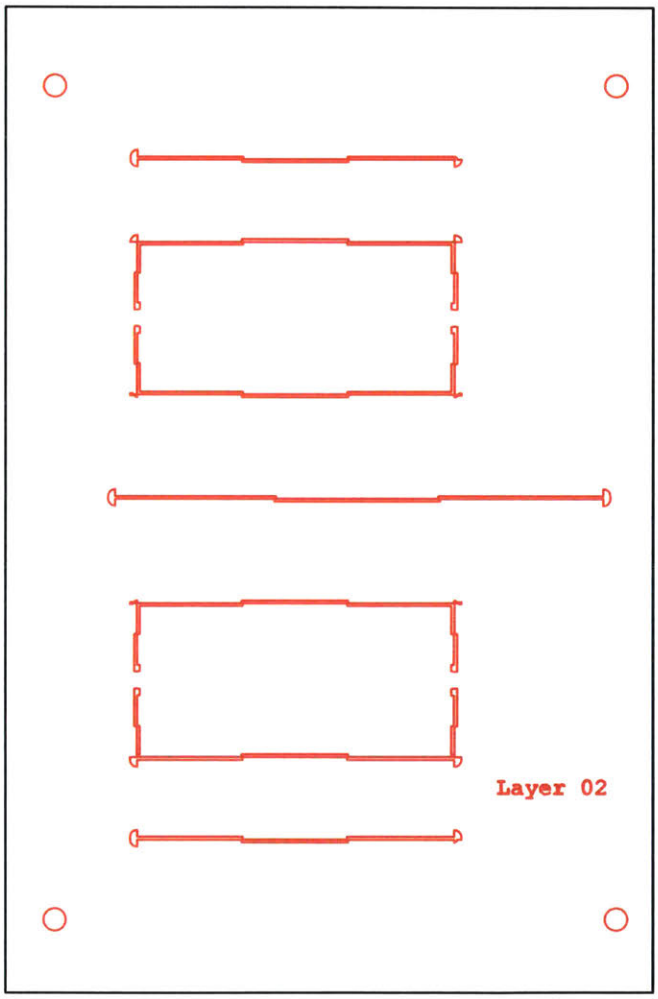


Figure A-8: Layer 8 of the Flying Monkey

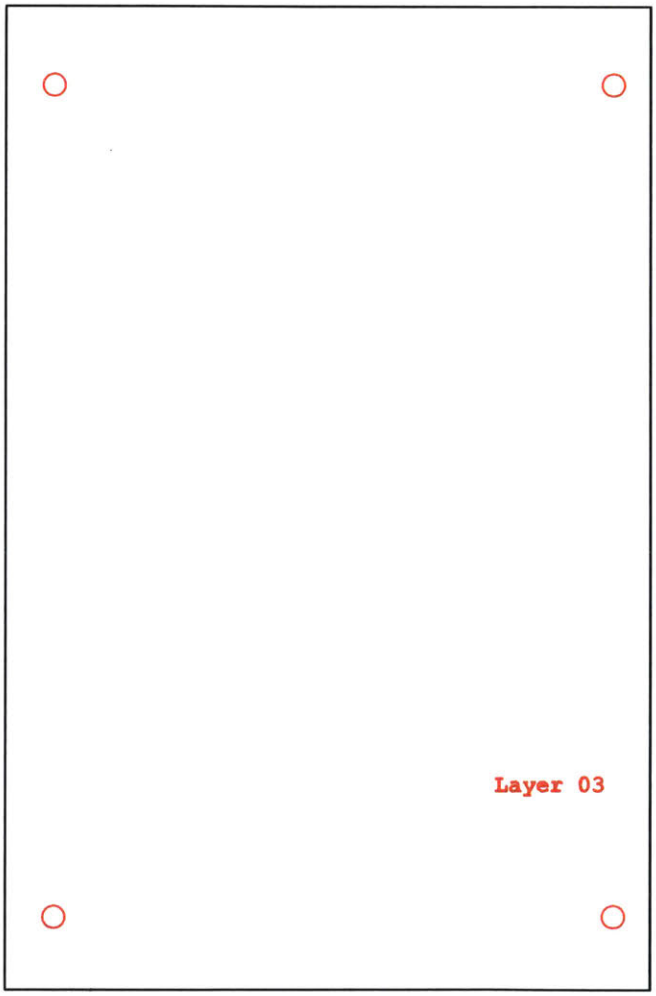


Figure A-9: Layer 9 of the Flying Monkey

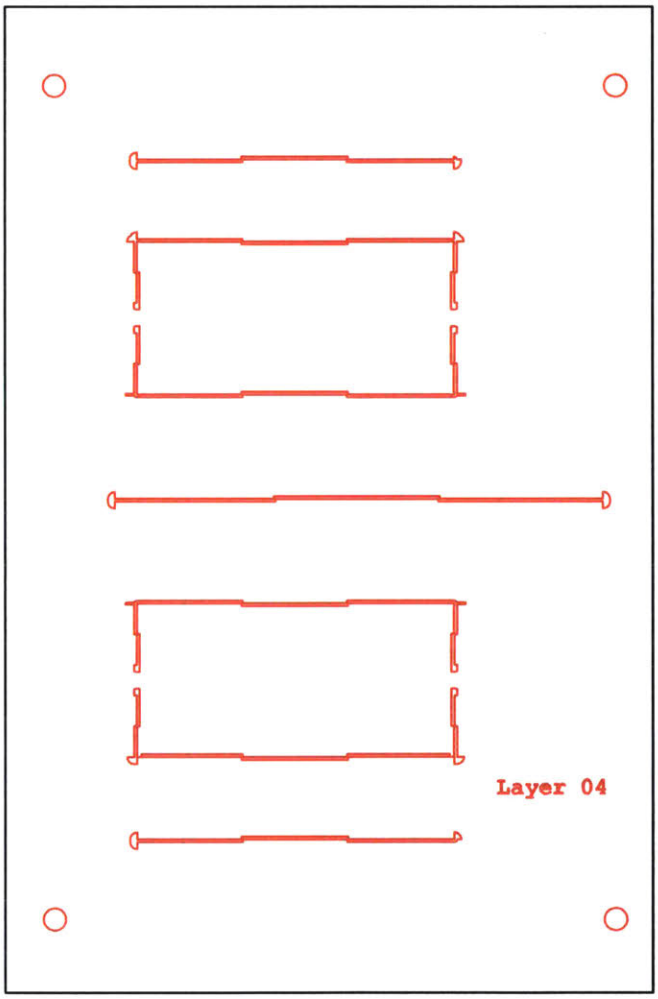


Figure A-10: Layer 10 of the Flying Monkey

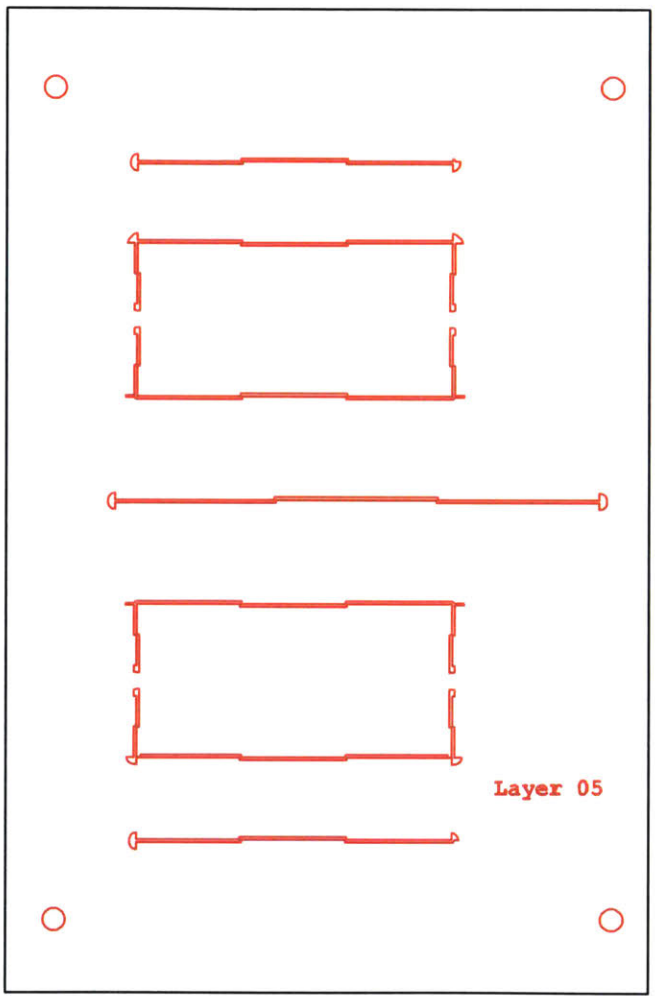


Figure A-11: Layer 11 of the Flying Monkey

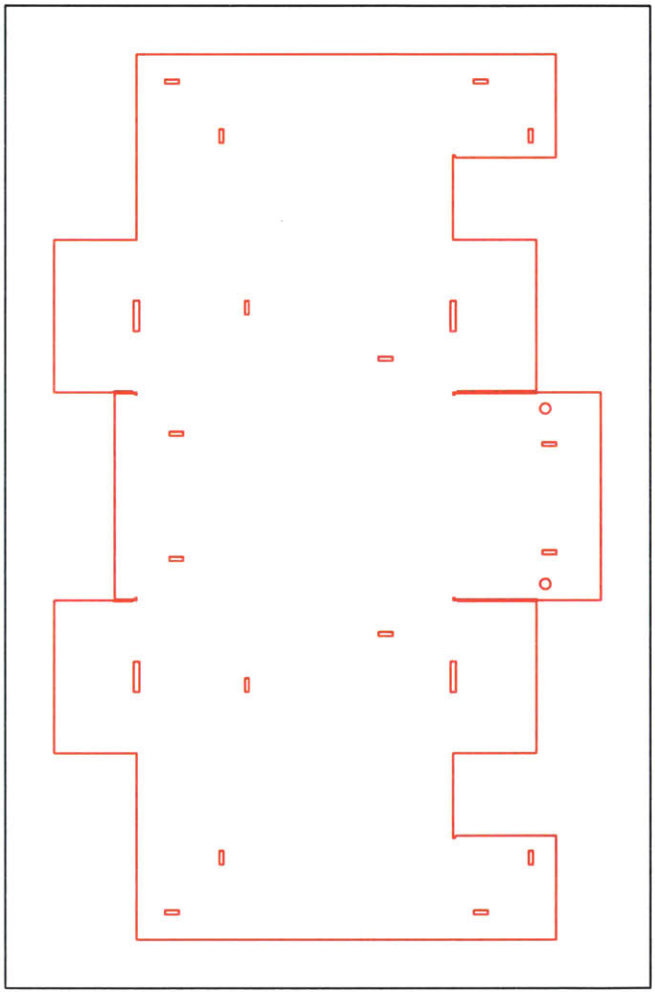


Figure A-12: Final Cut of Bottom Laminate of the Flying Monkey

Bibliography

[1] Vance A Tucker. Energetic cost of locomotion in animals. *Comparative Biochemistry and Physiology*, 34(4):841–846, 1970.

[2] Jared R Page and Paul EI Pounds. The quadroller: Modeling of a uav/ugv hybrid quadrotor. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4834–4841. IEEE, 2014.

[3] Maximilian Kriegleder, Raymond Oung, and Raffaello D'Andrea. Asynchronous implementation of a distributed average consensus algorithm. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1836–1841. IEEE, 2013.

[4] Daler Ludovic, Mintchev Stefano, Stefanini Cesare, and Floreano Dario. A bioinspired multi-modal flying and walking robot. *Bioinspiration & Biomimetics*, 10(1):016005, 2015.

[5] Richard J. Bachmann, Frank J. Boria, Ravi Vaidyanathan, Peter G. Ifju, and Roger D. Quinn. A biologically inspired micro-vehicle capable of aerial and terrestrial locomotion. *Mechanism and Machine Theory*, 44(3):513–526, 2009.

[6] A. Kalantari and M. Spenko. Modeling and performance assessment of the hytaq, a hybrid terrestrial/aerial quadrotor. *Robotics, IEEE Transactions on*, 30(5):1278–1285, 2014.

[7] SH Jeong and S Jung. A quad-rotor system for driving and flying missions by tilting mechanism of rotors: From design to control. *Mechatronics*, 24(8):1178–1188, 2014.

[8] Paul Birkmeyer, Kevin Peterson, and Ronald S Fearing. Dash: A dynamic 16g hexapedal robot. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2683–2689. IEEE, 2009.

[9] Daniel E Soltero, Brian J Julian, Cagdas D Onal, and Daniela Rus. A lightweight modular 12-dof print-and-fold hexapod. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1465–1471. IEEE, 2013.

[10] Michael Karpelson, Benjamin H Waters, Benjamin Goldberg, Brody Mahoney, Onur Ozcan, Andrew Baisch, Pierre-Marie Meyitang, Joshua R Smith, and Robert J Wood. A wirelessly powered, biologically inspired ambulatory microrobot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2384–2391. IEEE, 2014.

[11] C.D. Onal, M.T. Tolley, R.J. Wood, and D. Rus. Origami-inspired printed robots. *Mechatronics, IEEE/ASME Transactions on*, 20(5):2214–2221, Oct 2015.

[12] Cagdas D Onal, Robert J Wood, and Daniela Rus. An origami-inspired approach to worm robots. *IEEE/ASME Transactions on Mechatronics*, 18(2):430–438, 2013.

[13] Ankur M Mehta, Daniela Rus, Kartik Mohta, Yash Mulgaonkar, Matthew Piccoli, and Vijay Kumar. A scripted printable quadrotor: Rapid design and fabrication of a folded mav. In *Robotics Research*, pages 203–219. Springer, 2016.

[14] RJ Wood, S Avadhanula, R Sahai, E Steltz, and RS Fearing. Microrobot design using fiber reinforced composites. *Journal of Mechanical Design*, 130(5):052304, 2008.

[15] Robert J Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE transactions on robotics*, 24(2):341–347, 2008.

[16] Je-Sung Koh, Eunjin Yang, Gwang-Pil Jung, Sun-Pill Jung, Jae Hak Son, Sang-Im Lee, Piotr G Jablonski, Robert J Wood, Ho-Young Kim, and Kyu-Jin Cho. Jumping on water: Surface tension–dominated jumping of water striders and robotic insects. *Science*, 349(6247):517–521, 2015.

[17] Katie L Hoffman and Robert J Wood. Passive undulatory gaits enhance walking in a myriapod millirobot. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1479–1486. IEEE, 2011.

[18] Ankur Mehta, Joseph DelPreto, and Daniela Rus. Integrated codesign of printable robots. *Journal of Mechanisms and Robotics*, 7(2):021015, 2015.

[19] Adriana Schulz, Cynthia Sung, Andrew Spielberg, Wei Zhao, Yu Cheng, Ankur Mehta, Eitan Grinspun, Daniela Rus, and Wojciech Matusik. Interactive robogami: data-driven design for 3d print and fold robots with ground locomotion. In *SIGGRAPH 2015: Studio*, page 1. ACM, 2015.

[20] popupcad. http://www.popupcad.org/.

[21] TM Rao and Ronald C Arkin. 3d path planning for flying/crawling robots. In *SPIE Mobile Robots IV*, volume 1195, pages 88–96, 1989.

[22] John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[23] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[24] Michal Čáp, Jiří Vokřínek, and Alexander Kleiner. Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[25] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298. IEEE, May 2012.

[26] Daniel Martin Kornhauser, Gary L Miller, and Paul G Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master's thesis, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984.

[27] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(1-4):477–521, 1987.

[28] Michal Čáp, Peter Novák, Alexander Kleiner, and Martin Selecký. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 12(3):835–849, 2015.

[29] David Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.

[30] Jingjin Yu and Steven M LaValle. Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics. *arXiv preprint arXiv:1507.03290*, 2015.

[31] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[32] Daniel M Aukes, Benjamin Goldberg, Mark R Cutkosky, and Robert J Wood. An analytic framework for developing inherently-manufacturable pop-up laminate devices. *Smart Materials and Structures*, 23(9):094013, September 2014.

[33] Daniel M Aukes, Önur Ozcan, and Robert J Wood. Monolithic design and fabrication of a 2-dof bio-inspired leg transmission. In *Conference on Biomimetic and Biohybrid Systems*, pages 1–10. Springer, 2014.

[34] Yash Mulgaonkar, Brandon Araki, Je-sung Koh, Luis Guerrero-Bonilla, Daniel M Aukes, Anurag Makineni, Michael T Tolley, Daniela Rus, Robert J Wood, and Vijay Kumar. The flying monkey: A mesoscale robot that can run, fly, and grasp. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4672–4679. IEEE, 2016.

[35] Y. Mulgaonkar, G. Cross, and V. Kumar. Design of small, safe and robust quadrotor swarms. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2208–2215, May 2015.

[36] Pixhawk. http://www.pixhawk.org/.

[37] Bitcraze Wiki Expansion Decks. https://wiki.bitcraze.io/projects:crazyflie2:expansionboards:index.

[38] Benoit Landry. *Planning and control for quadrotor flight through cluttered environments.* PhD thesis, Massachusetts Institute of Technology, 2015.

[39] Taeyoung Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425, Dec 2010.

[40] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The grasp multiple micro-uav testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, 2010.

[41] Wolfgang Hoenig, Christina Milanes, Lisa Scaria, Thai Phan, Mark Bolas, and Nora Ayanian. Mixed reality for robotics. In *IEEE/RSJ Intl Conf. Intelligent Robots and Systems*, pages 5382 – 5387, Hamburg, Germany, Sept 2015.

[42] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5628–5635. IEEE, 2011.

[43] Robot Operating System (ROS). http://www.ros.org/.