Towards Cooperative SLAM for Low-Cost Biomimetic Robots

Austin Buchan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-142
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-142.html

August 10, 2017

Acknowledgement

Thanks to the Biomimetic Millisystems Lab, specifically Andrew Chen, Duncan Haldane, and Shiong Lun James Lam Yi for their contributions to the VelociRoACH and Zumy platform development.
The author would also like to thank Ben Morse and Dave Rollinson for their insights to practical Kalman filtering. Thanks to Sam Burden, Henrik Ohlsson, and Roy Dong for discussions on Hybrid Systems.
Many thanks to members of the Technical University of Hamburg-Harburg MuM Lab for a fantastisch research and cultural exhange experience.
Thanks to Miguel Heleno for guidance on what the ``dignity of the thesis'' truly means.
Thanks to Andrea Bajcsy for cooperatively exploring cooperative exploration.

Towards Cooperative SLAM for Low-Cost Biomimetic Robots

by

Austin D. Buchan

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ronald S. Fearing, Chair
Professor Ruzena Bajcsy
Professor Avideh Zakhor
Professor Mark Mueller

Summer 2017

Towards Cooperative SLAM for Low-Cost Biomimetic Robots

Abstract

Towards Cooperative SLAM for Low-Cost Biomimetic Robots

by

Austin D. Buchan

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ronald S. Fearing, Chair

Bio-inspired millirobots exhibit unique locomotion modalities that allow them to traverse complex, unstructured terrain that traditional robots cannot. This makes them potentially useful in urban search and rescue, structure inspection, environmental monitoring, and surveillance, all of which require some level of situational awareness. The low-cost, lightweight design and difficult-to-model dynamics of these robots also create unique challenges when applying state-of-the-art Simultaneous Localization and Mapping (SLAM) approaches usually used to gain this awareness. In this thesis, we develop a collection of estimation and control techniques that addresses these challenges, allowing teams of millirobots to localize within and map complex, unstructured environments. The analysis covers several facets of the problem, including low-cost millirobot team design, motion modeling, cooperative state estimation, and mapping.

We first show the utility of disposable low-cost robots in hazardous environments by using teams of picket and observer robots for exploration. Next, we explore a data-driven motion modeling approach to approximate the non-linear stochastic dynamics of legged millirobots. Then we develop an inter-robot pose estimation technique using monocular vision and active markers that can operate in visually feature-poor environments, and can scale to teams of computationally constrained robots. We demonstrate this technique first on autonomous underwater vehicles, and then extend it to a team of ground robots cooperatively navigating three-dimensional terrain. Finally, we explore a simple scanning laser technique that can leverage cooperating robots with cameras to map an environment.

Contents

List of Figures

List of Tables

Acknowledgments

Thanks to the Biomimetic Millisystems Lab, specifically Andrew Chen, Duncan Haldane, and Shiong Lun James Lam Yi for their contributions to the VelociRoACH and Zumy platform development. The author would also like to thank Ben Morse and Dave Rollinson for their insights to practical Kalman filtering. Thanks to Sam Burden, Henrik Ohlsson, and Roy Dong for discussions on Hybrid Systems. Many thanks to members of the Technical University of Hamburg-Harburg MuM Lab for a fantastisch research and cultural exhange experience. Thanks to Miguel Heleno for guidance on what the "dignity of the thesis" truly means. Thanks to Andrea Bajcsy for cooperatively exploring cooperative exploration.

Chapter 1

Introduction

1.1 Preface

> "Driven by the reality of experiments with actual robots our ideas took a route different from the traditional approach in Artificial Intelligence. Our approach emphasized
>
> (1) that there would be no traditional notion of planning
>
> (2) that no central representation was needed
>
> (3) that notions of world modeling are impractical and unnecessary
>
> (4) that biology and evolution were good models to follow in our quest
>
> (5) that we insist on building complete systems that existed in the real world so that we would not trick ourselves into skipping hard problems"
>
> -Rodney A. Brooks and Anita M. Flynn, 1989. [10]

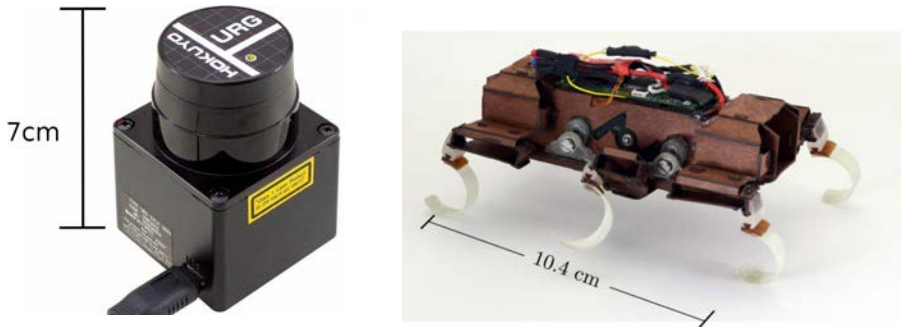
The work done in the Biomimetic Millisystems Laboratory has largely followed the guiding principles outlined by Brooks and Flynn when considering how to make robots "Fast, Cheap, and Out of Control." Especially relevant are notions (4) and (5), in that we build and test highly agile bio-inspired robots to prove that a full implementation is possible and robust to the non-idealities of construction. However, this thesis challenges the first three notions on the grounds that many motivating applications for low-cost robots working with and around humans in complex environments depend on representations of the environment to be effective. A large body of research on world modeling with robotic applications has progressed thanks in part to the availability of high-quality sensors, robotic platforms, and algorithms that mesh well with the assumptions imposed by the sensing paradigms. Implicit

in acquiring and using these high-performance sensors is a healthy research or engineering budget, which tends to work against making robots quickly or cheaply (though it is more likely they are in control). In order to bridge the gap between traditional sensing, control, and exploration approaches and the realities of low-cost robots, this thesis evaluates several of the assumptions underlying the approaches, and how they do not yet apply to our style of low-cost robots. By working towards implementations of motion models, pose estimation techniques, localization strategies, and mapping approaches with low-cost robots, we make the case that with appropriate consideration and co-design of robotic hardware and algorithms, the vision of ubiquitous low-cost teams of spatially aware robots may be a reality in the near future.

Simultaneous Localization and Mapping (SLAM) is a prominent field of study in robotics. It addresses the problem of creating a geometric representation of the environment (a map), and identifying where robotic agents are within that map (localization). In general, this problem can formulate the requirements for using one or more robotic agents to autonomously explore a space, generate map that can be used to reason about the space, and identify the location of the robots within that environment. This basic SLAM exploration problem needs to be solved first before other applications, such as team formation control, distributed sensing, and object or environment manipulation can be executed in initially unknown environments. Solving either the mapping or localization problem individually can be fairly straightforward given complete knowledge of the other, but building and maintaining an estimate of both of these quantities "simultaneously" has been the subject of much research.

For well-understood and modeled cases, the theory and implementation of SLAM is quite advanced. Thrun, Burgard, and Fox [72] outline a fairly complete approach to SLAM exploration problems using scanning range finders on wheeled robots in 2D planar environments. The fact that consumer technology can provide usable localization of smart phone users in GPS-denied environments based on Wi-Fi signal maps [39] is testament to the maturity of some domains of this problem. However, there are still many motivating problems that aren't solved in autonomous SLAM for exploration. These rich areas for research manifest when we try to extend SLAM approaches to robotic platforms, sensors, and environments that do not fit the assumptions of models used most frequently in SLAM due to the convenience and tractability of their formulation. In many cases, more expensive sensors, processing systems, and precision mobility platforms can be used to achieve implementations that more closely approximate modeling assumptions (uniform sampling time, Gaussian noise, rigid kinematics, abundant memory and processing time, et cetera).

The most popular type of sensor used for mobile SLAM implementations is the scanning laser range finder. These devices commonly provide a planar array of equiangularly-spaced range measurements using infrared (IR) laser Time-of-Flight (ToF) measuring technology. For full 3D environment mapping, a single sensor must either be moved by the robot through the desired spatial coverage, more sensors added at different angles, or sensors with larger spatial coverage. All of these approaches tend to increase the cost, size, weight, power consumption, and complexity of a SLAM solution. One of the most popular and economic planar range finders (shown in Figure 1.1a) is $1000 USD, 175cm^2, and 2.5W [33]. These



(a) Hokuyo scanning laser range finder.

(b) VelociRoACH Robot.

Figure 1.1: Rangefinder and millirobot scale.

specifications are quite far from anything that could be considered low-cost, especially if multiple scanners are needed.

In and of itself, the task of exploring low-cost implementations of sensing, control, and algorithmic robotics can be considered outside the domain of scientific research. However, seeking results with lower-cost hardware often elucidates the limits of state-of-the-art research approaches, and can inform where research effort may best be spent in improving sensing, actuation, and computational hardware. Much of the research of the UC Berkeley Biomimetic Millisystems Laboratory dwells in this realm of seeing how far the core of robotics research algorithms can be extended to novel systems that exhibit impressive and unique mobility. This work focuses on working towards a SLAM implementation that could be used with a team of small-scale, low-cost, bio-inspired robots. The running, flying, jumping, and climbing robots developed in the lab offer extreme mobility in complex unstructured environments. While highly mobile, the often erratic motion and environment interactions of these kinds of robots provides additional challenges in modeling and controlling their motion. These robots employ light-weight and low-cost construction methods such as cardboard and plastic laminate composites. The scale and construction methods also impose significant constraints on the size, weight, power, and computational capacity of the robots.

The unique mobility and small scale of bio-inspired millirobots anticipate applications that are currently infeasible with available robotic platforms. Planetary surveying and exploration, environment and structure inspection, surveillance, and environmental monitoring are just a few of the tasks that would benefit greatly from a robotic solution to free humans from the dull, dirty, and dangerous aspects of their execution. However, we do not currently have a general-purpose robotic platform that combines universal mobility with robust sensing that would allow them to function well in all of these environments.

Urban Search and Rescue (USAR) is an incredibly compelling, while incredibly challenging, use case for robots that can navigate in harsh unstructured terrains that may well damage or destroy the agents. Here, using many disposable low-cost robots rather than a few expensive robots is a much more attractive approach to dealing with the uncertainty of

the environment exploration task, and improves the likelihood of overall search success. In an environment that is likely to destroy or impair robots, being able to send a team of 100 cheap and redundant robots, losing half of them to the environment, but still completing the search task is better than only having 10 expensive robots and losing all of them. Critical to the USAR application is the ability to produce a map of an environment. Assessing structure condition, prioritizing exploration strategies, searching for and localizing evidence of survivors or hazards all depend on constructing an accurate map. As such, we seek to first develop an approach to mapping that can be scaled to the constraints of low-cost robots.

As outlined by Cadena, et al., [14], these challenges of fast stochastic dynamics, unstructured and feature-poor environments, and severely resource (size, power, weight, computation) constrained platforms are open problems in SLAM research. The modeling, sensing, and control techniques developed in this work move towards a viable SLAM approach that can work within the constraints of low-cost systems while being able to leverage their unique mobility. Through analysis of the motion capabilities of the robots, and available sensing technologies, our basic hypothesis is that a monocular vision-based approach can solve both the relative localization and mapping problems when using active markers on robots, and scanning lasers to illuminate specific points in the environment.

1.2 Contributions

This thesis primarily develops a monocular vision sensing modality that can enable localization of a team of low-cost robots, with outlined extensions for mapping. Aspects of motion modeling and cooperative exploration are delved into, but we find that the approaches in pose estimation and mapping largely compensate for the large amount of progress yet to be made in effective motion modeling for legged millirobots.

Chapter 2 explores an exploration application with a heterogeneous team of low-cost robots using a conventional pose estimation technique with passive reflective robots tags. This chapter serves primarily to motivate how a team of disposable and prioritized robots can be used to explore unpredictable environments that may have undetectable yet debilitating hazards. This technique sacrifices "picket" robots to discover the hazards, allowing more costly "observer" robots to avoid the hazards. It also shows the basic feasibility of monocular localization methods for inter-robot pose estimation, and the use of this data to produce a 2D map for motion planning.

Chapter 3 evaluates a novel system identification technique for modeling the complex dynamics of bio-inspired crawling robots. We show that data-driven Piecewise-Affine (PWA) model generation techniques can predict the dynamic behavior of the robot on short time scales (100ms). Ultimately this chapter shows there is still more work to be done before fine-grained models can be used to plan and control motion effectively. However, we show that approximating the legged robot as a differential-drive system allows the system to be controlled (albeit noisily) on a treadmill. This result motivates the exploration of pose

estimation techniques that can compensate for the stochastic dynamic behavior observed in the model identification experiments.

Chapter 4 extends a low-cost monocular relative pose estimation technique using active colored markers on a low-cost underwater robotic platform. An in-depth analysis of the physical properties of visual pose estimation shows there is a region of operation where active markers are more energy efficient than reflective markers in an underwater scenario. By extending the technique to use colored markers, we show how this improves the marker correspondence matching problem, improving scaling to larger teams of robots.

Chapter 5 shows the application of the pose estimation technique to a team of robots to maintain relative localization while navigating a 3D environment. The novel "inchworm" technique to cooperatively move a team of robots through an environment provides a group odometry estimate on par with encoder-based approaches where encoder approaches would fail due to 3D features and wheel or track slippage. This makes the technique ideal for application to crawling robots, as it is robust to noisy 3D motion.

Finally, Chapter 6 looks at how the techniques presented here could be extended to mapping areas of the environment with a scanning laser system, thus paving the way for a full SLAM system using only on-board monocular vision on low-cost bio-inspired millirobots. This technique positions the robots to cooperatively triangulate the position of a steerable on-board laser reflected in the environment in the intersecting fields of view of the robot cameras. We analyze the uncertainty in environment scan points resulting from this technique to function as sensor model, and use a simple motion model to evaluate the cost versus information gain for simple exploration strategies.

1.3 Background

1.3.1 Biomimetic Millirobots

Several unique and low-cost construction methods have been developed in order to achieve the wide variety of crawling, flying, climbing, and jumping robots produced and studied in the Biomimetic Millisystems Laboratory. The Scaled Composite Manufacturing (SCM) process developed by Hoover, et al. [36] enables several types of highly agile robots to be constructed out of mostly recyclable cardboard and PET. The intersection of high-maneuverability and low-cost robotics is motivated by applications such as Urban Search and Rescue (USAR), inspection, surveillance, and exploration tasks that require navigating complex, unstructured terrain, which may be so dangerous as to likely cause damage or destruction of robot agents sent through them. In these applications, low-cost robotics offer the advantage of being able to deploy many redundant agents for the cost of a single higher-cost robot. Low-cost robots may even be deliberately sacrificed to gain information about dangerous environments that aids the overall success of the task. Chapter 2 explores just such a task that leverages the disposable natrure of low-cost agile robots.

Low-cost robotics also offer challenges in that they have unpredictable dynamics, and limited budget for sensing and computational payload. Chapter 3 delves into quantifying this unpredictability for the purposes of dynamic modeling. Ultimately we see that further techniques need to be developed before they can be interfaced directly with robust SLAM frameworks, but some progress is made in identifying regions of dynamic interest, and useful differential-drive motion approximations.

Regarding the constrained sensing and computation budget, much of this thesis focuses on a minimal hardware and processing pose estimation and range finding technique. With a robust, cooperative pose estimation strategy, noisy dynamics can be compensated for by sensing the resulting pose from other robot team members. We focus on a monocular localization strategy using active markers that fundamentally requires very low cost, weight, and power sensing and emitting hardware in Chapters 4 and 5. Finally, Chapter 6 shows how a team of robots augmented with scanning laser beams can cooperatively triangulate surfaces in the environment, paving the way for a full mapping approach. While these techniques do not reach a full SLAM implementation, we show that progress has been made on the major challenges of motion modeling and perception for low-cost bio-inspired robots, and that with further engineering and research efforts cooperative SLAM with teams of robots is not out of reach.

Related to the low-cost paradigm of our research is the guiding principle of open-source and reproducible robotics research. The primary robotic platforms used in this thesis, namely the VelociRoACH, Zumy, and HippoCampus, are all open-source research platforms with documentation for their construction and software available freely online. In theory all of the research platforms presented herein can be reproduced for under $500 USD with Commercial, Off-the-Shelf (COTS) components or with commercially available rapid prototyping tools, and programmed using the software in repositories maintained by the Biomimetic Millisystems Laboratory (as referenced in appendices). Assessing the practicality of such lofty goals as open-sourced research hardware and software reproducibility is left as an exercise to the reader.

1.3.2 SLAM Sensors

Key to localization and mapping tasks are proprioceptive and extroreceptive sensors that can measure relevant robot and environmental signals, ultimately informing a probabilistic estimate of the likely underlying robot system and environment state. The types of sensors used in robotic SLAM implementations can vary greatly depending on type of robot, and thus resource constraints, as well as the environment and task goals. By first examining and categorizing the types of sensors commonly used in SLAM applications, we look at the broad landscape of sensors available with the hope of choosing a collection thereof that is most likely to provide the necessary perception while meeting our goals under the resource constraints of low-cost bio-inspired millirobots. Ideally we could quantify the trade-offs in information rate, range, and sensing certainty with respect to power, weight, size, cost for every commercially available and experimental sensor. We can imagine that there exists

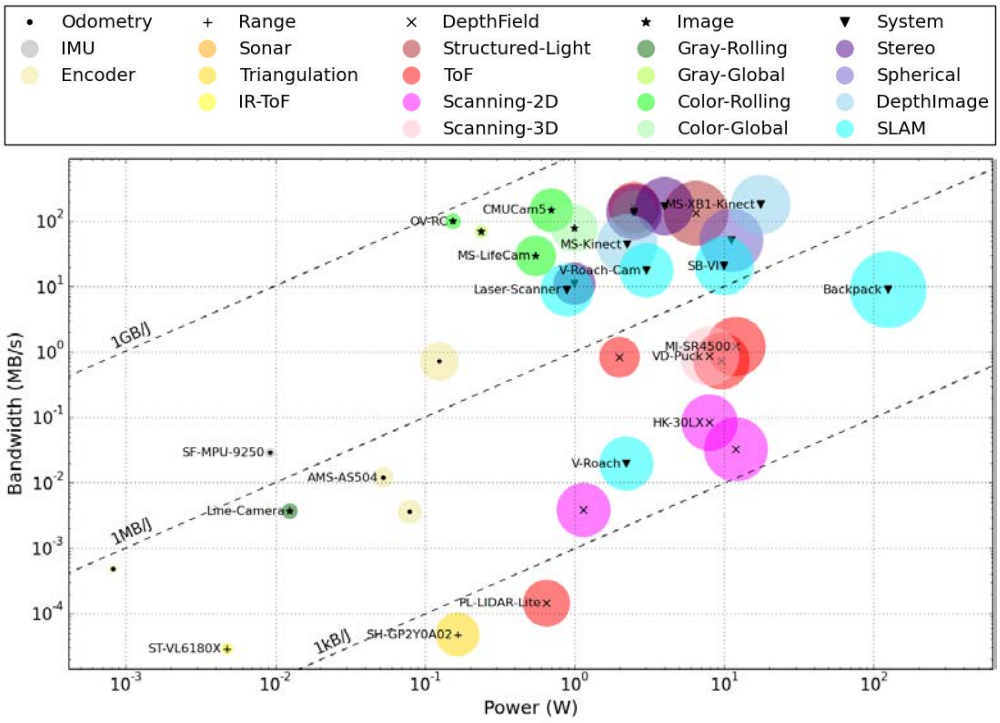


Figure 1.2: SLAM sensors and systems.

some Pareto frontier for information efficiency with respect to the exact task at hand for any number of metrics costs, such as bit/s, bit/J, bit/Kg, bit/m^3, or bit/\$. As a first step in this direction, we look at the raw bit rate produced, power consumed, and physical size of several types of common SLAM sensors, shown in Figure 1.2.

The sensors and systems we evaluate fall in five main categories: odometry, range, depth field, image, and system.

Odometry sensors measure quantities that ultimately inform an estimate of the path a robot has taken through an environment regardless of any environmental structure. Here we primarily consider Inertial Measurement Units (IMUs) and encoders. IMUs use inertial sensors to directly measure the linear acceleration and angular velocity of a robot body in 3D space. Encoders measure joint or wheel angles, and based on assumptions about the locomotive interaction with the environment can provide an estimate of the path that the robot has taken.

Range sensors are the simplest type of extroreceptive sensor in that they proved a single measurement of the distance from the robot to a surface in the environment. They all basically rely on the reflection of light or sound energy emitted from the sensor. Sonar and Infrared Time of Flight (IR-ToF) sensors measure the time it takes for an emitted packet of energy to be reflected from the environment and detected at the robot. Triangulation sensors also emit IR light, but measure the incident angle of reflected light to estimate the distance to a surface.

Depth field sensors produce a 2 or 3 dimensional measurement of depth in an area. This can either be done by scanning a 1D sensor through one or two axes, or actually measuring an entire field of emitted light (in the case of structured light sensors and ToF cameras).

Image sensors are the basic cameras we are familiar with; sensing a 1 or 2 dimensional array of light intensities, possibly in multiple wavelengths. We also distinguish whether these sensors can capture the entire array at a single instance (global shutter) or sequentially scan rows of the image (rolling shutter). These sensors usually require significant post-processing to produce an estimate of the geometry of the environment, but are interesting in their raw bit rate per Watt efficiency.

Finally, we consider some full SLAM systems that consist of multiple sensor types, actuators, and a computational system to fuses the sensor streams into a consistent estimate of pose and environment. Included in the graph are the VelociRoACH platform (V-Roach), the SLAM backpack system developed by Turner et al. [74], and the combination of a VelociRoACH and camera (V-Roach-Cam). As we will see through the developments in this thesis, the high informational bandwidth and low size, weight, power, cost of a camera make it an ideal way to augment the odometry sensors on the VelociRoACH to pursue a low-cost SLAM implementation.

Several sensors from these categories are plotted in Figure 1.2. The x-axis shows the active power they consume while producing data, and the y-axis shows the raw data bandwidth in MB/s they can produce. The point marker and color of disk respectively indicate the broad and specific class of sensors, and the area of the disk represents the physical size of the sensor or system. Isocontours of bandwidth/energy efficiency are shown as diagonal dotted lines of 1GB/Joule, 1MB/Joule, and 1kB/Joule. While there are several interesting trends in this data, the most prominent factor that motivates the rest of this work is that Odometry sensors and Image sensors are generally the smallest and most information-efficient sensors available. This trend can be seen in the prevalence of image and inertial SLAM work, and provides a launching point for the rest of this thesis. If a combination of IMU, encoder, and image measurements can be used to extract sufficient coverage of the proprioceptive space relevant to SLAM fusion tasks, it could prove to be a fruitful way forward for integrating SLAM techniques with small low-cost robots. The rest of this thesis can seen as evaluating the first steps of integrating sensing and estimation techniques in the directions of motion modeling, pose estimation, and environmental sensing under the constraints of minimal sensing. While there is still much to be done to reach a viable and robust SLAM framework at these resource scales, we hope to show that the challenges to bridging the gap between mature SLAM algorithms and agile robots is worthwhile and surmountable given enough consideration for the unique solution space we are evaluating.

Chapter 2

Safe Cooperative Exploration

2.1 Introduction

Cooperative millirobots are low-cost, low-energy systems that can be used to explore disaster areas during search and rescue operations. In this work, we address the problem of generating a terrain map with difficult to detect hazards that can incapacitate robot systems used during exploration. By exploring dangerous terrain with expendable *picket* robots, a safe path through the environment can be found for a more sensor-capable but less maneuverable *observer* robot. We evaluate three different picket robot exploration methods used to generate a map of safely traversable regions. The explored environment point cloud map is converted into a grid of safely navigable points that is suitable for path planning. We employ a conventional distance-based A* for path planning and navigation using the observer robot and demonstrate its ability to avoid hazardous terrain locations based on the final terrain map.[1]

Robots can reduce the risk to human workers during urban search and rescue operations by exploring hazardous environments. While a great deal of work has been presented on robotic terrain mapping, the issue of safe, fast, and inexpensive environment mapping and exploration is still an open research problem. Coordinated teams of bio-inspired millirobots are low-cost, low-energy systems that can be used to explore disaster areas during search and rescue operations. Biologically inspired millirobots can be inexpensive to produce, highly robust, and can exhibit remarkable dynamic performance [28]. Although they have limited computational capabilities and energy resources, millirobots can be sacrificed and easily replaced during exploration procedures. This allows for the design of exploration algorithms that maximize information gain by making riskier exploration decisions.

Considering these advantages, we present a computer vision-based approach to terrain mapping and navigation using a heterogeneous team of millirobots. Our goal is to address the

[1]This work was originally co-authored as part of the National Science Foundation Research Experiences for Undergraduates (NSF REU) project of A. Bajcsy, Summer 2015. A. Bajcsy developed and implemented the probabilistic exploration algorithm. A. Buchan designed and implemented the robotic systems, experiments, and algorithms for map generation and path planning.

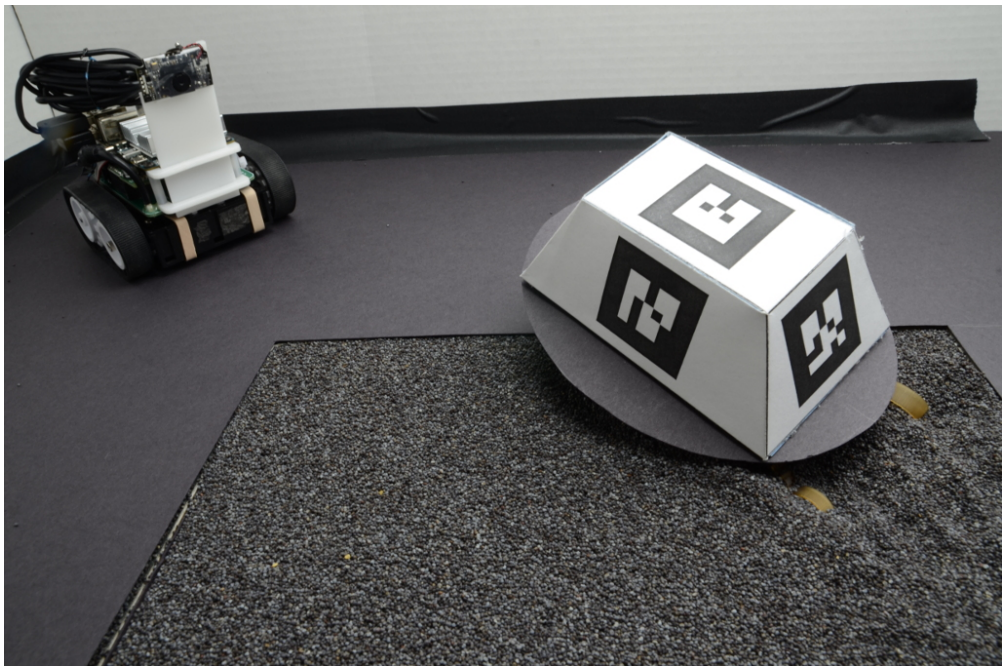


Figure 2.1: Observer Robot with Incapacitated Picket Robot

problem of generating a map of terrain with difficult to detect hazards that can incapacitate both robot systems used in the exploration. By exploring the terrain with expendable *picket* robots, a safe path through the environment can be found for a more sensor-capable but less maneuverable *observer* robot.

Since the *observer* robot is more expensive and computationally powerful than the *picket*, our design aims to ensure the safety of the observer robot as it moves through the unknown environment and accumulates the terrain map. This is achieved by sending the picket robot ahead of the observer robot to (1) investigate the reachable areas in the nearby environment and (2) to detect the presence of any unsafe terrain that may not be identified simply by visual inspection (for example, a quicksand trap). The picket robot has no sensing or vision capabilities and therefore depends on exploration goals and commands assigned by the observer robot. We limit the sensors used to generate the terrain map to a single camera carried atop the observer robot. Compared to other sensors, cameras are compact and energy efficient which meets the limited size and energy abilities of the millirobots. The camera is used to observe the picket robot's movements. Challenges addressed in this work include estimating depth with a monocular camera and safe path planning for the observer robot based on the generated terrain map.

2.1.1 Prior Work

Existing approaches to collaborative robotic terrain mapping have investigated both homogeneous swarms of millirobots and heterogeneous, hierarchical teams of robots. In order to map unknown environments using a homogeneous, stochastic swarm of millirobots, the authors of [19] introduce a classification approach to determine topology features of an environment from the interactions of the robot agents with the environment. The key to this approach is the emphasis on extracting a "sketch" of the environment rather than a detailed map. During the exploration, each robot performs a random walk through an arena with obstacles and the recovered point clouds exhibit features of the unique environment. Each of the millirobot agents are equipped with IR proximity sensors, camera and microphone, omni-directional wheels, and an IR beacon. We aim to avoid the addition of extra sensors on-board the millirobots in our work, since it increases both the cost and the risk associated with damaging the robots during exploration.

Alternatively, heterogeneous robotic teams may consist of millirobots, medium-sized tank robots, and large all-terrain vehicles [25]. Since each robot is equipped with different capabilities, they collaboratively accomplish mapping and exploration tasks based on their unique configurations. A team leader robot synthesizes the millirobot data by using occupancy grid mapping algorithms to combine multiple streams of sensor data [25] [53]. However, these centimeter-scale millirobots are also configured with infrared (IR) sensors, a camera, and computation modules and use sonar distance measurements, GPS, and landmarks for localization. Similarly to the work of [19], these sensor additions increase the cost and exploration risk.

Compared to large teams of robots, cooperative observer and picket robot pairs have proved to be successful at safe terrain navigation. In [29] [52], a large, sensing *main robot* drives a small, limited-sensing *picket robot*. Our experiment follows a similar design but we use limited-sensing millirobots for both the *main robot* and the *picket robot*. Safety, as defined in [32][29] can be addressed as a coverage problem where the *picket* robots have to cover a certain size of area in front of the *main* robot in order to identify all dangerous terrain. Complete coverage is not necessary for ensuring the safety of the *main* robot; only the dangerous areas need to be explored and detected. In the work of [29], the main robot performs vision-based marker detection of an Augmented Reality (AR) Tag placed on the back of the picket robot in order to assess the slipperiness of the terrain ahead. This framework proved to be a feasible proof of concept for remote terrain detection, with over 90% accuracy when identifying slippage of the picket robot [29]. In contrast to terrain classification, we aim to build a terrain map that allows for the identification of reachable and safe regions in the environment.

In addition to the challenges of robotic collaboration, the autonomous navigation of millirobots through hazardous terrain is complicated due to the limited sensing capabilities of many millirobots. In [48], the authors use a single low-resolution camera in order to perform visual landmark-based localization for a crawling millirobot within a dark, cluttered, and confined environment. The path planning was based on Field D* algorithm while detection

and recovery from "stuck" conditions used optical flow algorithms on successive camera frames. The framework presented in [48] was able to have a millirobot autonomously find its way to within ~ 1 centimeter of a goal target.

2.2 Methods

2.2.1 Robotic System

Our robot team consists of two different types of robots: the VelociRoACH as the *picket robot* and the Zumy as the *observer robot*. Both robots can be run wirelessly on battery power, but for the purposes of this experiment are tethered. The VelociRoACH is the latest in several low-cost millirobots developed in the Biomimetic Millisystems Laboratory using the Smart Composite Microstructures process [36]. Due to their small size, it is important to minimize the number of actuators and the required actuator bandwidth on bio-inspired millirobots. Therefore, these robots [28][35][5] have been designed to be open-loop stable, allowing them to convert feed-forward motor power into stable and robust locomotion. For this experiment, the on-board microcontroller systems uses only the motor back-EMF measurement to control the average velocity of the left and right pair of legs. Based on this minimal control, and recent commercially available versions of these robots[2], we believe these robots could be manufactured for less than $30 U.S..

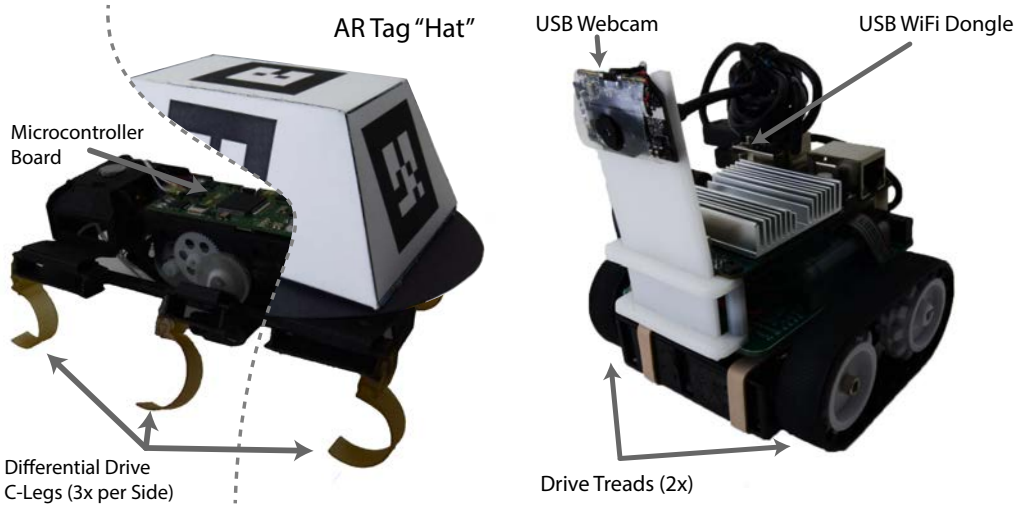
The Zumy robotic platform is a millibot-scale tracked robot designed to be able to support a full Linux computing system for convenient software development, networking, and accelerated vision processing. This robot can carry a Microsoft Lifecam 3000 camera for visual tracking, and supports Wi-Fi wireless communication to a host computer. While not bio-inspired, this robot was designed to be easily built from commercially available off-the-shelf (COTS) parts while being about the same size scale as the VelociRoACH. References to full hardware and software specifications can be found in Appendices B.2 and C.3. This makes the Zumy robot ~ 10 times the cost ($300 USD) of the VelociRoACH.

2.2.2 Experimental Setup

Our test environment (shown in Figure 2.3a) consists of a 60cm by 75cm arena bounded on two out of the four sides with white cardboard walls. The ground terrain is planar with "safe" cardboard regions and a "hazardous" trap in the center of the arena. The granular media (poppy seeds) was chosen since it incapacitates both types of robots when entered, and is difficult to distinguish with visual or range sensing methods.

The picket robot is outfitted with an AR Tag "hat" so that it can be visually tracked by the observer robot. The AR Tag hat is shaped as a rectangular frustum with a unique AR Tag on each of the five visible faces. This guarantees that from any position and orientation

[2] `http://www.dashrobotics.com`

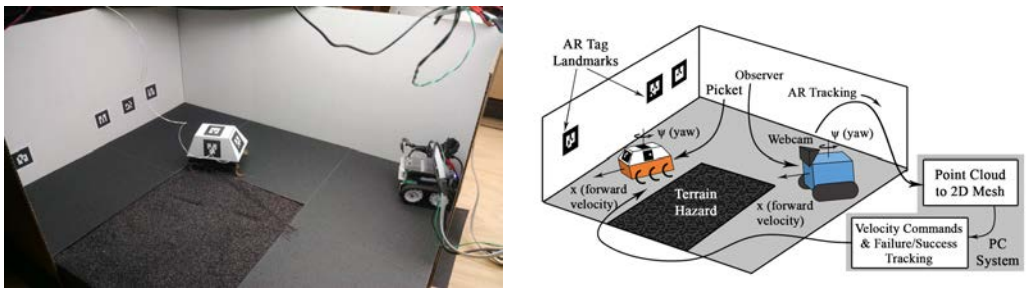


(a) Picket: VelociRoACH (b) Observer: Zumy

Figure 2.2: Robot Team

		VelociRoACH	Zumy
Size	cm	14x10x9	10x10x12
Weight	g	45	325
Maximum Velocity	m/s	2.7	1
CPU		dsPIC33	4x Cortex-A9
CPU Frequency	MHz	40	1700
Compute and Sense Power	W	2.7	7.5
Battery Voltage	V	3.7	7.2
Battery Capacity	mAh	120	850

Table 2.1: Robot Specifications



(a) Environment (b) System Organization

Figure 2.3: Experimental setup

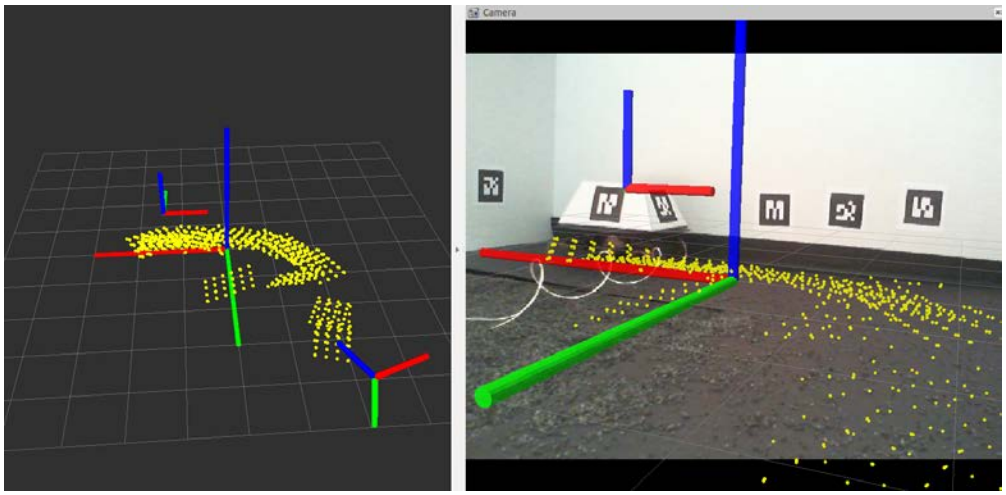


Figure 2.4: Point Cloud Collection

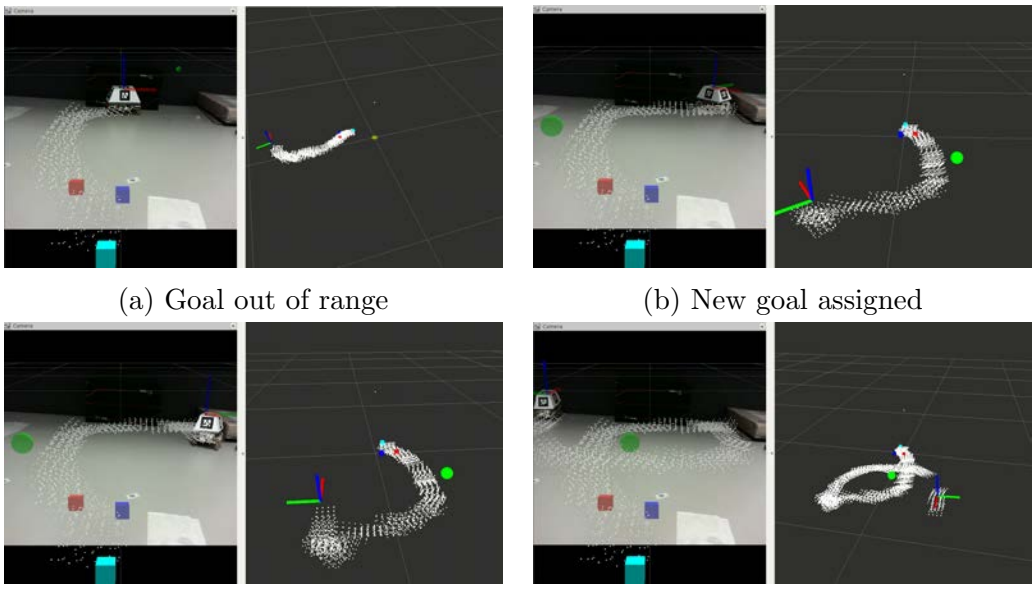
we can see at least one face of the hat with an AR Tag. AR Tags were also added to environment walls to allow registration of terrain map and for navigation.

Figure 2.3b details the communication and computational system. During a field deployment, the two robots can be run untethered, but still coordinated by a host PC system. The observer robot communicates with the host over WiFi, while the picket robot uses an IEEE 802.15.4 radio to receive velocity commands from the host. We use the open-source Robot Operating System (ROS) [62] to manage software and coordinate on-line information sharing between the various parts of this system.

We use the ALVAR software library[3] to track the pose of the AR markers relative to the observer robot's camera. In addition, we developed calibration software which allows us to estimate the position and orientation of the top-most "master" marker (whose orientation matches the VelociRoACH's body) based on any visible AR markers even if the master marker is not currently in sight. The AR Tag tracking from the camera video feed is processed on-board the Zumy, which would greatly reduce the bandwidth requirements for running multiple observer robots coordinated by a single host system.

During operation, the host system receives the pose information of the picket robot via the observer robot. The host uses this information to construct a point cloud representing the explored environment surface. Based on this information, the host then sends velocity commands to the picket robot guiding it to unexplored regions. The host also monitors the progress of the picket robot to determine if it has failed to reach a goal, or become trapped by the environment. Figure 2.4 shows this view from the observer robot's camera, and a perspective view showing the positions of the picket robot and camera relative to the environment frame.

[3] `https://github.com/sniekum/ar_track_alvar`



(a) Goal out of range (b) New goal assigned

(c) Approaching second goal (d) Final map

Figure 2.5: Probabilistic Exploration Sequence

2.2.3 Environment Exploration

We investigate three picket robot exploration methods: random walk, probabilistic random walk, and manual drive. These three methods represent three points on a spectrum of exploration techniques, ranging from automated and non-deterministic to human-operated. Note that these approaches were chosen for simplicity and generality, but more complex explorations can be used when a specific navigation goal is in mind (see [75]).

During random walk, the picket robot alternates between four phases of right, forward, left, backward at constant velocities and spends a random amount of time in each phase. This is to try to mitigate the problem of the picket robot getting stuck in front of a wall for an extended period of time. If the picket robot gets stuck at an obstacle or in a trap, we place it back at the start location near the observer robot. For the purposes of this experiment, this action symbolizes a new picket robot being deployed after the last one failed. Additionally, we record the location of where the picket robot was stuck in order to mark the final mesh and occupancy grid map with a flag for dangerous terrain.

Probabilistic random walk deals with one of the main assumptions in our purely vision-based approach: that the observer robot can see the picket robot at all times. In this method, we bound the region of exploration for the VelociRoACH to the camera FOV. We use a homography matrix to convert between real-world coordinates to pixels in the camera frame. The homography matrix is constructed from the real-world measurements of the VelociRoACH's four corners and the corresponding pixel coordinates in the camera image. Once we have defined the exploration plane, we can randomly assign goal locations to the picket robot that are always guaranteed to keep the robot in the line of sight. Over

time, if there exists an obstacle in the environment and the VelociRoACH cannot reach the obstructed area from any angle, then the probability of assigning that location as a goal will approach zero. However, since we do not immediately assign a probability of zero after a single failure, we allow for additional exploration to the same region in an effort to verify that the picket cannot get past the obstacle from another direction or angle. The VelociRoACH runs for some fixed time T while generating a map, after which it stops and allows for Zumy path planning and navigation. Our exploration and goal-assignment algorithm is as detailed in Algorithm 2.1.

Algorithm 2.1 Probabilistic Exploration Algorithm

Require: $goalGrid_{ij} \leftarrow 1/(N \times M); 0 \leq i < N, 0 \leq j < M$
  $goalPoint_{xyz} \leftarrow assignGoal$
  while $stillExploring(time, map)$ **do**
    if $dist(picket, goalPoint_{xyz}) < \epsilon$ **then**
      $goalPoint_{xyz} \leftarrow assignGoal$
    else
      if $dist(picket, goalPoint_{xyz}) \geq \epsilon \; after \; T \; tries$ **then**
        $goalGrid \leftarrow updateGoalGrid_{ij}$
        $goalPoint_{xyz} \leftarrow assignGoal$
      end if
    end if
  end while

Figure 2.5a shows an example where the assigned goal location, marked by the green sphere, is impossible for the robot to reach. After failure, the probability of reassigning the location is decreased and the goal is is randomly reassigned (see Figure 2.5b). The picket robot proceeds to turn towards the new goal and in Figure 2.5d it reaches the destination and produces a point cloud of the path it took to reach the goal location.

Finally, in manual drive, a human operator moves the picket robot through the experimental arena, intentionally running the robot into traps or obstacles in order to mark the dangerous regions in the final terrain map.

We expect other automated methods will exhibit performance between random walk and manual drive (potentially even superseding manual drive). However, future work will need to focus on modelling the motion and sensing of the picket-observer robot system in order to be effective at coordinating teams of robots during exploration.

2.2.4 Map Generation

Our raw terrain map consists of a point cloud created by estimating the pose of the top-most AR Tag (and thereby of the picket robot). We use the open-source Point Cloud Library (PCL)[4] for storing the point cloud map as an unordered list of (x,y,z) coordinates. Since

[4]http://pointclouds.org

Algorithm 2.2 Assign Goal

Require: $H \leftarrow homographyMatrix$
  $pdf \leftarrow pdf(goalGrid)$
  $randNum \leftarrow rand(0.0, 1.0)$
  while $k < N * M$ **do**
    if $k == 0$ **and** $randNum \leq pdf[k]$ **then**
      $gridCell \leftarrow k + 1$
      break
    else
      if $pdf[k-1] < randNum$ **and** $randNum \leq pdf[k]$ **then**
        $gridCell \leftarrow k + 1$
        break
      end if
    else
      $gridCell \leftarrow k + 1$
      break
    end if
  end while
  $goalPixel_{rc} \leftarrow getPixel(gridCell)$
  $goalPoint_{xyz} \leftarrow H^{-1} * goalPixel_{rc}$

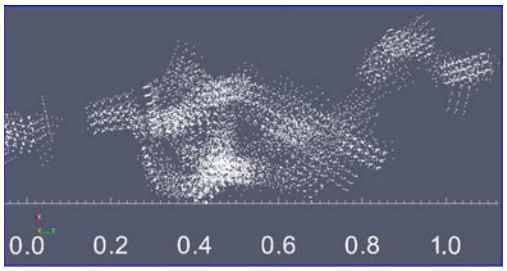
we know beforehand the precise measurements of the AR Tag hat, we can estimate the distance the robot is away from the camera as well as its orientation. The four corners of the uppermost AR Tag can be projected to match the size and orientation of the VelociRoACH at any given time, allowing us to place a rectangular point cloud under its body. During exploration, we mark points when the picket robot has gotten stuck or trapped. Thus, the final point cloud represents all reachable area in the observer robots FOV.

After collecting the point cloud, we convert it into a 2D mesh representing navigable terrain using PCL and OpenSCAD[5]. In order to reduce the complexity of the 2D mesh, after the first-pass meshing, interior points are removed from the data structure and the filtered cloud is re-meshed. The simplified mesh can be better converted into an occupancy grid to allow for path planning using the observer robot.

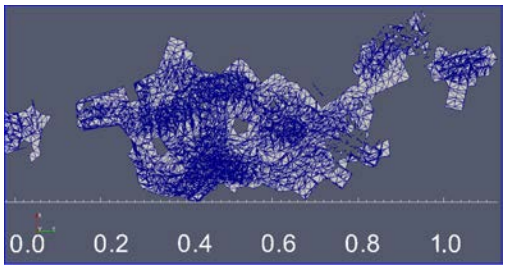
2.2.5 Path Planning

Once the explored environment has been converted into a grid of safely navigable points, it is suitable for path planning. In exploration regions where there is discovered dangerous terrain, grid points within a certain radius can simply be removed from the grid (see Results section for diagram). For the results below, we employed conventional distance-based A* [30], but any other path planning and navigation algorithm can be used.

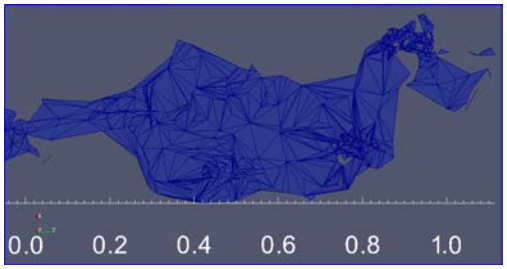
[5]http://www.openscad.org



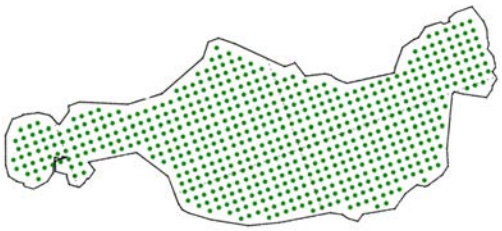
(a) Exploration Point Cloud (scale in meters)



(b) Meshed Point Cloud (scale in meters)



(c) Simplified Mesh (scale in meters)



(d) Navigation Grid

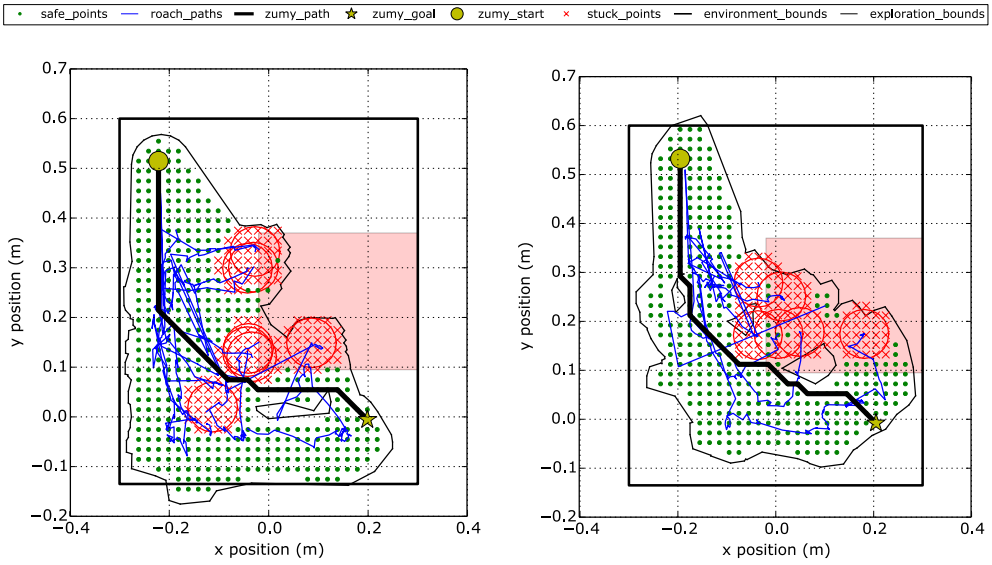
Figure 2.6: Map Generation

2.3 Results

Figures 2.7a and 2.7b show the results of two techniques for generating a map from our exploration method. In each, the bounds of the exploration environment are the outermost black outline, with the sticking hazard shown as a red rectangle. The paths that the picket robots took while exploring the space are shown in thin blue. The resulting explored region is outlined in thin black. This region has a navigation grid superimposed in green dots and red "X"s. The red dots represent grid points that are within a specified radius of a location where the explorer robot got stuck (this radius is shown in as a red circle). The green dots are the remaining navigation points that are safe to use for path planning. Finally, the yellow circle and gold star represent the start and goal location of the observer robot, and the thick black line represents the path that has been planned through the safe region.

The random trial took 326 seconds while becoming stuck 9 times, while the manually driven trial took 223 second while becoming stuck 6 times.

Noise from AR tag reads is evident in both trials. Essentially, when an AR tag position is misread, it places a point cloud collection in an area outside the environment, or in a location that is clearly in the hazard. Future work will need to address this issue by filtering the poses read for the explorer robots. This can involve modifying the AR tracking software to require a more confident read of the tag location, combined with a motion model and Kalman filter for the explorer robot motion. For the purposes of clarity in this work, poses that were clearly outside the exploration environment were manually removed from the tracking dataset.



(a) Random-drive exploration (b) Manual-drive exploration

Figure 2.7: Mapping Results

Figure 2.7a also shows the result of inappropriately labeling a point as being stuck in the bottom left corner of the hazard region. This could be problematic for path planning if several incorrect stuck poses are labeled resulting in the start and goal locations being disconnected in the planning grid. A method for mitigating this effect would use a probabilistic occupancy grid assignment, allowing subsequent exploration to determine that incorrectly labeled stuck regions are actually free.

2.4 Conclusion

We have shown a technique for exploring a hazardous environment with picket robots to generate a map of safely navigable regions for high-value robotic systems. This technique relies on having expendable low-cost picket robots to traverse dangerous terrain, and discover hazards by becoming incapacitated by them. As these robots are marked with AR tags, an observer robot monitoring the exploration can generate a map of safely traversable regions, and plan a path to a desired goal.

2.4.1 Future Work

A key element for making this work practical is the overall automation of the exploration technique. The stochastic nature of the VelociRoACH movement made this difficult to implement as of yet, but future work will focus on better sensing and motion modeling of the picket robots to effectively guide them autonomously through an environment. Chapter 3

looks deeply at the nature of this stochasticity, and makes some progress in quantifying and predicting the dynamic behavior of the VelociRoACH robot. While still not at the level of prediction necessary for most SLAM implementations, simplifying assumptions about differential drive dynamic models are discussed as a substitute until more rigorous models can be explored. Once this is accomplished, further extensions can be made to generate maps of 3D environments, as well as environments that do not have incapacitating hazards, but regions that are of varying difficulties to traverse. Generating a map of expected cost of transport using picket robots would be highly valuable for terrains that are difficult to assess with vision or range-finding techniques alone.

Chapter 3

Motion Model Identification

3.1 Introduction

This chapter presents a simple, data-driven technique for identifying models for the dynamics of legged robots. Piecewise Affine (PWA) models are used to approximate the observed nonlinear system dynamics of a hexapedal millirobot. The high dimension of the state space (16) and very large number of state observations (∼100,000) motivated the use of statistical clustering methods to automatically choose the submodel regions. Comparisons of models with 1 to 50 PWA regions are analyzed with respect to state derivative prediction and forward simulation accuracy. Derivative prediction accuracy was shown to reduce average in-axis absolute error by up to 52% compared to a null estimator. Simulation results show tracking of state trajectories over one stride length, and the degradation of simulation prediction is analyzed across model complexity and time horizon. We describe metrics for comparing the performance of different model complexities across one-step and simulation predictions.[1]

Biologically inspired millirobots are inexpensive to produce, highly robust, and can exhibit remarkable dynamic performance [28]. Due to their small size, it is important to minimize the number of actuators and the required actuator bandwidth. Therefore, these robots [5][35][28] have been designed to be open-loop stable, allowing them to convert feedforward motor power into stable and robust locomotion. The ability of these robots to run dynamically has allowed us to observe a number of emergent dynamic maneuvers such as rapid turns, reversals, jumps exceeding body height, flips, and cartwheels. However, we currently have no modeling approaches that are accurate enough to predict these aggressive maneuvers on the time scales significant to our robotic systems. The work presented here

[1]This chapter originally published as "Automatic Identification of Dynamic Piecewise Affine Models for a Running Robot," IROS 2013 [11]. A. Buchan developed the model framework, regression methods, statistical analysis, prediction methods, and experimental data collection setup. D. Haldane developed the VelociRoACH platform, treadmill hardware and control, control strategy assumptions and implementation, and data filtering pipeline. It was supported by the National Science Foundation under IGERT Grant No. DGE-0903711, and Grant No. CNS-0931463, and the United States Army Research Laboratory under the Micro Autonomous Science and Technology Collaborative Technology Alliance.

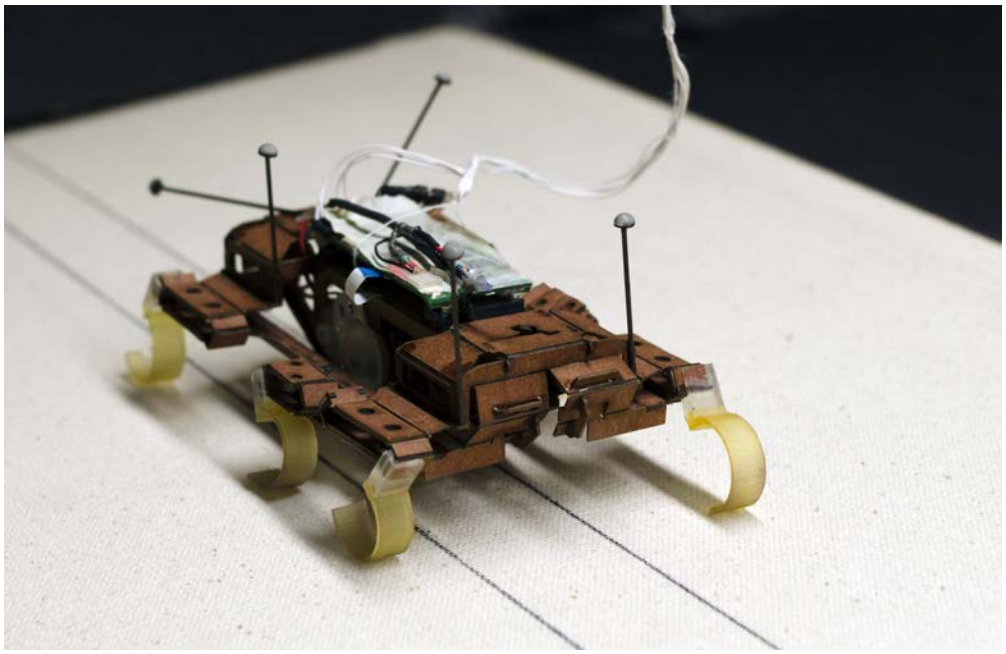


Figure 3.1: The VelociRoACH on a treadmill, equipped with tether.

is taking the first steps towards modeling techniques that will enable predictive control to execute these actions on command.

Legged locomotion has been modeled using a diverse array of reduced order templates [34]. By applying traditional analyses to these analytic models, predictions of gait characteristics such as speed or stability can be produced. System identification for legged systems has been based on fitting parameters to some of these analytic models. Several bodies of work [2][43][44] fit Spring Loaded Inverted Pendulum (SLIP) model parameters to the stance phase of the RHex family of robots. Bloesch et al. [6] developed a method to fit kinematic parameters and sensor models to the StarlETH, a legged robot. Recently, a parameter fitting approach has been extended to hybrid dynamics by reducing the dimensionality of the system around a periodic orbit [13].

Another technique has been to approximate the rhythmic dynamics of a legged system with a data-driven Floquet model [64]. This work was able to find periodic orbits in a low-dimensional state space to which high frequency disturbances converge. The Floquet models and parameter-fit analytic models can make good predictions for steady-state behavior, but they rely on an assumption of periodic dynamics. They cannot describe transient maneuvers observed when periodic gaits are not enforced. We believe that understanding the full space of behaviors with non-periodic leg motion will be necessary to make headway on useful predictive models for under-actuated yet highly maneuverable legged robots.

To this end, we present a data-driven statistical approach for general modeling of the dynamical behavior of nonlinear systems. Our approach has the benefits of requiring no knowledge of the underlying dynamics, and using computationally simple models to describe

the behaviors observed. This identification approach scales well with the dimension of the dynamical state space and number of observations. It is also easy to tune the granularity of the model approximation to make the best use of available computational resources for on-line predictions. The automatic identification of state-space partitioning based on the statistics of the observations can give initial insights into the behavior of a complex dynamical system, which is useful for later analysis.

We assume the dynamics of our millirobots are locally governed by linear dynamics (rigid body motion, kinetic friction, damped springs) with highly nonlinear shifts in these dynamics (footfalls, four-bar linkage singularities). As such, Piecewise Affine (PWA) systems are a reasonable first approximation of these dynamics. PWA models describe a system as a collection of affine submodels, and a partition of the state-space where each model is applied. These models are very fast to compute for forward prediction, a desirable property for state estimation on low-power systems. Concepts of stability, controllability, and observability have been extended to PWA systems as approximations of nonlinear systems [69], which will enable further analysis with the models generated by this method. In addition, Tedrake et al. [71] have shown that LQR controllers can piece together these regions of state-space for control.

Automatic identification of PWA models is actively being researched [22]. In general, this iterative identification process grows in exponential time with the number of samples, and is thus intractable to compute on very large datasets. Heuristic methods can avoid incurring the computational cost, associated brute-force techniques, but are sensitive to initial region partitioning and can converge to non-global optima. As such, we chose simple, statistically-driven clustering methods to identify the partition.

We describe current methods for empirically deriving models of dynamical systems. Section 3.2 gives the methods for collecting data on our dynamic millirobots and fusing sensor data into a high-fidelity state trajectory. In Section 3.2.5 we explain the type of models derived by our method, and the way in which the collected data is partitioned to learn these models. Finally, in Section 3.3 we show how the collection of models learned with our method was able to predict future states through simulation of the identified dynamics and control.

3.2 Methods

This work presents a processing pipeline that identifies several models for a dynamic robotic system, as well as methods for comparing the performance of those models. First, sensor data are collected and fused into a high-fidelity state trajectory (Sections 3.2.1 to 3.2.4). Models are then fit to this data, as described in Section 3.2.5. Section 3.2.6 describes how the models were forward simulated, and Section 3.2.7 gives our validation method for the model accuracy and simulation. Figure 3.4 illustrates this process, and will be referred to in the description of each step of the pipeline.

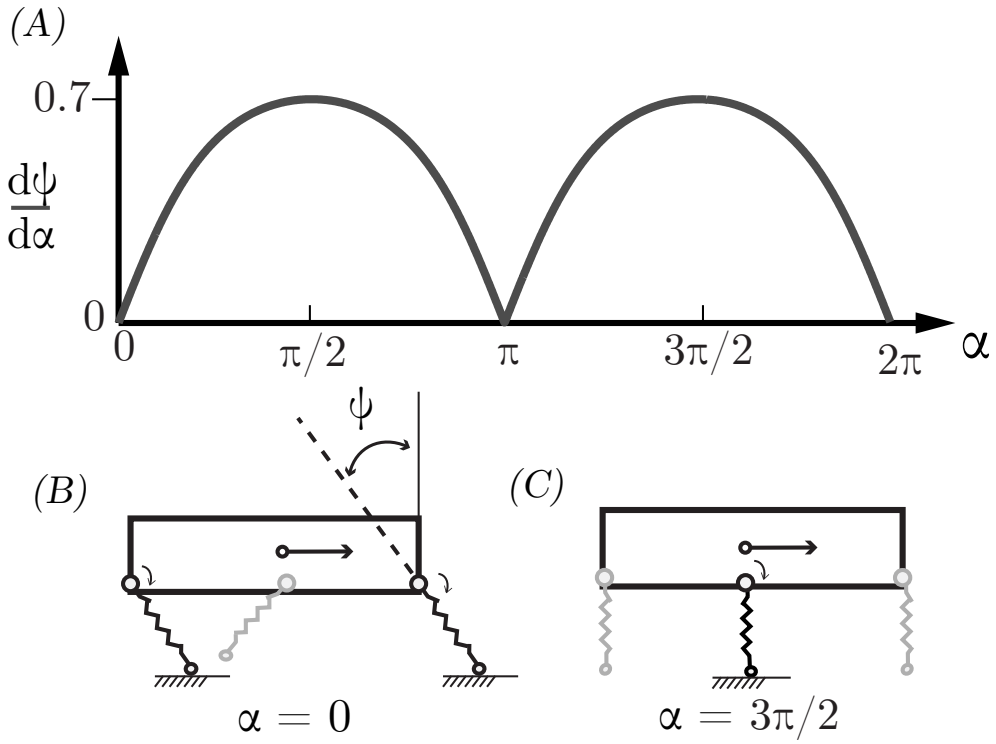


Figure 3.2: (A) Change in leg angle, ψ, with respect to motor crank angle, α. Note regions of dead-band around multiples of $\alpha = \pi$. (B) Robot leg configuration at $\alpha = 0$, corresponding to touchdown of the fore and aft legs. (C) Robot leg configuration at $\alpha = \frac{3\pi}{2}$, mid-stance of the middle leg.

3.2.1 Robotic System

For this analysis, we observed the dynamics of the VelociRoACH [28] running on a treadmill. The VelociRoACH is a hexapedal robot capable of stable running at 27 body-lengths per second with an alternating tripod gait. At the cost of some speed and stability, we did not enforce an alternating tripod gait in order to allow differential drive (diff-drive) steering. In this way, we used the robot as its own disturbance and were able to observe the dynamics of running in a larger region around a stable operating point. It has one actuated degree of freedom per side, a coreless brushed DC motor driving a rotary crank, which drives a set of kinematic linkages. These linkages are made with the Smart Composite Microstructures process [36], and convert rotary motion to leg abduction and adduction. See Haldane et al. [28] for details on mechanical design and dynamic performance. A key feature of this robot is that the leg kinematics for each side are predetermined by the geometry of the kinematic linkages, i.e. there is 1 degree of freedom per side. As the motor crank rotates, the robot takes one step with the fore and aft legs, followed by a step with the middle leg, as shown in Figure 3.2. We define one stride to be one full rotation of the motor crank.

A lightweight electronics package controls the VelociRoACH's two motors and streams

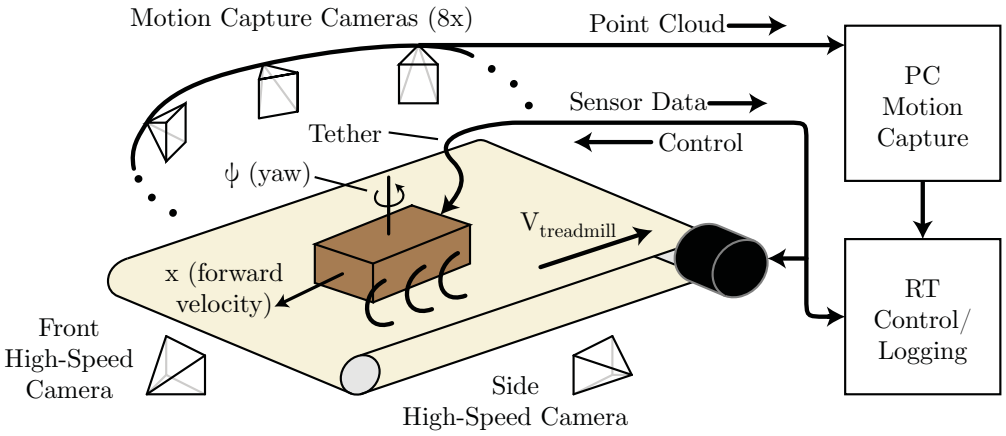


Figure 3.3: Experimental setup.

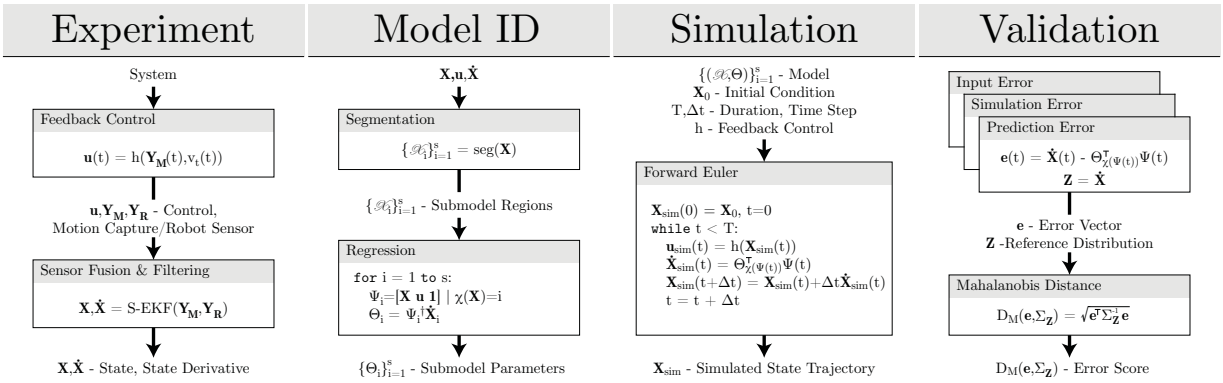


Figure 3.4: A block diagram of the method used to learn and test PWA models. Data flows from top to bottom and left to right.

telemetry data from its on-board sensors. These sensors include a six axis inertial measurement unit (IMU) and 14-bit absolute encoders on the output cranks of each side. The time series of robot sensor information is referred to as $\mathbf{Y}_R(t)$. The full design for the electronics and embedded software can be found from the references in Appendices B.1 and C.2 respectively.

3.2.2 Experimental Setup

To collect data on dynamic running, we designed a treadmill system with motion tracking and closed-loop position control. The treadmill simulates constant velocity forward running on flat ground, while the motion capture system streams and records robot position information to a real-time control system, which in turn provides motion commands to the robot. High speed video footage was captured from the front and side of the treadmill to observe

ground-leg interactions and verify the state trajectory information. Figure 3.3 shows the experimental setup.

For this experiment, the VelociRoACH was modified by adding reflective markers and a lightweight tether. The markers enable motion capture, and the tether provides power and serial communication to the controller. The combined weight of these modifications is less than 1 gram, with negligible effects on the inertia of the robot. The tether did not generate any measurable forces on the robot, and the robot was loaded with a battery for mass. We therefore assume that data collected on the treadmill will be valid for tether-free running on a similar surface.

The treadmill area measures ∼30 cm x 50 cm, allowing 5 robot body lengths of motion in each respective planar axis. A canvas belt provides sufficient traction such that the robot does not slip during nominal running. The belt is driven by a motor with integrated speed controller, allowing the belt to be commanded to speeds up to 5 m/s. Using the motion capture system, we verified that the belt velocity was a linear function of the commanded velocity with a maximum standard deviation of $\sigma = 0.023$ m/s.

The motion capture system calculates the position and orientation of the robot $\mathbf{Y}_M = [x\ y\ z\ \psi\ \theta\ \phi]^\top$ at 100 Hz with sub-millimeter resolution and maximum standard deviations of $\sigma_{xyz} = 0.027$mm and $\sigma_{\psi\theta\phi} = 0.0026$rad within the capture volume, respectively. These data are streamed to a real-time control system, which calculates and streams a control signal to the robot. The position control loop is executed at 100 Hz, using the most recent available data from the motion capture system to reduce network latency and jitter. The max round-trip latency was observed to be 30 ms, with an average latency of 10 ms.

To time synchronize the robot, motion capture, and video data, an infrared LED on the robot was driven by a square wave originating from the real-time controller. This LED was detected as a marker by the motion capture system. The synchronization signal, along with all robot sensor data, are streamed to the real-time controller for logging at 1 kHz. The VelociRoACH has been shown to have significant oscillatory energy in frequencies up to 100 Hz during high speed running [28], which necessitates this high sampling rate.

3.2.3 Feedback Control

To control the heading and velocity of the robot, a simple planar differential-drive model was assumed. This clearly ignores much of the complexity of legged locomotion dynamics, but we found it sufficient for keeping the robot on the treadmill long enough to gather data. Our trials were approximately two minutes in length. Figure 3.5 shows the coordinate system and variables used for this simplified control. Under this model we apply the proportional control in Equation (3.1) to find a desired robot velocity command. Figure 3.4a refers to this function as $\mathbf{u} = [\tilde{\dot{x}}\ \tilde{\dot{\psi}}]^\top = h(\mathbf{Y}_M, v_t)$ since it operates directly on motion capture sensor information and treadmill velocity v_t.

$$\tilde{\dot{x}} = \frac{v_t - k_x x}{\cos(\psi)} \quad , \qquad \tilde{\dot{\psi}} = -k_\psi \psi \tag{3.1}$$

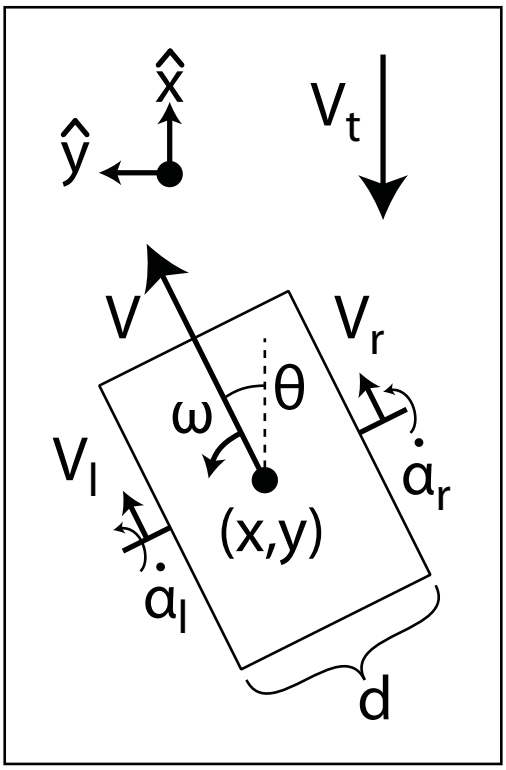


Figure 3.5: Differential drive model used for control.

Assuming positive v_t and a bounded region of the controlled state-space around $\mathbf{x}_c = [x\ y\ \psi]^\top = [0\ 0\ 0]^\top$, positive gains $k_{x,\psi}$ can be found to guarantee the stability of this model for the given region and treadmill velocity. This be shown by considering Lyapunov function $V(\mathbf{x}_c) = ||\mathbf{x}_c||_2^2$. In practice, the control gains were manually tuned to keep the robot stable at each treadmill velocity. We recognize that assuming this model limits the region of the state space, and thus model dynamics, we can expect to exercise and measure on the robot. This is part of an open problem of data-driven model identification where it is not trivial to control towards all viable regions of the state space of the robot. The approach presented here is a first step in model identification for the VelociRoACH, with the understanding that it will be incomplete in fully specifying the dynamics of the robot. Future work may be able to use a completely unsupervised state space exploration approach to move closer to true black-box model identification.

Low level PID control on the position and velocity of the legs is executed at 1 kHz on-board the robot. This control translates a desired robot longitudinal and yaw angular velocity $[\tilde{\dot{x}}\ \tilde{\dot{\psi}}]^\top$ to nominal left and right leg crank angular velocities $[\tilde{\dot{\alpha}}_l\ \tilde{\dot{\alpha}}_r]^\top$. Equation 3.2 shows this conversion for a robot with width d and effective leg radius r.

$$\begin{bmatrix} \tilde{\dot{\alpha}}_l \\ \tilde{\dot{\alpha}}_r \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & {}^{d}/_{2} \\ 1 & -{}^{d}/_{2} \end{bmatrix} \begin{bmatrix} \tilde{\dot{x}} \\ \tilde{\dot{\psi}} \end{bmatrix} \tag{3.2}$$

3.2.4 Data Fusion and Filtering

In order to study leg function and body motion in dynamic running and turning, we implemented an off-line sensor fusion framework to estimate full robot state. The end result of this off-line data processing is a pair of state and state derivative trajectories $\mathbf{X}(t)$ and $\dot{\mathbf{X}}(t)$ that can be used to learn a model mapping from $\mathbf{X}(t)$ to $\dot{\mathbf{X}}(t)$. The 16-dimensional state vector $\mathbf{X}$ consists of the first-order position variables $\mathbf{q}$ and their continuous-time derivatives $\dot{\mathbf{q}}$. Shown in Equation 3.3, $\mathbf{q}$ contains the position and orientation of the robot body, and the crank angles (α_r, α_l) of each leg mechanism. Robot position and orientation are recorded in the world frame, while all other variables are considered in robot-fixed axes. The highest order physical variables are accelerations in the $\ddot{\mathbf{q}}$ portion of $\dot{\mathbf{X}}$.

$$\mathbf{q} = \begin{bmatrix} x & y & z & \psi & \theta & \phi & \alpha_l & \alpha_r \end{bmatrix}^\top \tag{3.3}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}, \; \dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} \tag{3.4}$$

This effectively imposes a structure on the endogenous motion models we learn to be of the form

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} 0 & \mathrm{I} \\ A_{\ddot{\mathbf{q}},\mathbf{q}} & A_{\ddot{\mathbf{q}},\dot{\mathbf{q}}} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}, \tag{3.5}$$

where the only free parameters are in the blocks $A_{\ddot{\mathbf{q}},\mathbf{q}}$ and $A_{\ddot{\mathbf{q}},\dot{\mathbf{q}}}$. This is imposed in the regression steps of Section 3.2.5 by only including $\ddot{\mathbf{q}}$ in $\mathbf{X}$ in the actual pseudoinverse calculation.

The goal of this filtering framework is to produce a consistent trajectory for $\mathbf{X}(t)$ and $\dot{\mathbf{X}}(t)$ given the multiple noisy measurements of the state from the motion capture system and on-board sensors, such that for periods on the order of a time step, the error in the Euler integration approximation $||\mathbf{X}(t + \Delta t) - (\Delta t\, \dot{\mathbf{X}}(t) + \mathbf{X}(t))||_2$ is low. A Kalman filter framework provides a natural way to impose simple inertial dynamic constraints, and to appropriately weight the contributions of the motion capture and on-board sensors by their respective covariances in measurement.

Measurements of position and orientation were collected at 100 Hz using an OptiTrack™ motion capture system. Sensor noise models for the motion capture and IMU data were empirically derived by analyzing 1,000 seconds of telemetry data, which were streamed while the robot was stationary. Spatially independent Gaussian noise was assumed on each measurement channel, and the variance for each was subsequently calculated for the sensor

measurement model. This assumption mostly supports integration with simple dynamic models in the smoothing Kalman filter step.

The incoming data was filtered for outliers. All observed outliers occurred as motion capture tracking errors wherein the state measurement of the robot was miscalculated by several orders of magnitude. These outliers were filtered by rejecting any data point which was more than 4 standard deviations away from the mean. On average, this filter caught 3-4 outliers per 2 minute run. The initial and final twenty strides were clipped from the data set to rule out any transient effect from treadmill start up and slow down. A virtual bound was placed on the treadmill running surface to define a conservative operational area for the robot. If at any time the robot exited this area, a fault condition was set, and all consequent data was discarded from the analysis.

There was a small amount of variance in the sampling period for both the motion capture and robot data. A bicubic spline interpolation was used to time shift this irregularly sampled data to the nearest 1 ms time step. The maximum interpolation distance is bounded by one half of the sampling period.

To fuse the data, we used an Extended Kalman Filter (EKF) [72] combined with a minimum-variance smoother (S-EKF). The difference in sampling rates between the robot and the motion capture was accounted for by using a time varying observability matrix which limited observations of the motion capture state to valid timestamps. We used the motion model developed by van der Merwe et al. [50] to forward propagate the state. The empirically derived mass and inertial properties of the robot derived in [28] were used to predict the gravitational and Coriolis forces of the model.

The motion model for this approach predicts 6-DOF single rigid body dynamics in gravity for $\dot{\mathbf{q}}$ and $\mathbf{q}$ with high certainty, and with low certainty for $\ddot{\mathbf{q}}$. This was accomplished by setting a diagonal R matrix for the EKF with σ^2 values on par with the variance of the motion capture system for the position and velocity states, and σ^2 values greater than 10 times the IMU variance for the acceleration states. This results in the motion capture data being "trusted" more for the fused estimates of the position variables, a combination of the motion capture and IMU measurements for the velocity variables, and the IMU mostly contributing to the estimates of the accelerations.

To reduce the variance of the filtered data we performed a backwards pass using a Rauch-Tung-Striebel smoother [63]. The accuracy of the smoothed data was confirmed by applying a known motion profile to a dummy robot, and verified with high-speed video.

3.2.5 Model Identification

The models identified in this work are time-invariant piecewise affine state-space differential equations with exogenous inputs. Functionally this means a model maps a state and input to a single estimated continuous-time state derivative. For a fixed submodel, this map in classical linear system terms is: a system matrix A, input matrix B, with an affine forcing vector f. For convenience we concatenate A,B, and f to a parameter matrix Θ_i, and the

state, input, and affine component to a regressor vector $\Psi(t)$. Equation 3.6 describes this map to state derivative prediction, $\hat{\dot{\mathbf{X}}}(t)$.

$$\hat{\dot{\mathbf{X}}}(t) = \begin{bmatrix} A & B & f \end{bmatrix}_i \begin{bmatrix} \mathbf{X}(t) \\ \mathbf{u}(t) \\ 1 \end{bmatrix} = \Theta_i^\top \Psi(t) \tag{3.6}$$

Submodel parameters are indexed by i, which indicates the submodel region $\mathscr{X}_i \subset \mathbb{R}^n$ in which those dynamics hold. The regions are disjoint ($\forall i \neq j, \mathscr{X}_i \cap \mathscr{X}_j = \varnothing$), and complete ($\bigcup_i \mathscr{X}_i = \mathbb{R}^n$). A complete model consists of the collection of tuples of submodel parameters and regions $\{(\Theta, \mathscr{X})_i\}_{i=1}^s$, where s is the number of submodels. As Figure 3.4b illustrates, our model identification technique first partitions the space via statistical segmentation methods, then finds the submodel parameters using linear regression.

3.2.5.1 Segmentation

In a practical sense, we are interested in a balance between the number of model partitions and the overall accuracy of the estimation. To explore this relationship, we introduce and evaluate three different methods of partitioning: "Average", "z-score", and "k-means". To test the null hypothesis of no linear dynamics, we also define a null model $[A\ B\ f]_{\text{null}} = [0\ 0\ \mu]$, where μ is the average value of $\dot{\mathbf{X}}_i$. We use this collection of automatic region identification methods tests values of $s = [0, 1, 2, 10, 50]$.

The Average method considers all observed data in one region, and thus identifies one affine model for the entire system. This approach is a base case for a dynamical system.

The z-score method divides the observations into two partitions based on the empirical likelihood of the observation. First the data are z-score normalized, giving the distribution of observations mean 0 and variance 1 in each axis. A parameter σ_z is used to separate the data. Observations within σ_z of the origin in z-score space are considered one region, and the remaining data outside this ball are a second region. This approach is primarily used to see if a submodel fit to the most likely observations improves the overall model fit. We tested the z-score model with $\sigma_z = 1.5$ (listed as Z-1.5), which empirically improved the fit of the inner region over the outer region by an average of 25% per axis.

To extend this method to more partitions, we used the k-means clustering method. First, all data in $\mathbf{X}$ are z-score normalized. Then, standard k-means is applied to the normalized data to identify $k = s$ model partitions. We evaluated this approach for $s = 10$ and $s = 50$ to define the K-10 and K-50 models. Varying s allows tuning of the model granularity; we chose a maximum of s=50 so that the partition could be computed in a few minutes. Empirically we found that the number of observations in each region was approximately uniform for each trial.

For convenience, we define the submodel selecting function χ in Equation 3.7, which maps a state vector to the submodel region index i that the state portion of $\mathbf{X}$ resides in. We also use the notation $\chi(\Psi)$, which is understood to act only on the $\mathbf{X}$ portion of Ψ. Where relevant, the subscript M denotes which model partition is used.

$$\chi : \mathbb{R}^n \to \{1, ..., s\}$$

$$\chi_M(\mathbf{X}(t)) = i \quad \text{s.t.} \quad \mathbf{X}(t) \in \mathscr{X}_i \tag{3.7}$$

In practice the submodel regions can be defined explicitly with polytopic separating planes, or implicitly by other methods. In the case of z-score and k-means segmentation algorithms, χ can be more simply evaluated by storing the parameters of the model and calculating the regions of a new observation from those. For example, by storing the means of the k-means segmentation method as $\{\mu_i\}_{i=1}^s$, the region index can be calculated as:

$$\chi_{K-s}(\mathbf{X}) = \underset{i \in \{1,...,s\}}{\arg\min} ||\mathbf{X} - \mu_i||_2 \tag{3.8}$$

We use this approach to simplify the representation in the simulation results discussed below. In general any method that reproduces the complete and disjoint mapping will suffice.

3.2.5.2 Regression

Once the data are segmented, we estimate the dynamics of each region of the observed data using least squares. Given a collection of correlated observations, the learned models are an estimator of $\dot{\mathbf{X}}(t)$, given $\mathbf{X}(t)$ and $\mathbf{u}(t)$. The estimation error $\mathbf{e}(t)$ for a particular observed $\dot{\mathbf{X}}(t)$ and model is therefore:

$$\mathbf{e}(t) = \dot{\mathbf{X}}(t) - \Theta^\top_{\chi(\Psi(t))} \Psi(t) \tag{3.9}$$

By collecting all observations in a region $\mathscr{X}_i$ as column vectors in Ψ_i, we can calculate the submodel parameters that minimize the region sum squared prediction error in $\mathbf{e}_i$ as Equation 3.10, where the dagger represents the Moore-Penrose pseudoinverse.

$$\Theta_i = \Psi_i^\dagger \dot{\mathbf{X}}_i \tag{3.10}$$

3.2.6 Simulation

The PWA models identified by this approach can be used to simulate a system state trajectory (Figure 3.4c). The simulations presented in this work use Euler integration with exogenous control and state determined submodels (Equation 3.8) to generate trajectories. At each time step, the control $\mathbf{u}_{sim}$ is calculated from $\mathbf{X}_{sim}$ using the same control law discussed in 3.2.3. The submodel parameters Θ_i used to calculate the state derivative are chosen such that $i = \chi(\mathbf{X}_{sim})$.

3.2.7 Validation

Comparing the performance of these models requires a metric that relates vector quantities in different spaces based on the distribution of values in that signal. We chose the Mahalanobis

Distance (D_M) of an error vector $\mathbf{e}$ with respect to collection of observations $\mathbf{Z}$. $\Sigma_{\mathbf{Z}}$ is the covariance matrix of the distribution of $\mathbf{Z}$.

$$D_M(\mathbf{e}, \Sigma_{\mathbf{Z}}) = \sqrt{\mathbf{e}^\top \Sigma_{\mathbf{Z}}^{-1} \mathbf{e}} \tag{3.11}$$

The remainder of our discussion will refer to a vector $\mathbf{e}$ as an "error", and the scalar value calculated by D_M as a "score". A higher score corresponds to a greater error, and thus a poorer prediction.

3.3 Results

Our results are presented in three major sections. The first describes how the models perform strictly as a predictor of the state acceleration based on the state and input. Next we evaluate how forward simulation of state trajectories varies across the modeling approaches and duration of simulation. Finally we examine how the state-space partitions identified by segmentation may be correlated with physical non-linearities.

3.3.1 Data Selection

Our approach to model identification relies on collecting data from a robot running in a stable or desirable operating regime. Models fit to this zone of operation would then be most accurate around a nominal point of stable running. We collected running data from the VelociRoACH with treadmill speeds in the range of 0.1 to 0.5 m/s. Figure 3.6 shows the resulting distribution of leg phase observations as a function of treadmill speed. Leg phase is defined to be $\Phi = \alpha_l - \alpha_r + \pi$. The most stable operating speed was 0.25 m/s, corresponding to a stride frequency of approximately 5 Hz. At this leg frequency, the robot tends to passively converge to an alternating tripod gait ($\Phi = 0$) which indicates an empirical basin of stability for the alternating tripod gait in this region.

3.3.2 Model Evaluation

All of the models discussed in this work are predictors of state derivative, conditioned on state and control input. We have constrained the parameters of our identification to second-order state-space differential equations and so our prediction output space is the acceleration vector, $\ddot{\mathbf{q}}$. Table 3.1 reports the average prediction error for each of these variables for all five of the modeling approaches we consider. These results were generated using 10-fold cross-validation between model generation and prediction. The table values are the average absolute error between the predicted and actual value, normalized by the standard deviation of that value. The standard deviation of each variable is reported in the last row to provide scale.

The null model shows a baseline normalized error of 1 for all variables. The prediction accuracy improves as the number of model regions increases. This validates the hypothesis

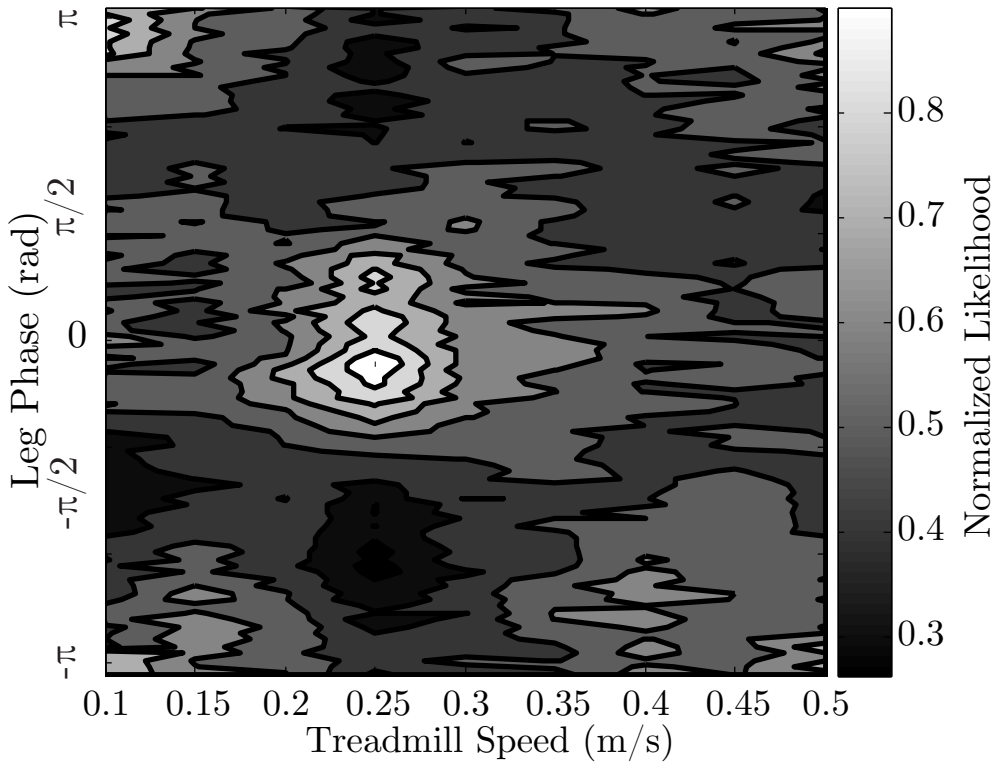


Figure 3.6: Contour map of probability of leg phase plotted against velocity. Lighter regions are more likely to occur. There are approximately 100,000 observations for each measured speed.

Table 3.1: Model Comparison

	Variance Normalized Average Error							
Model	$\ddot{x}$	$\ddot{y}$	$\ddot{z}$	$\ddot{\psi}$	$\ddot{\theta}$	$\ddot{\phi}$	$\ddot{\alpha_L}$	$\ddot{\alpha_R}$
Null	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Avg	0.843	0.781	0.672	0.950	0.790	0.737	0.820	0.821
Z-1.5	0.791	0.760	0.622	0.925	0.768	0.730	0.744	0.765
K-10	0.719	0.728	0.594	0.839	0.648	0.663	0.727	0.713
K-50	0.627	0.638	0.521	0.684	0.516	0.529	0.575	0.555
$\boldsymbol{\sigma}$	6.581	10.18	9.449	515.9	294.7	112.1	2259	2203
units	m/s^2	m/s^2	m/s^2	rad/s^2	rad/s^2	rad/s^2	rad/s^2	rad/s^2

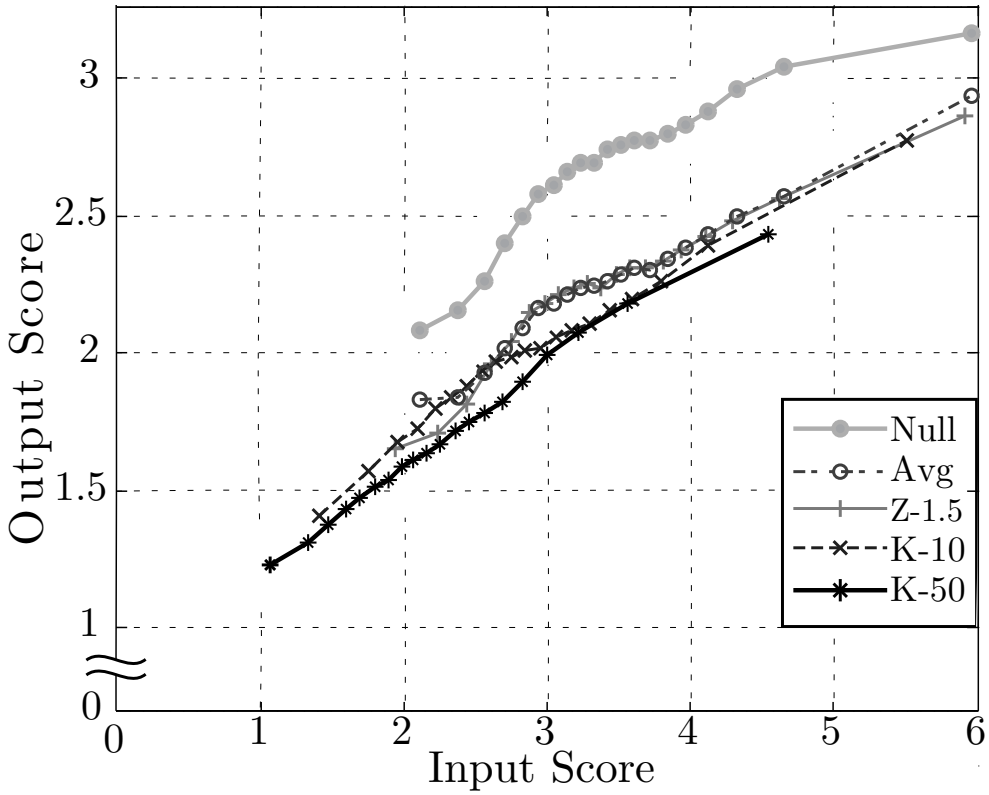


Figure 3.7: Model performance on an Input/Output basis, as a function of approach. Input score is defined as $D_M(C_{\chi(\mathbf{X}(t))} - \mathbf{X}(t), \Sigma_{\mathbf{X}})$ where $C_{\chi(\mathbf{X}(t))}$ is the centroid of the submodel region containing $\mathbf{X}(t)$. Output score is defined as $D_M(\dot{\mathbf{X}} - \hat{\dot{\mathbf{X}}}, \Sigma_{\dot{\mathbf{X}}})$. The marked points on each series are the mean of an equal number of observations, so that the density of the points indicates the distribution of observations on each axis.

that a linear model at least improves over the Null model with no dynamics, and that more submodel regions allow better local approximations of non-linear dynamics.

Figure 3.7 shows that as more regions are added, the average distance to a submodel centroid decreases. This empirically shows the intuitive positive correlation between input score and output score. We expect that an input observation close to the centroid of a subregion is representative of the behavior in that region, and thus the output prediction is best in that vicinity.

3.3.3 Simulation

As discussed in section 3.2.6, the models can be used to predict the behavior of the system. Figure 3.8 shows characteristic simulations for several models compared to the experimentally observed trajectory of the system. We chose to show forward velocity ($\dot{x}$) and yaw (ϕ), which are examples of first and zeroth-order state predictions. These variables are also

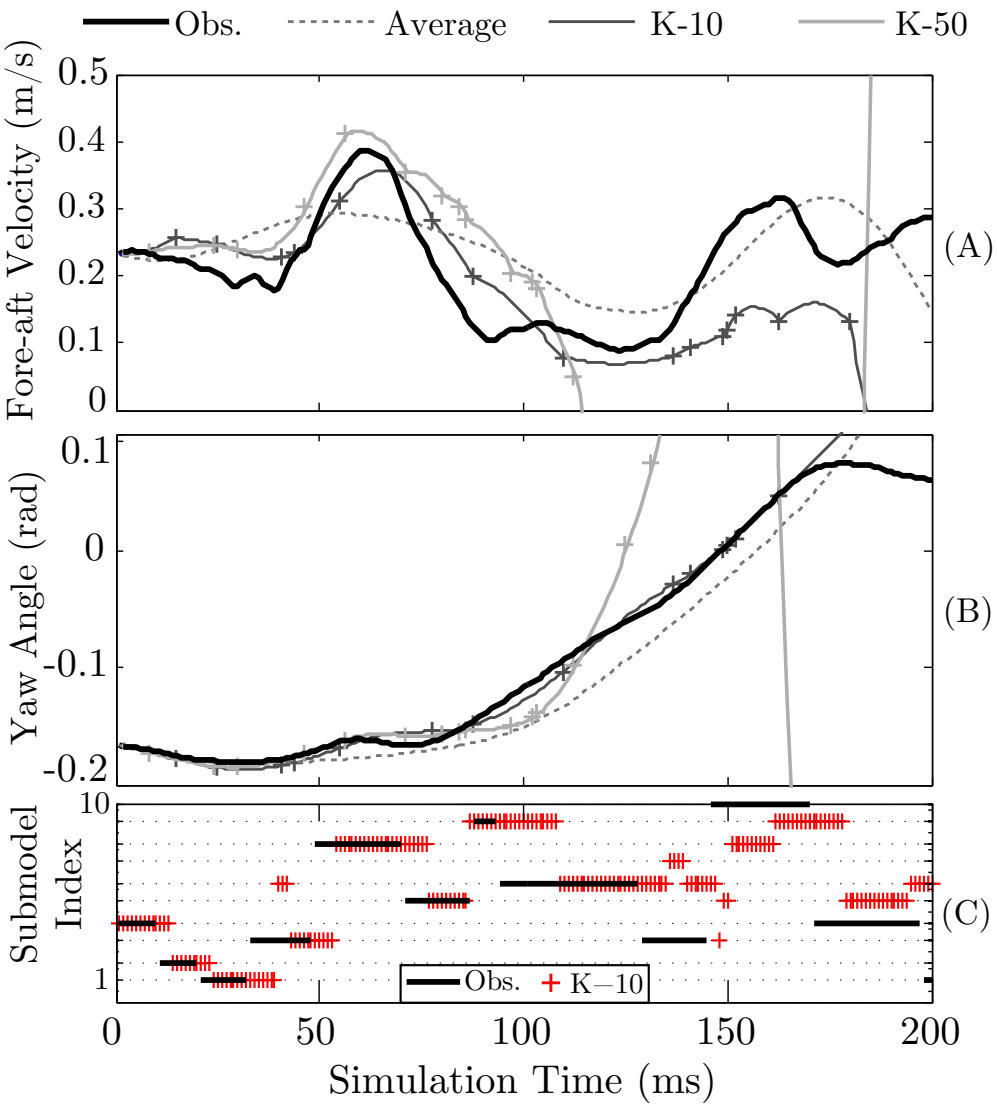


Figure 3.8: Predicted time trajectories of fore-aft velocity (A), Yaw angle (B), and the submodel index of the K-10 simulation state $\chi_{K-10}(X_{K-10})$ superimposed on the K-10 submodel index of the observed trajectory $\chi_{K-10}(X_{obs})$ (C). In (A) and (B), transitions between submodels are marked with + symbols.

useful to predict robot motion for turning maneuver planning. In each case we initialized the simulation state to an observed state, and then simulated the state and control for 200 1 ms time steps. The observed data for the duration of these simulation times was excluded from the data used to train the models. Figure 3.8 (A) and (B) show simulated trajectories of the Average, K-10, and K-50 models, along with the observed trajectory of the system.

The simulations qualitatively follow the behavior of the system through several submodel regions. After a simulation time horizon of 100-125 ms the model prediction of the state diverges from the observed state trajectory. Figure 3.8 (C) shows that the submodel transitions in the simulation lag the observed submodel transitions. This lag causes a divergence from the observed submodel region trajectory for the K-10 model. The dynamics of the K-50 model are noisier than those of K-10 due to a larger number of transitions between the affine submodels. The K-50 submodel trajectory diverges earlier than K-10 which shortens the time horizon of its usefulness.

The time horizon of useful predictions from simulations decays with the number for submodels. Figure 3.9 shows this trend of prediction accuracy versus simulation duration. The plot shows a Pareto frontier for the model complexity. The K-50 model is most accurate for short time spans (up to 40 ms), K-10 for a brief midrange (40-60 ms), and the average model remains closest to the observed trajectory from 60 to 200 ms. After 200 ms, all models make a poorer estimate than a null model with constant acceleration. We hypothesize that this phenomenon is due to almost all modeling approaches learning at least some unstable models at the periphery of the traversed stated space, and a transition to these models will most likely lead to exponential growth in the state.

3.3.4 Interpretation of Model Regions

A hypothesis of our model identification approach is that model regions will be roughly associated with distinct dynamic behaviors which are caused by physical nonlinearities. To explore how the models identified for our robot correspond to a physically interpretable dynamic structure, we project the likelihood of model transition onto a plane of the state-space where we expect regions of disparate dynamics to be readily identifiable.

The leg crank angle largely determines whether a leg is on or off the ground, so we expect it to generate the clearest distinctions in dynamics. To examine this hypothesis, we analyze how closely the observed dynamics match a simple diff-drive kinematic model as a function of the projection to the leg phase space (the plane of (α_l, α_r)). We consider the diff-drive state $\mathbf{X}_D = [\dot{x}\ \dot{\psi}]^\top$. The kinematic model predicts that $\mathbf{X}_D$ is the inverse of the linear mapping in Equation 3.2; we label this prediction $\hat{\mathbf{X}}_D$.

Figure 3.10a shows the differential drive Dynamic Score, defined as $D_M(\mathbf{X}_D - \hat{\mathbf{X}}_D, \Sigma_{\mathbf{X}_D})$, projected onto leg phase space. We see that the diff-drive model does not make accurate predictions of robot behavior over the course of a stride. Variations in the local dynamics of the robot from the kinematic model are shown by the color in this figure. Regions of high contrast indicate that the dynamics of a region are changing rapidly. These changes were

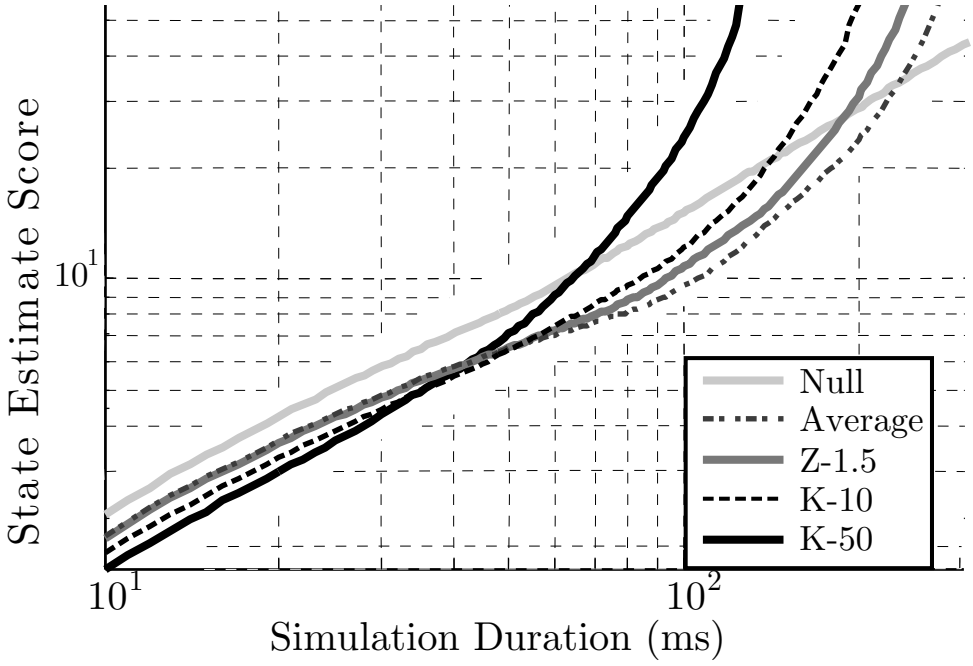


Figure 3.9: This plot was generated using 1000 simulations with initial conditions randomly selected from $\mathbf{X}$. State Estimate Score is defined as $D_M(\mathbf{X}(t_0 + t_{sim}) - \mathbf{X}_{sim}(t_{sim}), \Sigma_{\mathbf{X}})$.

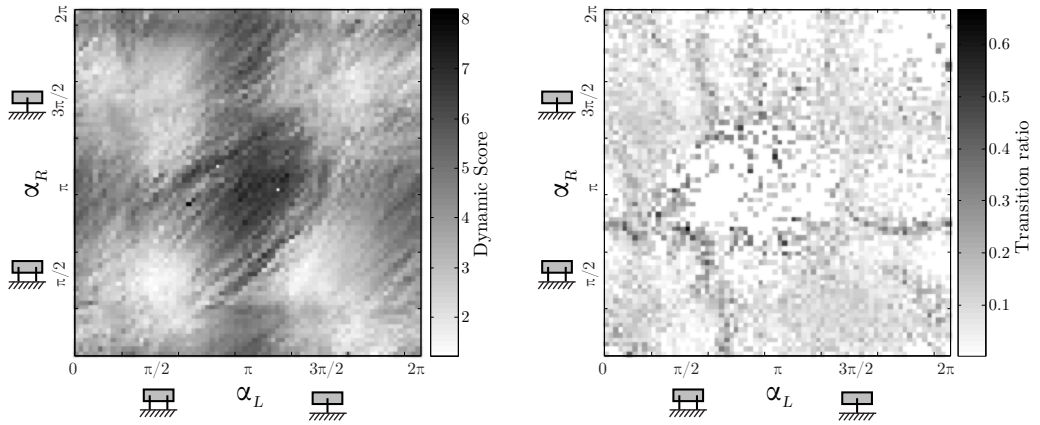
caused by leg touchdown and liftoff events, physical nonlinearities which ideally would be matched by submodel transitions.

Figure 3.10b shows submodel transition ratio projected onto the same leg angle space. The transition ratio is calculated by binning the state trajectories on this space, and reporting the ratio of trajectories that experience a submodel transition to the total number of trajectories in the bin.

If there were no structure to the observed state trajectories, then we would expect there to be an even distribution of submodel transitions. From a mechanical perspective, we would expect leg touchdown events to be hybrid transitions in the robot dynamics. The most salient trend visible in Figure 3.10b are the high probability bands in the vicinity of $3\pi/4$, which corresponds to liftoff of the front and rear legs (see Figure 3.2). These bands match well with several regions of high-contrast in Figure 3.10a.

3.4 Discussion

Figure 3.12a shows the vertical acceleration of the VelociRoACH for one full stride, while traveling at 0.25 m/s. For these strides, the VelociRoACH has passively converged to alternating tripod gait, which is most likely gait at this speed (see Figure 3.11). The first stride shows a characteristic SLIP-like pattern of acceleration. During the second stride, the robot reacts to a pitch perturbation, which disturbs the motion away from a SLIP-like regime.



(a) Heat-map of observed differential drive Dynamic Score.

(b) Heat-map of model transition ratio for the K-50 model.

Figure 3.10: Heat-maps showing observed robot and model dynamics projected on to leg phase space. The robot stick figures on the axes show how mid-stance for each of the legs is associated with the α variable. Top dead center for the front and rear legs occurs at $\pi/2$, and at $3\pi/2$ for the middle leg.

Both the K-10 and K-50 models accurately predict the motion for the full stride, whereas a SLIP model fit to the system would not have been able to predict the full body dynamics.

Because we used a data driven approach to modeling this system, we were able to document this newly observed dynamic behavior during the data collection process. We were interested on analyzing the performance of the steering controller, so we extracted a set of high speed turns from the experimental data collected at $0.25\,{}^{\mathrm{m}}/_{\mathrm{s}}$. Figure 3.12b shows how well the hybrid models tracked the observed Yaw angular acceleration, $\ddot{\psi}$ during one of these turns. The robot started facing forward, in the nominal operating condition. The models tracked well near the nominal state (with K-50 outperforming K-10) because a large amount of training data was available for this region. The performance of the models degrades as the robot exits this region due to the relatively small amount of training data around the perturbed state. The dynamics of this region could be learned using our method by introducing repeated disturbances such that the robot more frequently occupied this region of the state space.

3.5 Conclusion

We applied a highly simplified differential drive model to a legged robot, which fit well enough to control its position and heading on a treadmill up to speeds of 5 body-lengths per second. This allowed for the collection of a large dataset of robot state observations. We demonstrated a data-driven methodology which learns models that predict the observed robot 16 dimensional state and derivative on time scales of about one stride. The method-

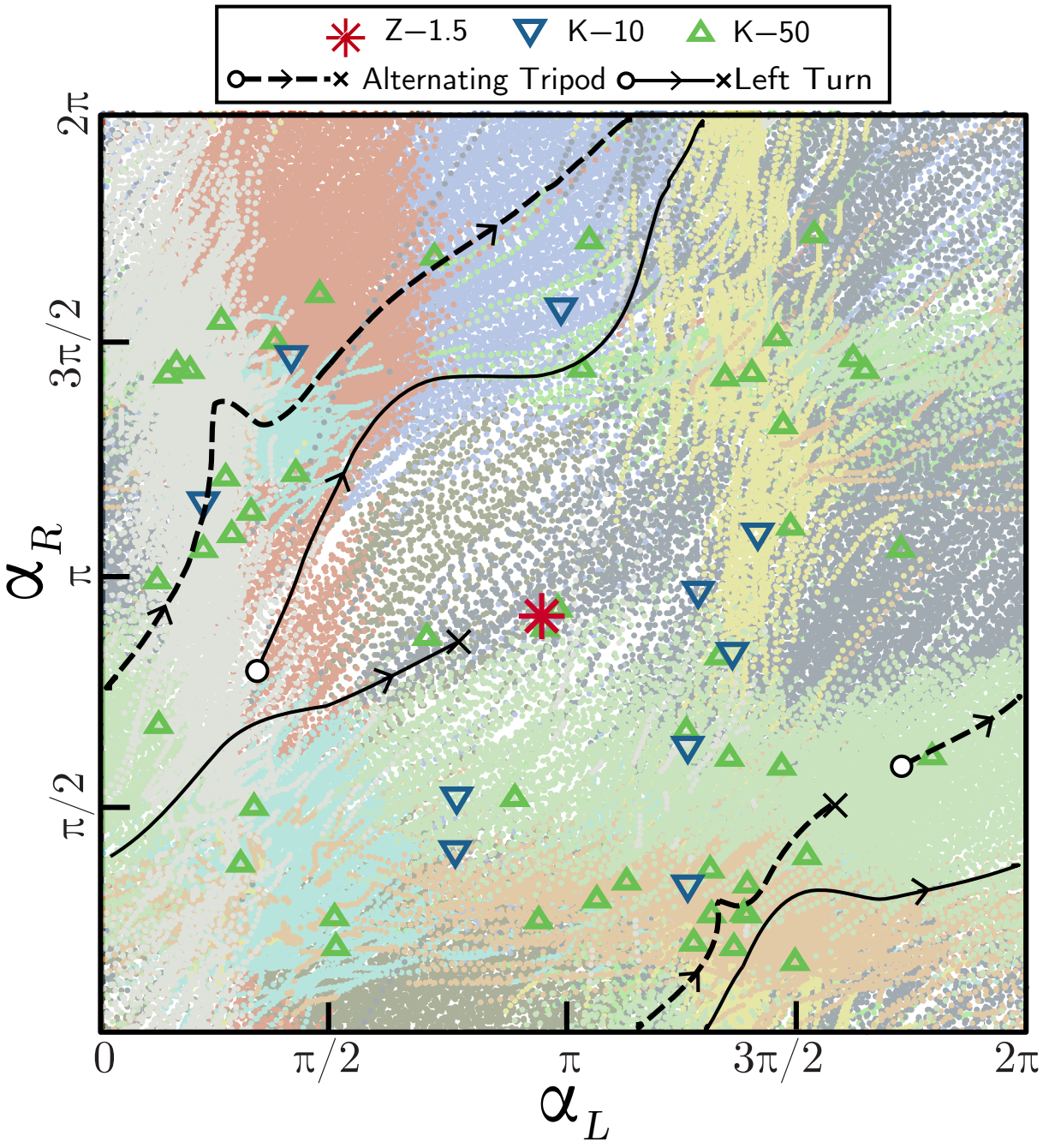
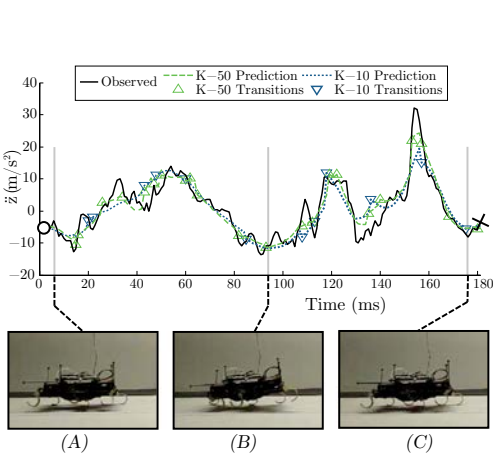
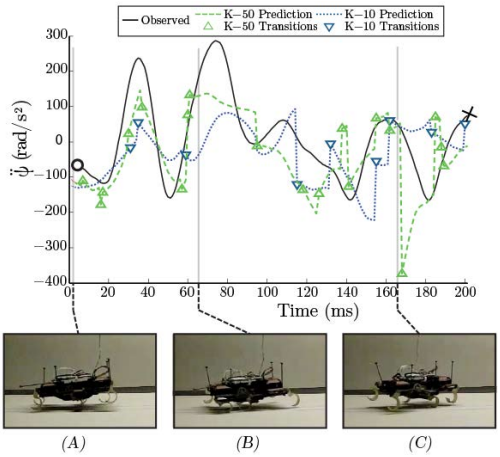


Figure 3.11: Plot of trajectory through phase space.



(a) $\ddot{z}$ for tripod gait.



(b) $\ddot{\psi}$ for left turn.

ology is platform agnostic, which will allow application of our approach to determine the dynamics of a wide variety of systems.

The predictive ability of the sets was measured via a Mahalanobis Distance metric (D_M). The one-time-step predictive performance of learned models increased with the granularity of the state space partitions. However, forward simulations of models with more subregions tended to diverge from the observed state trajectory in shorter time periods than simulations of simpler models. We expect this balance between short and long-term predictions to be a trend in models identified with this technique that have highly energetic transitions. The choice of model granularity will depend on the prediction and simulation goals of a specific robotic state estimation problem. Understanding exactly when the region-based modeling does not capture these transitions is relegated to future work.

Future work will target further improving the predictive ability of the data derived models. More sophisticated partitioning strategies could be employed by allowing the shape of the k-means clusters to more closely match the dynamical transitions of the robot. Iterative optimization techniques could then modify the regions to move towards improving the descriptive ability of the models with fewer model subregions. Analysis under the model selection framework of the Bayesian Information Criterion (BIC) could help automatically tune the balance of model complexity and prediction.

Simulation could also be improved over Euler integration by using models that explicitly allow non-continuous dynamics, and probabilistic dynamic transitions. Frameworks such as Gaussian mixture models and Markov Chains could add information about the likelihood of model transitions, and reduce the instability seen due to diverging from observed submodel index trajectories.

Finally, we are planning future work on using these learned models for on-line state estimation and control. The models are particularly amenable to an on-board Kalman filter or particle filter, as the simple to compute affine dynamical models can also be stored with an estimate of their accuracy. This relationship can be easily approximated with the

Input/Output score relationship in Figure 3.7, or extended to a full empirical motion model covariance. Extending to control and planning, other future investigations could use the models to investigate the dynamics of aggressive maneuvers, such as rapid turns.

With these additions, the work presented here would be a first step to a modeling and control paradigm that could be used on nearly any type of dynamical system. It has been shown to work well for identification and on-line control in lower dimensional systems, including ornithopter robots developed by Rose [66] in the Biomimetic Millisystem Laboratory. If state trajectories in regions of interest can be explored, our approach can identify a collection of affine models that predict the dynamics with tunable granularity. These models can then identify physical parameters of interest, provide a variable horizon simulation of the system, or provide empirical measures of prediction uncertain in probabilistic planning frameworks.

Chapter 4

Monocular Localization

4.1 Introduction

We present an approach for estimating the absolute poses of a swarm of Micro Autonomous Underwater Vehicles (μAUVs) by decomposing the problem into few absolute position estimations and many relative pose estimations. As power constraints are critical to small mobile robots, we develop an extension of Active marker pose estimation using color information to solve the marker correspondence problem, and show that this approach is more energy efficient than Reflective estimation approaches. We show the feasibility of this approach by localizing a robot navigating in an underwater test tank environment. Detailed analysis is presented characterizing the noise and error properties when estimating robot poses from fixed on-board markers. Moreover, we provide comparisons in power and computational cost for other popular methods of underwater localization.[1]

4.1.1 Motivation

Localization is a key capability for robotic operations in unknown environments when performing tasks such as exploration and inspection. Localization systems for small-sized autonomous robots are specifically challenging, because they need to consider design specifications of being small, low-cost, and low-power.

Micro autonomous underwater vehicles (μAUVs) have become an active research area in recent years. They open groundbreaking possibilities for applications related to automated information gathering such as monitoring of nuclear storage ponds [26] as well as of large industrial process tanks and port basins.

[1]This chapter to appear as "Low-Cost Monocular Localization with Active Markers for Micro Autonomous Underwater Vehicles," IROS 2017 [12]. A. Buchan developed the power theory of monocular estimation, adapted the pose estimation framework to include hue information, and designed and conducted experiments. E. Solowjow and D. Duecker developed the HippoCampus platform, and assisted with robotic control and experiment implementation. This work was supported by the German Research Foundation (DFG) under grant Kr752/33-1.

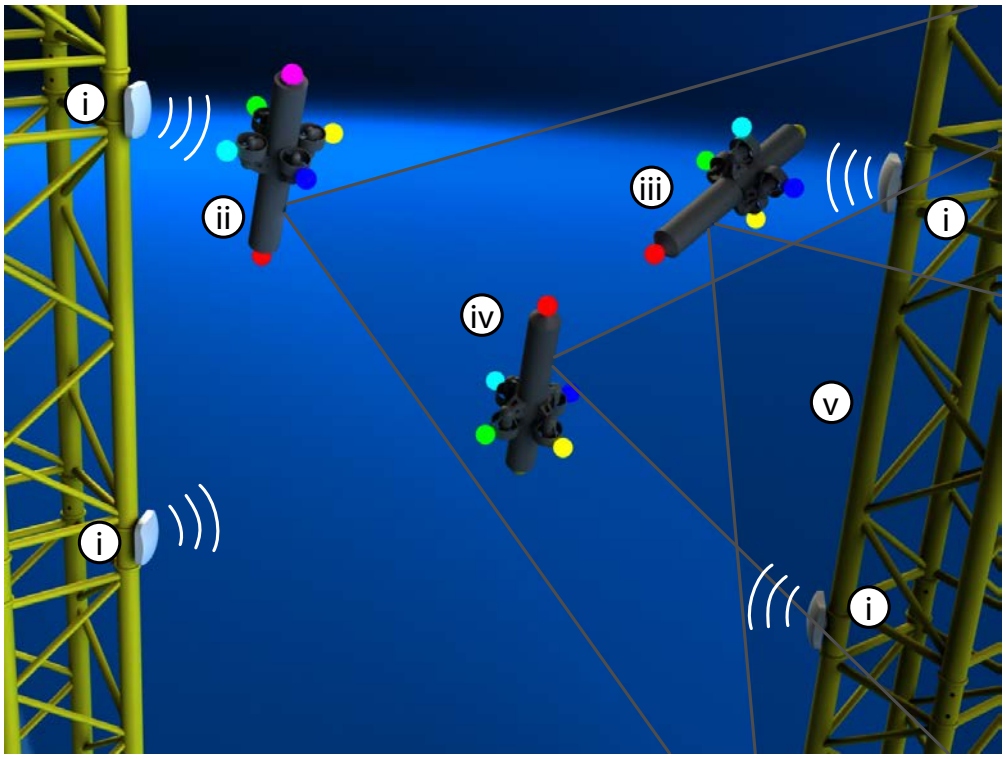


Figure 4.1: Underwater localization for inspection with proposed monocular vision and active marker technique. RF or acoustic beacons (i) provide global position information. An observer μAUV (ii) can measure the relative 6DOF pose of inspection μAUVs (iii, iv) to provide precise formation control through a communication channel to aid the overall inspection of structure [17] (v).

The requirements of μAUVs put unique constraints on localization methods. Aquatic environments attenuate most visual/electromagnetic signals quickly relative to air or space environments. When seeking localization solutions for low-cost teams of μAUVs, the payload size, cost, and power available for sensing and computation is limited. Localization of μAUV groups is a challenging topic and is considered as the current bottleneck to increased autonomy.

4.1.2 Localization Concept

We propose a strategy that allows localization of swarms of μAUVs with team members operating in proximity to each other, comparable to a school of fish. We assume the μAUVs can communicate with each other, e. g. via an acoustic channel. Instead of directly determining each swarm member's absolute pose, we suggest for μAUVs to obtain each other's relative poses first. Similar to a rigid system of particles we are left with six unknown degrees of freedom (6DOF) for the whole group (three translational, three rotational), because the

relative poses are invariant to translational and rotational motions of the μAUV group. Measuring the absolute positions (no orientations required) of just three swarm members called observers is sufficient to break the translational and rotational symmetries and to obtain the desired absolute poses of all swarm members. The three measured absolute positions must not be collinear. Alternatively, a single observer is sufficient for symmetry breaking if it can also determine its global orientation. Our approach is motivated by the fact that most underwater absolute positioning systems do not scale with team size because of two-way signal transmission, while the appeal of μAUVs lies in deploying many robots.

Since systems for determining global positions of underwater robots have been covered [24, 58], we focus on developing and characterizing the part of the system for estimating relative poses of μAUVs with monocular vision. We show that the proposed strategy is able to determine the full 6DOF relative pose of a μAUV with respect to a camera in underwater environments using minimal sensing hardware with low computational and power requirements. This approach is tested on a hardware setup that can be easily miniaturized and integrated to eventually enable autonomous swarm localization in the field. Figure 4.1 shows how this technique can be applied in an underwater structure inspection task. Only the observer vehicle (ii) needs access to global pose information. The remaining vehicles' poses are obtained via monocular vision.

4.1.3 Prior Work

Current underwater localization systems can be categorized in terms of the physical effect they are based on, i. e. acoustics, electro-magnetism and vision.

Acoustic positioning is the method of choice in open seas and thus full scale AUVs. However, its performance degrades significantly in confined environments, because of interference and reflections [18]. Furthermore, while localization errors of several AUV body lengths are usually tolerable in an ocean environment, small-scale operations require much smaller absolute errors. Small-sized, low-cost acoustic systems which would be suitable for μAUVs are therefore rare [24].

Recently a localization method based on the attenuation of electro-magnetic carrier waves in water was introduced, which delivers accurate absolute positioning results [58]. However, the range is limited to a couple of meters and a bulky signal analyzer is required to compute a radio spectrum. So far the electro-magnetic method is not suitable for μAUVs.

The work presented by Faessler et al. [21] provides an approach using monocular vision and infrared LEDs markers on-board the robotic system to be localized. Given a known 3D configuration of markers and an image of the system, this approach searches for a marker correspondence and estimated pose that minimizes the reprojection error of detected marker positions to predicted marker positions. This method has been shown to work well for quadrotor control, and is generally easy to implement and low-cost compared to other methods. The small size of current camera sensors make this approach ideal for implementing on board of μAUVs to localize to each other.

Active marker beacons with monocular vision are more suitable for short-range pose estimation in underwater environments, as opposed to methods that require ambient illumination such as Augmented Reality (AR) tags. Since μAUVs may be required to operate in environments with no ambient light, the energy losses associated with illuminating a target and sensing the reflected light are compounded compared to directly detecting active markers. In addition, AR tags would require a large flat area on board the sensed AUV be visible at all desired poses, which would be detrimental to the agile hydrodynamic performance of an AUV.

However, several shortcomings motivate our work in adapting this approach to μAUVs. Infrared light is more absorbed than the visible spectrum underwater [59], and given the ambiguity between markers it is difficult to distinguish between poses that have similar 2D projections of markers. Finally, the approach given in [21] will not scale well to identifying multiple robots, as the combinatorial problem of marker correspondence grows exponentially with the number of markers visible. Thus we propose using visible spectrum LED markers that can be distinguished by color for relative localization of a μAUV.

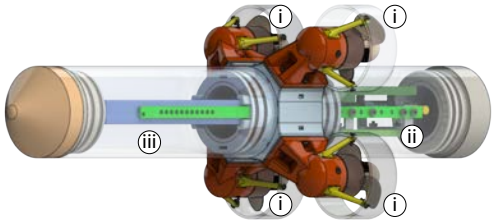
The prerequisites in [21] can be relaxed with the introduction of color for marker correspondence. Especially with μAUVs, finding configurations of markers that do not produce symmetric projections to an image plane when viewed from perspectives all around the robot is particularly difficult.

This technique has already been shown to work for a team of three low-cost terrestrial robots [54]. The following sections detail our adaptation of this approach to use colored LED markers to localize the μAUV HippoCampus, which is shown in Figure 4.2a and was developed for monitoring scalar and flow fields [27]. The cutaway render in Figure 4.2a shows the payload bay at the front of the robot that could carry one or more cameras for swarm monocular localization.

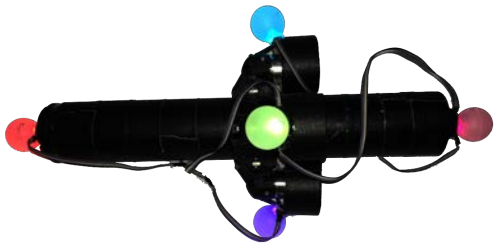
By just studying the monocular relative pose estimation of a single μAUV we condense the problem down to its essentials. This is the key aspect for the localization concept described in Section 4.1.2 and the main contribution of this work.

4.2 Theory

Here we develop a model for the energy required to obtain a pose estimate from two different vision based methods: Reflective and Active. Reflective methods encompass approaches like AR Tags, where light reflecting from a known visually patterned surface produces an image that can be used to extract the homography between the camera and surface. To evaluate the best-case engineered solution, we consider using a retro-reflective surface material that can reflect the majority of the optical energy in a narrow cone back towards the source (and also camera if it is placed nearby). For the purposes of μAUVs, we cannot assume there will be enough ambient illumination to view the surface, so we derive the energy necessary from a light source at the observer robot directed at the target robot to capture a single pose estimate image.



(a) CAD render of platform showing drive impellers (i) and hardware bay regions for housing processor and batteries (ii), with additional payload area (iii).



(b) Platform with Active LED markers used for localization.

Figure 4.2: Render and photo of HippoCampus μAUV platform.

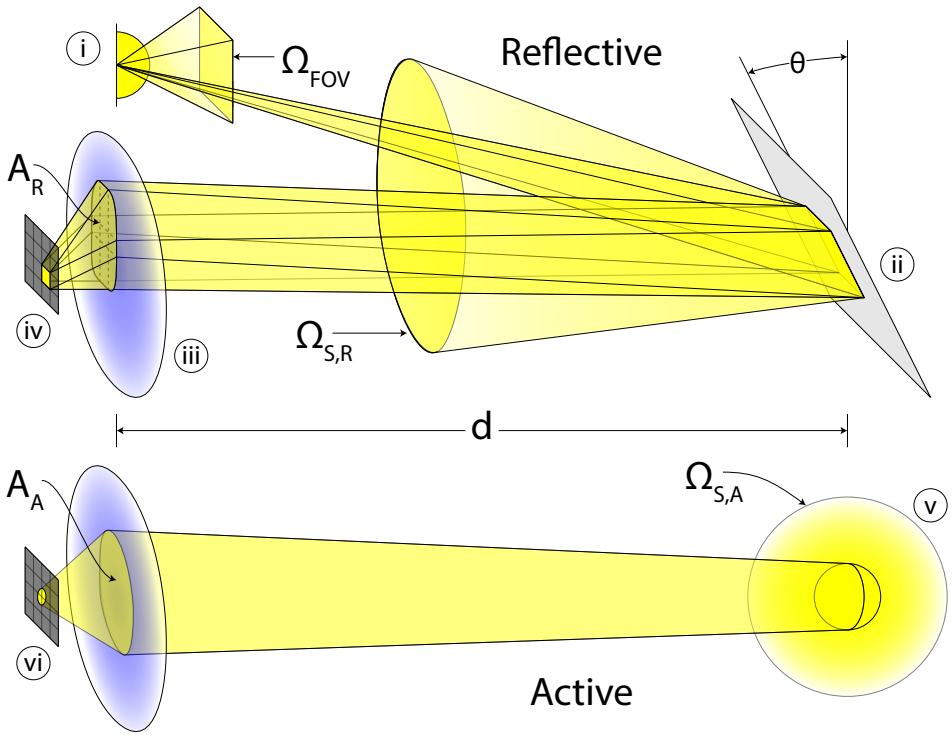


Figure 4.3: In Reflective estimation, the observer source (i) illuminates the field of view of the camera. A surface on the target robot (ii) reflects light through lens (iii) to illuminate a pixel on image sensor (iv). In Active estimation, multiple markers (v) on the target system produce an image on observer sensor (vi).

The Active approach (this work) derives the total energy required of Active LED markers on board the target robot necessary to produce the same type of image for extracting homography. In both cases, we first find the transmittance efficiency T for each approach, which represents the fraction of energy measured to the energy input to the system. The overall transmittance $T(\lambda, d)$ is a product of individual process transmittances T_x, which are variously functions of light wavelength λ, distance d, and properties of the medium and optics. Figure 4.3 illustrates the geometry of these processes for the two approaches considered.

We use a formulation similar to [8] to include all of the relevant processes: the source conversion of electrical to optical energy T_s, environmental Beer-Lambert scattering and absorption T_e, retro-reflection T_r, optics geometry T_o, and detection via the conversion of optical energy to signal T_d. The total transmittance is thus:

$$T(\lambda, d) = \prod_{x \in X} T_x = T_s(\lambda) T_e(\lambda, d) T_r T_o(d) T_d(\lambda). \tag{4.1}$$

Environmental transmittance is an exponential decay with a rate depending on the wavelength:

$$T_e(\lambda_c, x) = e^{-\gamma(\lambda_c)x} = e^{-\gamma_c x} \tag{4.2}$$

The difference between red ($\gamma_r \approx 0.3\text{m}^{-1}$), green ($\gamma_g \approx 0.04\text{m}^{-1}$), and blue ($\gamma_b \approx 0.01\text{m}^{-1}$) absorption has a significant effect on the Active case.

For retro-reflectors we assume that all light gathered within the projection of a pixel comes back towards the source in a very small cone ($\sim 10^\circ$ as per ASTM D4956 Type I rating for retro-reflective tape), thus, this is just the fraction of light collected by a single pixel:

$$T_r = \frac{1}{n_p} \tag{4.3}$$

The optics transmittance T_o is the ratio of the light collected by the pixel effective aperture A to the solid angle of the source Ω_S (either the reflected cone, or Active marker), which grows with the distance squared to the source. A is the effective aperture of a lens, given by the standard relation of the focal length f and F-Number N:

$$T_o(x) = \frac{A}{\Omega_S x^2}, A = \pi \left(\frac{f}{2N}\right)^2 \tag{4.4}$$

We assume for a particular wavelength λ_c there is a minimum total energy a pixel must collect $E_{D,\min}$ to achieve a desired Signal to Noise Ratio (SNR) for pose estimation. Thus, the minimum energy required by all sources $E_{S,\min}$ to measure a pose is the sum over the set of wavelengths C of the ratio of minimum detection energies over the transmittances:

$$E_{S,\min}(d) = \sum_{c \in \mathcal{C}} \frac{E_{D,\min}(\lambda_c)}{T(\lambda_c, d)} \tag{4.5}$$

These formulations will next allow us to compare the minimum energy required for a pose estimate in the Reflective and Active case as a function of distance.

4.2.1 Reflective Estimation

Assuming illumination with blue light in water, the Reflective transmittance is:

$$T_R(d) = T_s(\lambda_b)T_e(\lambda_b, 2d)T_r T_o(d)T_d(\lambda_b) \tag{4.6}$$

$$= \frac{T_{s,d}(\lambda_b)A_R}{n_p \Omega_{S,R}} \frac{e^{-2\gamma_b d}}{d^2} \tag{4.7}$$

where $\Omega_{S,R}$ is the solid angle of the retro-reflector cone. The minimum energy is:

$$E_{S,R,\min}(d) = \frac{E_{D,\min}(\lambda_b) n_p \Omega_{S,R}}{T_{s,d}(\lambda_b)A_R} \frac{d^2}{e^{-2\gamma_b d}} \tag{4.8}$$

$$= E_{b,\min} \frac{n_p \Omega_{S,R}}{A_R} d^2 e^{2\gamma_b d} \tag{4.9}$$

where $E_{b,\min}$ is an energy factor that depends only on the camera and illuminator properties. We report the energy at a given distance as multiple of this quantity since it is the same for both approaches.

4.2.2 Active Estimation

For Active estimation, we must consider the worst case of an individual marker image being split between 4 pixels. Thus, the transmittance of a single marker is:

$$T_A(\lambda_c, d) = \frac{1}{4} T_s(\lambda_c)T_e(\lambda_c, d)T_o(d)T_d(\lambda_c) \tag{4.10}$$

$$= \frac{T_{s,d}(\lambda_c)A_A}{4\Omega_{S,A}} \frac{e^{-\gamma_c d}}{d^2}. \tag{4.11}$$

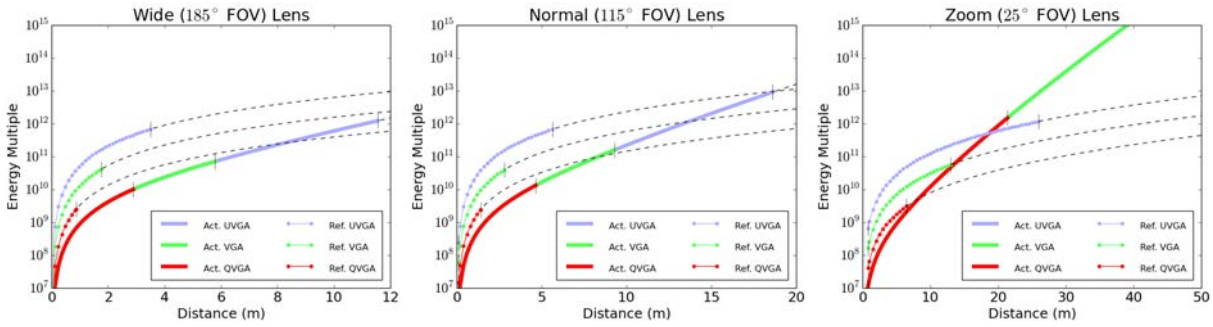
$\Omega_{S,A}$ is a full sphere (4π), but we leave it in for comparison. Assuming that the efficiency of production and detection of light is approximately the same for each color ($T_{s,d}(\lambda_c) \approx T_{s,d}(\lambda_b)$) yields

$$T_A(\lambda_c, d) = \frac{T_{s,d}(\lambda_b)A_A}{4\Omega_{S,A}} \frac{e^{-\gamma_c d}}{d^2}. \tag{4.12}$$

Finally, we consider the fact that we need 3 detections of each of red, green, and blue pixels to read the six colors used in this method (red, yellow, green, blue, cyan, magenta):

$$E_{S,A,\min}(d) = 3 \sum_{c \in \{r,g,b\}} \frac{E_{D,\min}(\lambda_b) 4\Omega_{S,A}}{T_{s,d}(\lambda_b)A_A} \frac{d^2}{e^{-\gamma_c d}} \tag{4.13}$$

$$= E_{b,\min} \frac{12\Omega_{S,A}}{A_A} d^2 (e^{\gamma_r d} + e^{\gamma_g d} + e^{\gamma_b d}). \tag{4.14}$$



(a) (b) (c)

Figure 4.4: Energy comparison between Active and Reflective estimation across lenses and resolutions.

4.2.3 Relative Efficiency

Figure 4.4 shows the required energy (as a multiple of $E_{b,\min}$) to acquire a pose estimate as a function of distance. Each subplot shows the behavior of each approach on a single lens across different standard resolutions (QVGA: 320×240, VGA: 640×480, UVGA: 1280×960). We considered the three lens types (wide angle, normal, and zoom) available for the open-source OpenMV embedded computer vision platform [57].

The line for each approach is solid between the minimum and maximum distances that the approach is viable. Minimum distance is limited by fitting the entire 35 cm robot within the vertical field of view of the camera, and maximum distance is limited by the camera resolution. We assume that a Reflective approach would need to be able to resolve 1cm features of a 5×5 pixel AR Tag, and the Active approach needs to resolve 3.3 cm features (allowing for markers spaced a minimum of 10 cm apart, and resolving at least one dark pixel in between on the diagonal). In general, the Active approach will have ~ 3 times the range of the Reflective approach since the features it needs to resolve are 3.3 times larger.

This is shown in Figures 4.4a and 4.4b for a wide and normal angle lens respectively. The Active approach always requires less energy for a given resolution in the distance range where the approaches are viable. Even though the energy losses of red wavelengths will eventually dominate the required energy in an Active approach, the limited range of Reflective approaches usually prevents them from realizing the benefits of using only blue light. Figure 4.4c shows that with sufficient magnification and resolution, a Reflective approach can require less energy than the Active approach at far ranges. The energy requirements for a Reflective approach at UVGA resolution drop lower than the Active requirement at around 18 meters. However, at closer ranges the Active approach still requires less energy than the Reflective approach at any lower resolutions, and for most ranges at UVGA resolution. In general, for having a balance between field of view and energy efficiency, the Active approach proves to be much better in theory than Reflective approaches. Of course, this model is subject to variability in implementation and assumptions of parameters, therefore

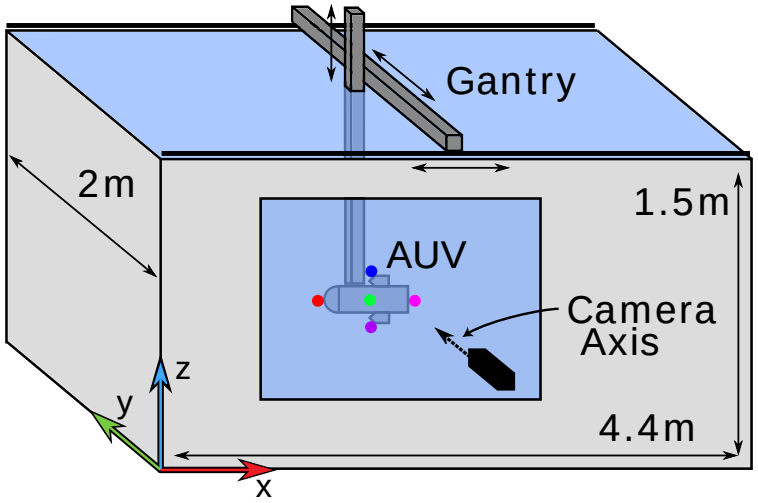


Figure 4.5: Experimental water tank setup for localization.

we experimentally verify the efficacy and energy requirements of the Active approach in the following sections.

4.3 Methods

4.3.1 Hardware Setup

The HippoCampus μAUV was outfitted with six waterproof RGB LEDs and spherical 35 mm diameter diffusers at the front, rear, and each of the four motor struts, as shown in Figure 4.2b. The RGB LEDs are programmable, which allows the exact colors of each marker to be arbitrarily reassigned during experimentation. The colors were chosen to provide as close to an even distribution in the Hue coordinate of the HSV color space [41] as possible when viewed underwater. This corresponds roughly to red, yellow, green, cyan, blue, and magenta colors (respectively, 0°, 60°, 120°, 180°, 240°, and 300° in Hue coordinates).

We simulated the behavior of how a camera on-board an observer μAUV would perform while viewing a different μAUV. We placed a robot in a water tank with observation panels. Localization image data was gathered by placing a Logitech QuickCam E3500 against a clear section of a water tank, and observing the robot with markers inside the tank. This tank was outfitted with an XYZ gantry that allowed repeatable and rigid positioning of the robot along the three axes. Orientation of the robot was held with a lockable spherical wrist gimbal. This orientation lock maintained a rigid pose, but was much more prone to human error when trying to attain accurate pose angles. Figure 4.5 shows this setup with the world coordinate frame in which we report results.

Adjustments to the exposure, gain, and white balance settings of the camera proved crucial to obtaining clear and identifiable images of the markers underwater. By manually

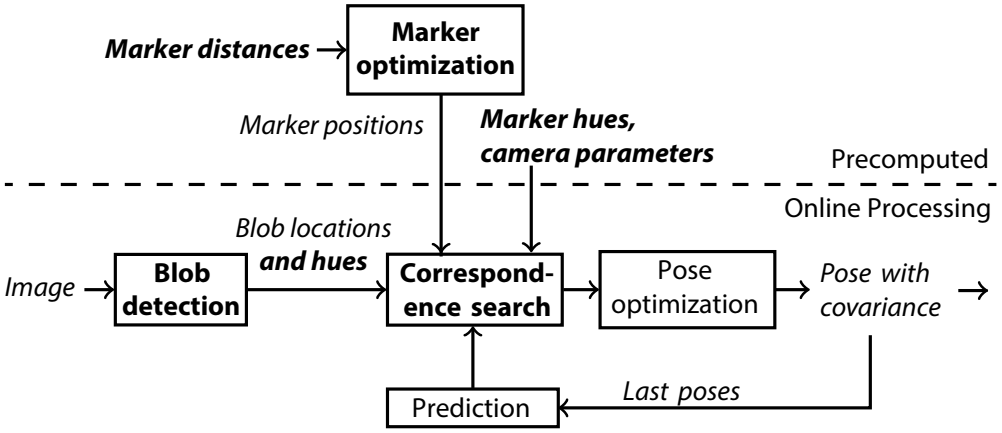


Figure 4.6: Algorithm dataflow (novel contributions in bold).

setting the white balance correction to accentuate red hues we were able to partially compensate for the higher absorption of red colors in water. We also found that using the lowest exposure time with high gain produced the highest contrast of markers to the environment. This had to be balanced by the fact that too high values of gain or LED brightness would wash out the markers to appear white, losing distinguishing color information.

4.3.2 Algorithm

4.3.2.1 Overview

Our approach follows that in [21] with a few key modifications. The main steps of Blob Detection, Correspondence Search, Pose Prediction, and Pose Optimization are used as shown in Figure 4.6. New information and modified steps are shown in bold. Blob Detection determines the centers and dominant hue of bright blobs detected in the camera image. The correspondence search assigns detected blobs to markers to obtain an initial pose estimate. The correspondence search can be initialized with a guess for where markers should be given a constant velocity model using the pose prediction based on the last two poses. Finally, once a consistent correspondence is found, the final pose is optimized using the Gauss-Newton method. This step also reports an estimated covariance of the pose estimate. In the following sections we detail the novel contributions to the blob detection and correspondence search steps allowing the algorithm to leverage color information.

4.3.2.2 Calibration

Camera calibration was performed using a checkerboard test pattern in the tank, allowing the system to identify the intrinsic properties of the camera viewing markers through the water. As it was difficult to measure the exact spatial configuration of the markers on the

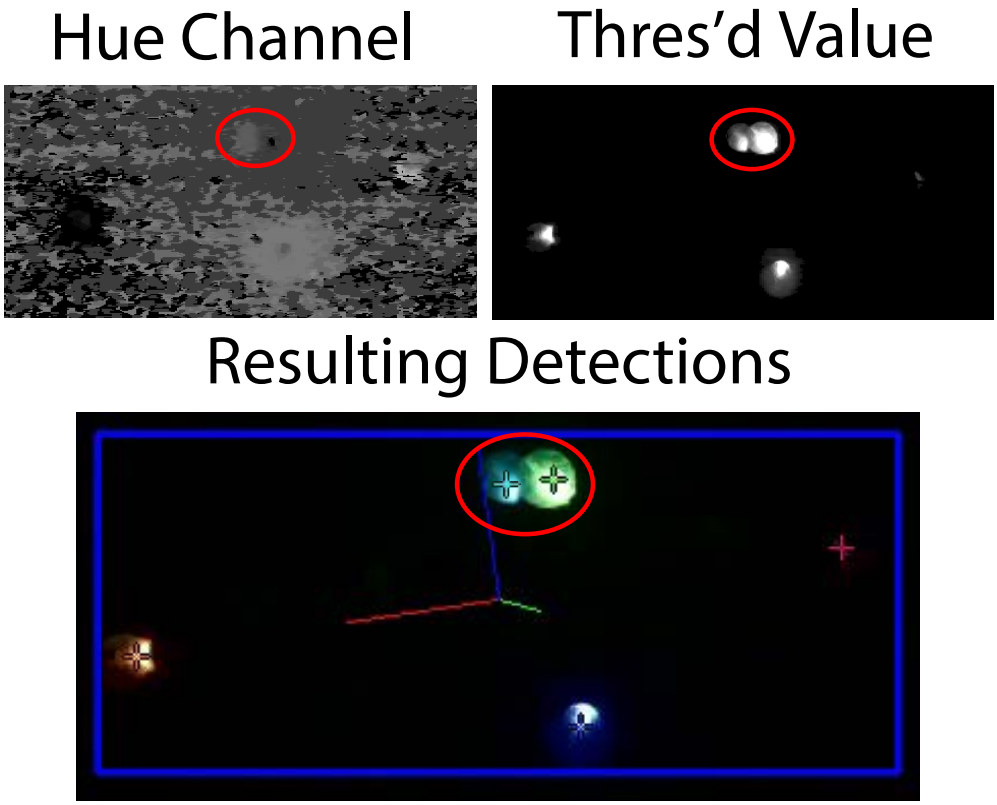


Figure 4.7: LED detections and resulting pose estimate on sample image. The final image shows the result of separating overlapping blobs based on their Hue coordinate.

robot, pairwise measurements between markers were taken. Then the BFGS optimization method [55] was used to find the positions of the markers that minimized the sum squared error of these measurements.

4.3.2.3 LED Detection

The image used in our approach is captured from the camera as a raw RGB pixel array. The image is converted to the Hue Saturation Value (HSV) color space using the OpenCV Library [7], which makes blob thresholding and color segmentation more convenient. LED blobs are detected by thresholding the Value coordinate of the HSV image, and then finding the contours of the remaining bright regions. The standard deviation of the Hue coordinate within these contours is calculated. If the variance is above a threshold within a contour, the blob is split into two regions with Hue values above and below the mean of the original blob. This allows the detection and separation of markers even when they partially occlude each other. In practice using half the separation between colors (30°) for this threshold worked well.

Figure 4.7 shows a grayscale representation of of the Hue and thresholded Value channels of a sample image. The Hue channel shows that the regions representing LED blobs have a roughly uniform and distinct coordinate, especially in the red circled area showing the cyan and green markers. The thresholded Value channel image shows the regions identified as blobs by removing all pixels below a certain magnitude (∼25% of the maximum possible Value coordinate). Note the single blob in the circled red region for two markers. Finally, Figure 4.7 shows the original image with the detected blob centers as colored crosshairs, the Region of Interest (ROI) where markers are expected as a blue rectangle, and coordinate axes representing the estimated pose. Here the result of separating a single blob into two based on the Hue covariance is clear.

4.3.2.4 Correspondence Search

The correspondence search step assigns detected blobs to markers on the object in order to calculate a pose using a Perspective-Three-Point (P3P) solver [42]. In the original algorithm, the search required initialization by computing a pose for every combination of three detected blobs assigned to each combination of three markers. The remaining blobs were then checked to see if they were consistent with the expected marker location. With this method the number of poses checked grows factorially in the number of markers on the object, and in the number of detected blobs.

Our approach uses the hues of the detected blobs to match them to the nearest marker in Hue coordinate, which can drastically reduce the number of pose checks. In the case that there are fewer detections than marker colors, there is only one required pose check. As the number of detections n_D grows larger than the number of colors n_C, the worst case number of required pose checks N grows as

$$N = \left\lfloor \frac{n_D}{n_C} \right\rfloor^{(n_D \bmod n_C)} \cdot \left\lceil \frac{n_D}{n_C} \right\rceil^{(n_C - (n_D \bmod n_C))} \tag{4.15}$$

where $\lfloor\rfloor$ and $\lceil\rceil$ respectively are the floor and ceiling functions. This function only grows exponentially in n_C and linearly in n_D. As our approach using colored LED markers tends to produce far more false detections than the IR LED marker approach, this modest growth in the pose checking computation time is an added benefit. For our use case of $n_C = 6$, this requires only 64 checks for a particularly poor $n_D = 12$, whereas the original approach would have required $105,600$ checks.

4.4 Results

In this section we present experimental results for dynamic open-loop trajectory tracking of a μAUV and static error analysis.

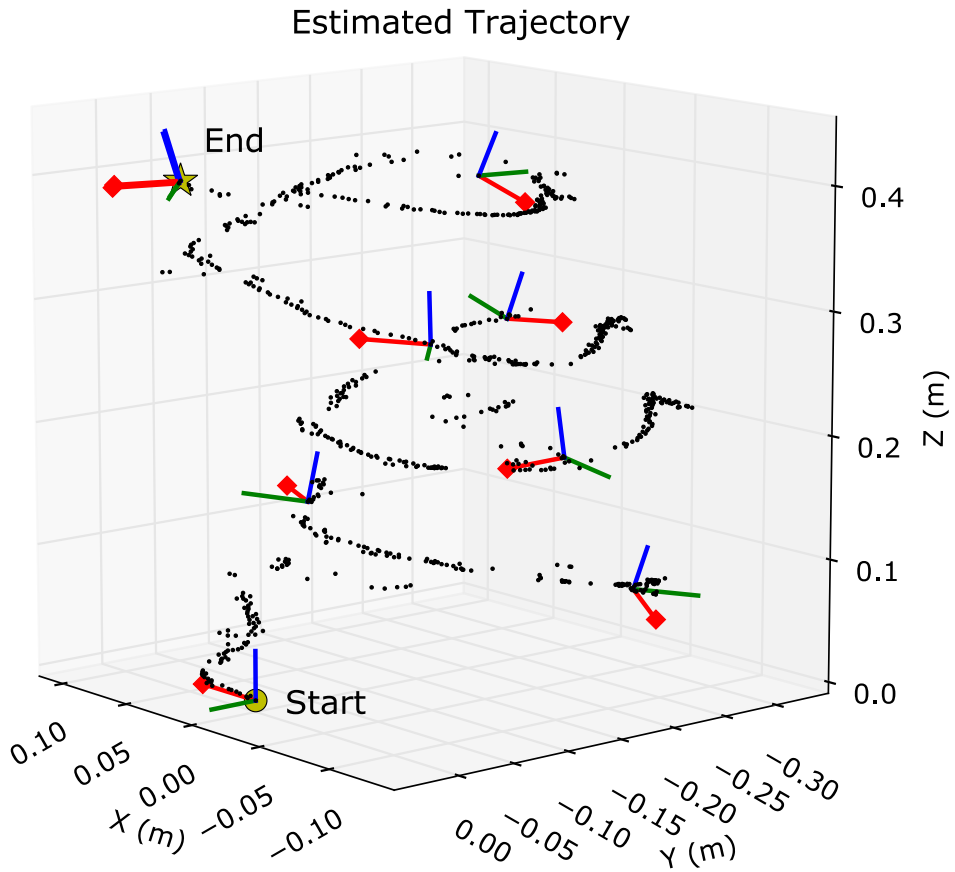


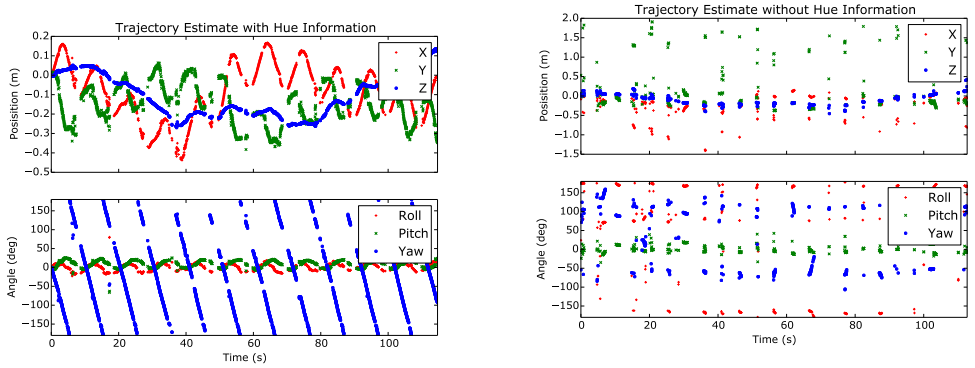
Figure 4.8: Monocular pose estimation of HippoCampus freely moving along a helical trajectory underwater.

4.4.1 Open-Loop Trajectory

A helical trajectory tracking experiment with a small radius was chosen to demonstrate the accuracy of our method. The μAUV HippoCampus freely follows a helical trajectory underwater by completing small circles in the XY plane while pitching downwards for the first half of the experiment, and then upwards when it reaches the bottom of the tank. This allowed it to cover nearly the full visual field of the camera within the test tank.

Figure 4.8 shows the 3D pose estimation results from the first half of the motion tracking HippoCampus going upwards. The coordinate system in the 3D graph is relative to the initial pose of the robot. The start and end positions are marked, as well as intermediate pose axes showing the orientation of the robot (the red axis with diamond tip is the long axis of the μAUV.)

Figure 4.9a shows a time trajectory of the estimated pose using the algorithm presented in this work using hue information to assign correspondences to markers. For comparison,



(a) Estimation with Hue information.

(b) Estimation without Hue information.

Figure 4.9: Comparison between pose estimation with and without hue information.

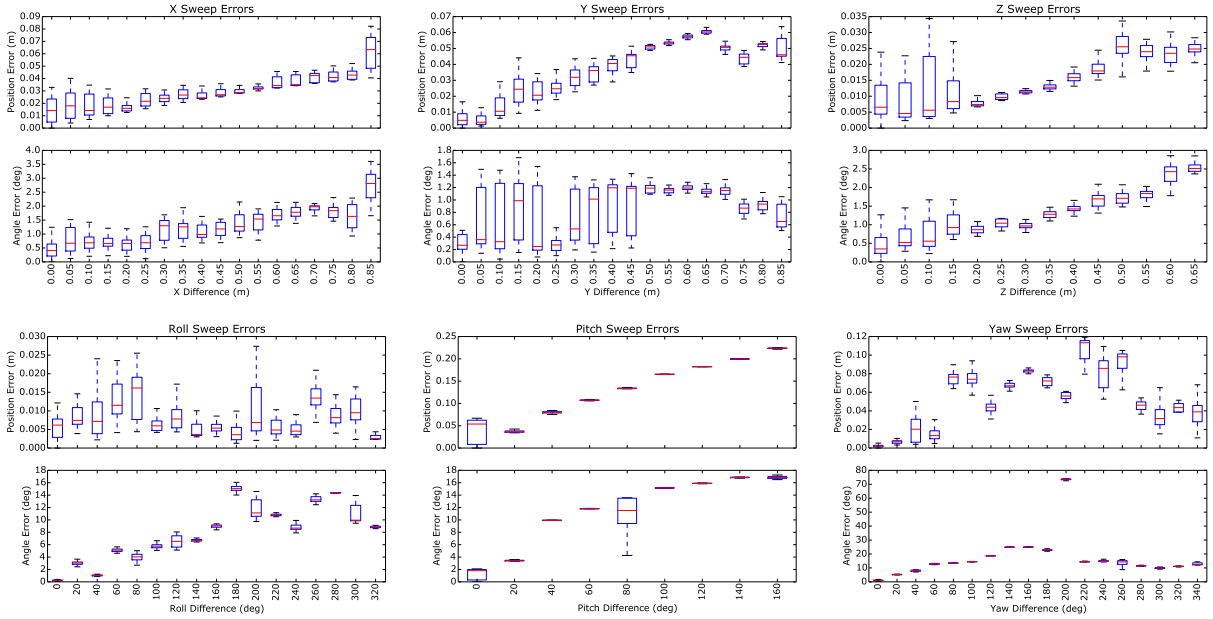


Figure 4.10: Distribution of errors in calibration experiment.

Figure 4.9b shows the result applying the original algorithm to the same image sequence using only the marker centers to determine a pose estimate.

4.4.2 Fixed Pose

The evaluation of the accuracy of the pose estimation method consisted of mounting the μAUV on the gantry in the experimental tank and collecting 30 images at each of a series of

fixed poses. These poses were grouped into six sets that swept the pose at even coordinate increments along only one world axis or angle, with the goal of viewing the μAUV at the extremes of the field of view of the camera. The same orientation was used for all of the axis sweep sets, and conversely the same position was maintained for each of the angle sweeps. The axes frames are defined in Figure 4.5, and Roll, Pitch, and Yaw are rotation about the X, Y, and Z axes respectively.

Figure 4.10 shows the errors in the pose estimation resulting from the fixed pose experiments. For each trial the first measured pose was set as the reference pose, and the subsequent relative poses were used to calculate the deviation from the expected pose based on the applied movement. Position errors are reported as the magnitude of the position error vector, and angle errors are reported as the magnitude of the angle in an axis-angle representation of the difference between the measured and expected orientation. This follows the convention for pose error quantification outlined in Sharma and D'Amico's work on visual pose estimation for satellites [68]. Note that the horizontal axes are aligned in each of the six data sets, but the vertical axes do not have the same ranges across the data sets.

4.4.3 Execution Time

The conversion of images from RGB to HSV color space approximately doubles the time spent in the blob detection phase compared to the original algorithm. Even though this was already the largest share of computational cost, the additional time did not prevent the pose tracker from being able to operate at the 30 Hz capture rate of the camera, thus did not significantly impact the desired application.

4.4.4 Power Usage

We compare the total localization system power including the beacons if deployed. The power required by the approach presented in this work was measured to be 600 mW for the markers (or 100 mW per marker) and 650 mW for the camera while active. In comparison, the approach based on electro-magnetic wave attenuation requires up to 100 mW per beacon. However, eight beacons would be required for the size of the presented tank, because four beacons need to be placed at two different depths [58]. The receiver power accounts to 100 mW. Hence, the total power used by the markers and cameras is 900 mW. Acoustic localization systems require significantly more power. The system presented in [24] which was applied within the same test environment as this work's system consisted of four 80 W acoustic transmitters.

To benchmark computational power requirements, the software pipeline was compiled and tested on an embedded compute platform that can easily be integrated onboard the HippoCampus. The platform used was the Hardkernel ODROID-U3, which has a quad core 1.7 GHz ARM Cortex-A9 processor and 2 GB of RAM. When on and idle, this platform draws 1.79 W of power. While capturing 1280 by 720 pixel RGB video at 21.6 Hz (the maximum rate achievable on the system), the device and camera together draw 3.16 W of

power (this includes the 900 mW for powering the camera). With the full pose estimation pipeline running, the system draws 4.16 W producing a pose estimate at the full 21.6 Hz frame rate. In comparison, running the AR Track Alvar[2] pose estimation pipeline draws a total of 4.28 W, while only producing a maximum pose estimate rate of 9.59 Hz.

Based on observations in [27] of a 12.8 Volt, 1.8 Amp-hour battery running the HippoCampus robot for 60 minutes under average operation, we can estimate 23 Watt of active power usage for processing and motion on the platform. The 4.16 Watt cost of running the tracking pipeline would reduce the operating time to 50 minutes under these assumptions.

4.5 Conclusion

4.5.1 Discussion

Figure 4.9 clearly shows that the Hue information produces a much cleaner estimate of the robot trajectory than only using marker positions for many more observed poses. This (expected) failure of the original method is largely due to the symmetry of the marker positions causing poor correspondence matches, and thus wildly varying pose estimates when not using Hue information. With further calibration and tuning of the parameters of the estimator, we believe that this augmented approach could achieve similar accuracy to the original method.

The trends in Figure 4.10 are largely due to human error in positioning the robot, and a likely poor calibration of the camera. Some of the trends in the angle sweeps may be due to particular orientations that are difficult to perceive and identify correct correspondences. Still, the fact that each box distribution represents 30 images of the robot in a rigid location provides some insight into the noise that the camera introduces. Future work will quantify the actual blob position noise given different camera conditions, and be used to inform a more accurate covariance estimate than the assumed single-pixel noise model presented previously.

Overall, our approach provides reasonable estimates of the relative poses. Combined with absolute positioning systems, e. g. acoustic methods, the system allows to localize possibly large fleets with member operating in proximity, whereby only few absolute positions need to be measured. Further, the presented method can be deployed in μAUV research to track vehicles in experimental tanks. While the range of the approach is limited by the visibility of the markers, we believe this approach could be ideal for precisely localizing and controlling μAUV systems at closer ranges.

4.5.2 Future Work

Our approach could benefit greatly from closed-loop control of the marker brightness and camera exposure settings to maximize marker visibility and Hue contrast. This will be necessary for the desired extension to teams of μAUVs, where a system must distinguish between

[2] `http://wiki.ros.org/ar_track_alvar`

several collections of markers on robots. While this approach incurred a high computational cost with the color space conversion, many cameras can provide raw data in YUV color space, which has a simple transform to HSV. Optimizing the blob extraction to operate on this type of data, and to take advantage of embedded hardware on board the robots could greatly reduce the computational cost of this method, and allow integration on-board the μAUVs. Finally, further empirical characterization of the noise properties of this approach would allow it to be fused with other position tracking methods based on IMUs and acoustics.

Chapter 5

Cooperative Inchworm Localization

5.1 Introduction

In this chapter we address the problem of multi-robot localization with a heterogeneous team of low-cost mobile robots. The team consists of a single centralized observer with an inertial measurement unit (IMU) and monocular camera, and multiple picket robots with only IMUs and Red Green Blue (RGB) light emitting diodes (LED). This team cooperatively navigates a visually featureless environment while localizing all robots. A combination of camera imagery captured by the observer and IMU measurements from the pickets and observer are fused to estimate motion of the team. A team movement strategy, referred to as inchworm, is formulated as follows: Pickets move ahead of the observer and then act as temporary landmarks for the observer to follow. This cooperative approach employs a single Extended Kalman Filter (EKF) to localize the entire heterogeneous multi-robot team, using a formulation of the measurement Jacobian to relate the pose of the observer to the poses of the pickets with respect to the global reference frame. An initial experiment with the inchworm strategy has shown localization within 0.14 meter position error and 2.18 degree orientation error over a path-length of 5 meters in an environment with irregular ground, partial occlusions, and a ramp. This demonstrates improvement over a camera-only localization technique that was adapted to our team dynamic which produced 0.18 meter position error and 3.12 degree orientation error over the same dataset. In addition, we demonstrate improvement in localization accuracy with an increasing number of picket robots.[1]

The size of a robot can greatly affect what it can do and where it can go. Advantages of small robots include increased accessibility and a wider range of capabilities such as crawling through pipes, inspecting collapsed buildings, exploring congested or complex en-

[1]This work originally published as "Cooperative Inchworm Localization with a Low-Cost Team," ICRA 2017 [54]. B. Nemsick developed the on-line localization filtering and fusion framework. A. Buchan incorporated multi-robot monocular pose tracking with hue information. A. Nagabandi developed the experimental methods, and all authors contributed to executing the experiments. This work was supported by the National Science Foundation under the National Robotics Initiative, Award 1427096.

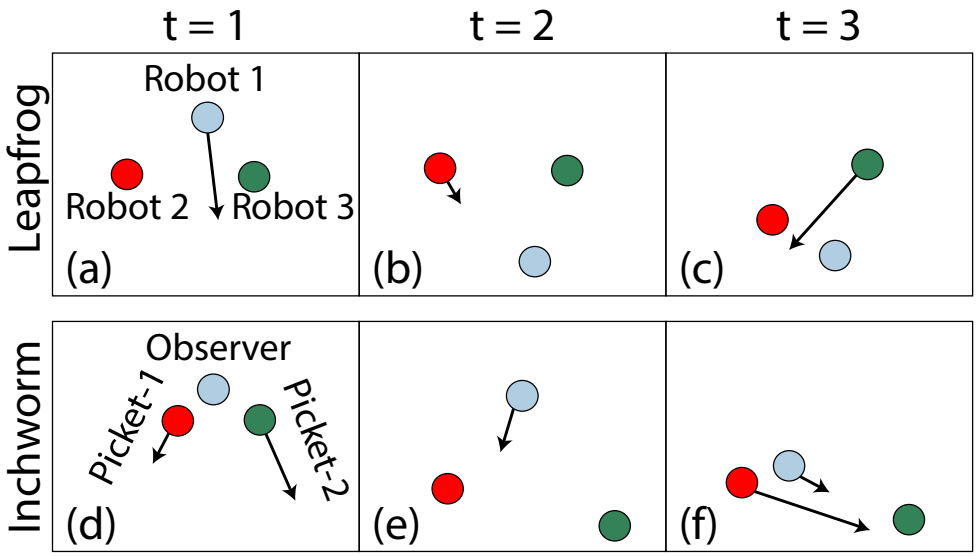


Figure 5.1: The above diagrams compare the leapfrog and inchworm strategies. Arrows are drawn to show motion that happens during a time step. In the Leapfrog method (a-c), all robots are the same type and at each time step one robot moves while the other two remain stationary. For example during (a) at t = 1, robots 2 and 3 remain stationary while robot 1 moves. This process repeats where the moving robot cycles at each time step. In our approach, the inchworm method, at least one robot remains stationary while two move. In addition the picket robots generally remain in front of the observer. For example during (d) the pickets move in front of the observer and during (e) the observer catches up to the stationary pickets. At (f) picket-1 and the observer both move leaving picket-2 stationary.

vironments, and hiding in small or inconspicuous spaces. However, these benefits also bring along challenges in the form of reduced sensing abilities, lower communication capability, limited computational resources, and tighter power constraints.

One way to overcome these limitations is to employ a heterogeneous team [25] of collaborative robots. This approach marks a design shift away from the traditional simultaneous localization and mapping (SLAM) ground robots that have expensive sensors and powerful processors, but less mobility in disaster environments. The goal is to have small, mobile, disposable robots with limited capabilities collaborate and share information to accomplish a larger task. Since each robot is expendable, reliability can be obtained in numbers because even if a single robot fails, few capabilities are lost for the team. Hierarchical organization and the idea of a heterogeneous team allows for robots to have different specializations, such as larger robots with higher computation power, smaller robots with increased maneuverability, and robots with different sensor modalities. Another advantage of a team of less capable robots, rather than one extremely capable robot, is that it allows sensing from multiple viewpoints and hence achieves a wider effective baseline. This is helpful for tasks such as surveillance, exploration, and monitoring. Furthermore, physically traversing an area

conveys much more information than simply looking at it from a distance. For example, an expensive scanner can scan the rubble of a disaster site from the outside, but cannot enter and inspect the inside. Knowledge that cannot be gained without physical presence includes detection of slippery surfaces, hidden holes, and other obscured hazards; these can completely incapacitate robots despite their state-of-the-art SLAM algorithms, expensive cameras, and complex laser range finders. Instead, these same hazards can be detected through sacrifice of highly mobile disposable picket robots that scout the area [29].

Localization is a central problem in many robotic applications. It is particularly important in collaborative situations where without position and orientation information of each robot, there is no global context in which to meaningfully share information between the robots. In this work, we focus on the localization problem and consider a heterogeneous multi-robot team consisting of two types of minimally equipped robots. A single, central, and more capable observer robot is equipped with a monocular camera and a 6-axis IMU consisting of a gyroscope and accelerometer. Multiple picket robots, which are expendable and less computationally capable, are equipped with no sensors other than 6-axis IMUs. A limited communication interface between the observer robot and individual picket robots is assumed. We consider a multi-robot team with a single observer and multiple picket robots in an unknown environment. We present a method for using a single EKF, which the observer uses to localize the entire multi-robot team, including itself, in six degrees of freedom (6-DOF) by fusing IMU measurements and relative pose estimates of the pickets. Relative pose estimation refers to the process of estimating the position and orientation of a picket's body frame with respect to the camera frame on the observer. This relative pose estimation is done using RGB LEDs that are mounted at known positions on the picket robots.

The datasets discussed in this work include one observer working together with two pickets to traverse given areas. Even with minimal sensors, the inchworm method is shown to work in dark environments with visual occlusions such as walls or obstacles, instances when line of sight between the robots is lost, and non-planar settings without external visual features or landmarks. The camera and IMU fusion approach employed by the inchworm method demonstrates improved performance over a camera only approach. In addition, we show that the localization accuracy of the inchworm method improves with an increasing number of picket robots.

5.1.1 Related Work

Existing localization strategies with stationary robots have been explored [25]. A stationary robot is defined as a robot that remains at rest while other robots move. Stationary robots and leapfrogging strategies build on the ideas from [25] and have shown promise in 3-DOF environments in [53][73]. These previous approaches have a stronger condition than our approach because they require two or three stationary robots at any given time. Our inchworm strategy relaxes these constraints to require only a single stationary robot at any given time, as shown in Figure 5.1.

A similar approach, cooperative positioning system (CPS), to inchworm is presented in [45]. The CPS approach focuses on 4-DOF (x,y,z,yaw) environments and partitions the robots into two groups, with each group consisting of at least one robot. Under the CPS system, the team alternates between which group moves: Either group A moves or group B moves. For the purposes of comparison we can consider group A to be the observer and group B to be the picket robots. An example of the CPS motion is shown in Figure 5.1 (d-e) where either the pickets move or the observer moves. Our inchworm strategy improves on CPS by allowing the observer and picket robots to move at the same time. An example of this is found in Figure 5.1(f) where both picket-1 and the observer move while picket-2 is stationary. Previous approaches and variants of leapfrogging strategies were focused on team dynamics with high redundancy where each robot produces relative pose estimates of all other robots. Our inchworm approach relaxes the sensor constraints to accommodate teams where only a single observer robot is required to have relative pose estimation capabilities. This leaves the picket robots with more flexibility and less computational burden.

Haldane et al. [29] use a heterogeneous team to detect slippery terrain by sending out a small picket robot and having it walk around the area of interest. The large robot is capable of accurately estimating it's own pose, and it uses an augmented reality (AR) tag on the picket robot to localize it. Then, features of the picket's motion are used to train a terrain classifier that is capable of detecting slippery terrain.

A follow-the-leader approach in [78] demonstrates a team composition similar to picket-observer. The leaders and children setup in [76] provides a relative localization scheme in 3-DOF; it assumes accurate localization of the leaders from an external source and localizes the children robots. This approach is extended in [77] to localize the leaders. The problem is subdivided into leader localization and then children localization. The localization of the leaders in [77] requires multiple leaders to maintain line of sight between each other. We extend the approach in [77] to jointly solve the leader and children localization problem without requiring multiple leaders.

A more recent approach [20] uses range-only sensors with a team of aerial vehicles for SLAM and builds on the limited sensor approach of [3]. These drones are equipped with on-board computers and expensive lasers. In contrast, our approach uses inexpensive and disposable picket robots in a 6-DOF environment.

Odometry-based propagation method have been successful in 3-DOF fusion architectures [25] [46]. However, in 6-DOF non-planar environments, wheel slippage causes systematic biases from encoders. Cell phone quality IMUs are a low cost alternative to wheel encoders in 6-DOF environments because they provide a motion model even under slippage. Extensive work in IMU-based propagation in visual-inertial systems has been explored in [70][51][38]. Additionally, monocular pose estimation has been explored in [21][9].

Many algorithms and approaches exist for multi-robot localization. Graph based approaches have been used [1][40], and the graph optimization algorithm in [1] relies on the locations of static landmarks and exploits the sparse nature of the graph. Existing EKF [67][47][46] or particle filter methods [37][16][60] demonstrate the capability of fusing data to provide accurate multi-robot localization.

The noted previous works have extensively and successfully explored multi-robot localization, but their experiments were conducted with access to significantly more capable robots, availability of GPS or beacons of known pose, 3-DOF settings with planar environmental assumptions and accurate wheel odometry, requirements of additional stationary robots, assumptions of light, and existence of landmarks or visual features. In this work, we relax these assumptions to localize a team consisting of a single observer robot and multiple picket robots. This is accomplished using an EKF approach with the inchworm strategy requirement of at least a single stationary robot at all times. IMU measurements are used for EKF propagation and relative pose estimates are used as an EKF update.

5.2 Methods

Algorithm 5.1 Cooperative Inchworm Localization (EKF)

Propagation: For each IMU measurement:

- buffer previous IMU measurements received from other robots
- propagate state and covariance for the team using the time-step, buffer and new IMU measurement (cf. Section III-B).

Update: For each camera image:

- identify RGB LEDs (cf. Section III-C).
- estimate the relative pose between the visible picket robots and the observer frame with P3P and Gauss Newton minimization (cf. Section III-C).
- propagate the state and covariance for the team using the time-step, and most recent IMU measurements
- perform state and covariance update for the team (cf. Section III-D, III-E).

Inchworm requirement: At least one stationary robot

The purpose of the multi-robot EKF is to localize all of the robot team's body frames with respect to a global reference frame. An overview of the EKF is provided in Algorithm 5.1 and can be described as follows: IMU measurements from both types of robots are used to propagate the state and covariance of the team with the same IMU motion model. RGB LEDs are placed with a known configuration on each picket robot such that images captured on the observer can be used to estimate the relative pose of the robots using [61][42] and Gauss-Newton minimization[21]. Relative pose is defined as the estimation of a picket's body frame (6-DOF position and orientation) with respect to the camera frame on the observer.

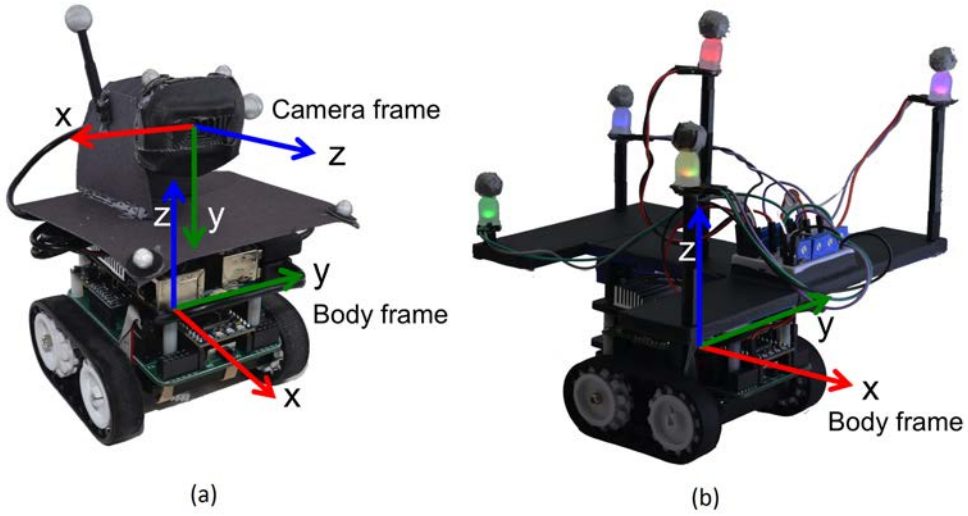


Figure 5.2: Coordinate frame overview for a sample team consisting of two robots. The observer, (a), is mounted with a camera and the picket, (b), with multi-color LED markers.

The coordinate frames of the team and example LED placement scheme are depicted in Figure 5.2. The relative pose estimates are subsequently used in the EKF update step.

A team movement strategy called inchworm is adopted, where the picket robots move ahead of the observer to scout and then the observer robot catches up. This movement strategy requires at least one stationary robot. This turn-based approach significantly reduces IMU dead-reckoning error and increases the robustness of the localization algorithm to temporary line of sight. An inchworm increment is a set of motions where the observer and picket robots all move at least once. An example inchworm increment is shown in Figure 5.1 (d-e).

We impose that a stationary robot does not propagate its corresponding states or covariances, thus bounding the uncertainty of the entire team. This is a strong assumption that can be violated by sensor failures (repeated incorrect pose estimates) and unobservable environmental phenomenon (slipping while wheels are stationary). In practice, we expect that scaling this approach to a team such that all robot positions can be redundantly observed will improve robustness to these failure modes. Importantly, this assumption enables the stationary robot to function as a temporary visual landmark and serves as a functional substitute to external visual features. Although external visual features are used in traditional visual odometry or visual SLAM systems, they are not consistently available in low light environments.

One benefit of a stationary picket robot is in situations of complete line of sight failure, where none of the picket robots are visible to the observer. In this case, a single future re-observation of a stationary robot, i.e. loop-closure, corrects the IMU dead-reckoning error of the non-stationary robots.

The following sections describe the EKF propagation and update steps in detail.

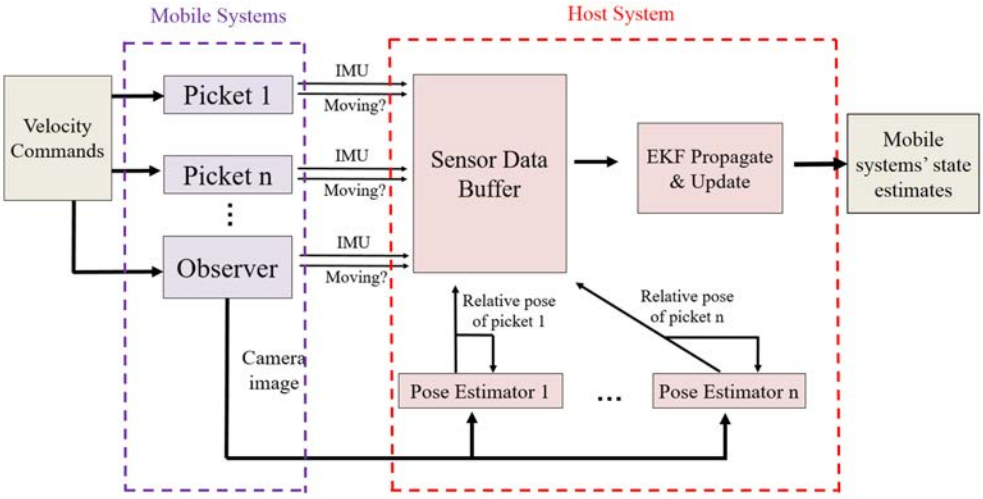


Figure 5.3: Block diagram of the asynchronous multi-robot team performing real-time cooperative localization algorithm. Asynchronous sensor data from the robots is sent over WiFi, sorted into a measurement buffer, and then used in the EKF propagate and update step. Currently, the host system is an external laptop.

5.2.1 State Vector

The EKF state and accompanying error-state vector stores the state of each single-robot in the multi-robot team. The state vector components with respect to i^{th} picket robot are:

$$\mathbf{x}_i = [{}^B_G\bar{\mathbf{q}}_i^T, {}^G\mathbf{p}_i^T, {}^G\mathbf{v}_i^T, \mathbf{b}_{i_g}^T, \mathbf{b}_{i_a}^T]^T \in \mathbb{R}^{16\times 1} \tag{5.1}$$

where ${}^B_G\bar{\mathbf{q}}_i^T \in \mathbb{R}^{4\times 1}$, is the unit quaternion representation of the rotation from the global frame $\{\text{G}\}$ to the body frame $\{\text{B}\}$, ${}^G\mathbf{p}_i, {}^G\mathbf{v}_i \in \mathbb{R}^{3\times 1}$ are the body frame position and velocity with respect to the global frame, and $\mathbf{b}_{i_g}, \mathbf{b}_{i_a} \in \mathbb{R}^{3\times 1}$ are the gyroscope and accelerometer biases.

The corresponding error-state components with respect to i^{th} picket robot are:

$$\tilde{\mathbf{x}}_i = [{}^G\tilde{\boldsymbol{\theta}}_i^T, {}^G\tilde{\mathbf{p}}_i^T, {}^G\tilde{\mathbf{v}}_i^T, \tilde{\mathbf{b}}_{i_g}^T, \tilde{\mathbf{b}}_{i_a}^T]^T \in \mathbb{R}^{15\times 1} \tag{5.2}$$

where ${}^G\tilde{\boldsymbol{\theta}}_i^T$ is the minimal representation from the error quaternion $\delta\bar{\mathbf{q}} \simeq [\frac{1}{2}{}^G\tilde{\boldsymbol{\theta}}^T, 1]^T$ [51] [38]. The non-quaternion states use the standard additive error model.

The observer robot is also a component in the EKF state and error-state vector:

$$\begin{aligned} \mathbf{x}_o &= [{}^O_G\bar{\mathbf{q}}_o^T, {}^G\mathbf{p}_o^T, {}^G\mathbf{v}_o^T, \mathbf{b}_{o_g}^T, \mathbf{b}_{o_a}^T]^T \in \mathbb{R}^{16\times 1} \\ \tilde{\mathbf{x}}_o &= [{}^G\tilde{\boldsymbol{\theta}}^T, {}^G\tilde{\mathbf{p}}_o^T, {}^G\tilde{\mathbf{v}}_o^T, \tilde{\mathbf{b}}_{o_g}^T, \tilde{\mathbf{b}}_{o_a}^T]^T \in \mathbb{R}^{15\times 1} \end{aligned} \tag{5.3}$$

where $\{O\}$ denotes the observer frame.

Combining the states in Equations 5.1, 5.2, and 5.3, the augmented EKF state vector and error-state vector with respect to the multi-robot team with n pickets becomes:

$$\begin{aligned} \mathbf{x} &= [\mathbf{x}_o^T, \mathbf{x}_1^T, \mathbf{x}_2^T, ... \ \mathbf{x}_{\mathrm{n}}^T]^T \in \mathbb{R}^{16(n+1)\times 1} \\ \tilde{\mathbf{x}} &= [\tilde{\mathbf{x}}_o^T \tilde{\mathbf{x}}_1^T, \tilde{\mathbf{x}}_2^T, ... \ \tilde{\mathbf{x}}_{\mathrm{n}}^T]^T \in \mathbb{R}^{15(n+1)\times 1} \end{aligned} \tag{5.4}$$

where n is the total number of picket robots.

5.2.2 IMU Propagation Model

The EKF propagation step occurs each time a new IMU measurement from any single-robot or a camera image is captured on the observer robot. The continuous dynamics of the IMU propagation model for a single-robot are [51][38]:

$$\begin{aligned} &{}^B_G\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}(\boldsymbol{\omega}){}^B_G\bar{\mathbf{q}}, \ {}^G\dot{\mathbf{p}} = {}^G\mathbf{v}, \ {}^G\dot{\mathbf{v}} = {}^G\mathbf{a} \\ &\dot{\mathbf{b}}_g = \mathbf{n}_{wg}, \ \dot{\mathbf{b}}_a = \mathbf{n}_{wa} \end{aligned} \tag{5.5}$$

where $\mathbf{n}_{wg}, \mathbf{n}_{wa}$ are Gaussian white noise vectors for the gyroscope and accelerometer respectively and

$$\lfloor \boldsymbol{\omega} \times \rfloor = \begin{bmatrix} 0 & -\boldsymbol{\omega}_z & \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z & 0 & -\boldsymbol{\omega}_x \\ -\boldsymbol{\omega}_y & \boldsymbol{\omega}_x & 0 \end{bmatrix}, \mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor \boldsymbol{\omega} \times \rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{5.6}$$

The discrete time linearized model and the error-state model are derived and discussed with detail in [51][38].

Critically, stationary robots receive no state or covariance propagation. This prevents IMU dead-reckoning drift from moving a temporary landmark and maintains a bounded covariance block pertaining to the stationary robot.

5.2.3 Relative Pose Estimation

Four (or more) RGB LEDs are placed at known configurations position on the picket robots to allow relative pose estimation on board the observer. Each picket robot receives a unique configuration with LEDs of various colors. Color and intensity thresholds are used to find the LED centroids, and these LED detections are passed into separate pose estimators (one pose estimator for each robot).

From the centroid detections, the approach from [21] is used to perform relative pose estimation. Pose correspondence is computed with the perspective-3-point (P3P) [61] [42] algorithm for each picket. Using different colors for the LEDs reduces the computational load by allowing the P3P correspondence search to search fewer possible configurations. Gauss-Newton minimization refines the initial solution from the P3P algorithm by minimizing reprojection error [21]:

$$P^* = \arg\min_P \sum_{<\mathbf{l},\mathbf{d}>\in\mathbf{C}} ||\pi(\mathbf{l}, \mathrm{P} - \mathbf{d})||^2$$

where P is pose estimate, $\mathbf{l}$ is the set of LED configurations, $\mathbf{d}$ is the set of LED centroids, $\mathbf{C}$ is the LED correspondences, and π projects an LED from $\mathbb{R}^3$ into $\mathbb{R}^2$ (camera image).

The pose estimate covariance ($\mathbf{Q}$) is calculated with the Jacobian ($\mathbf{J}$) from the Gauss-Newton minimization [21]:

$$\mathbf{Q} = (\mathbf{J}^T\Sigma^{-1}\mathbf{J})^{-1} \text{ where } \Sigma = \mathbf{I}_{2\times 2} \text{ pixels}^2 \tag{5.7}$$

5.2.4 Camera Measurement Model

In this section we describe how the relative pose estimates are used to compute the EKF update. We derive the residual and observation matrix that relates the relative pose estimates to the state vector as described in Section 5.2.1. The residual and the observation matrix are used to calculate the Kalman gain and correction.

An example overview of the multi-robot teams coordinates frames is shown in Figure 5.2. A static camera transform is defined as:

$$[{}^C_O\bar{\mathbf{q}}^T, {}^C_O\mathbf{p}^T]^T \in \mathbb{R}^{7\times 1} \tag{5.8}$$

With respect to a single visible picket robot, i, a relative pose estimate from the camera frame on board the observer is defined as:

$$\mathbf{z}_i = [{}^B_C\bar{\mathbf{q}}^T_{i_z}, {}^B_C\mathbf{p}^T_{i_z}]^T \in \mathbb{R}^{7\times 1} \tag{5.9}$$

In an EKF framework, a residual, $\mathbf{r}$, and a measurement Jacobian, $\mathbf{H}$ are used to compute the EKF update. The standard relationship between the residual and measurement Jacobian is:

$$\mathbf{r} = \mathbf{z} - \hat{\mathbf{z}} \approx \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n} \tag{5.10}$$

where $\mathbf{n}$ is noise. A prediction of the observation, $\hat{\mathbf{z}}_i$, is used to compute a residual in an EKF. This observation corresponds to a relative pose for each visible robot. Additionally, the quaternion states in $\mathbf{x}$ use the rotational error definition, $\delta\mathbf{q} = \mathbf{q} \otimes \hat{\mathbf{q}}^{-1}$ rather than the standard linear error, $\tilde{\mathbf{p}} = \mathbf{p} - \hat{\mathbf{p}}$.

To compute $\hat{\mathbf{z}}_i$, the state vector estimate is updated with the EKF propagation step. The poses of the picket robots are then converted from the global frame converted to the camera coordinate frame, {C}, in Equation 5.8 to match the relative pose estimate:

$$\hat{\mathbf{z}}_i = \begin{bmatrix} {}^B_C\hat{\bar{\mathbf{q}}}_i \\ {}^B_C\hat{\mathbf{p}}_i \end{bmatrix} = \begin{bmatrix} {}^B_G\hat{\bar{\mathbf{q}}}_i \otimes {}^G_O\hat{\bar{\mathbf{q}}}_O \otimes {}^O_C\hat{\bar{\mathbf{q}}}_O \\ {}^C_O\mathbf{R}\,{}^O_G\hat{\mathbf{R}}({}^B_G\hat{\mathbf{p}}_i - {}^O_G\hat{\mathbf{p}}_O - {}^C_O\mathbf{p}) \end{bmatrix} \tag{5.11}$$

where $\otimes$ represents quaternion multiplication.

The single-robot residuals with respect to each visible picket robot are calculated according to the definition Equation 5.10:

$$\mathbf{r}_i = \mathbf{z}_i - \hat{\mathbf{z}}_i = \begin{bmatrix} 2 \cdot \boldsymbol{\pi}({}^B_C\hat{\bar{\mathbf{q}}}_i^{-1} \otimes {}^B_C\bar{\mathbf{q}}_{i_z}) \\ {}^B_C\mathbf{p}_{i_z} - {}^B_C\hat{\mathbf{p}}_i \end{bmatrix} \tag{5.12}$$

where $\boldsymbol{\pi}$ is defined as $\boldsymbol{\pi}([\mathrm{q}_x, \mathrm{q}_y, \mathrm{q}_z, \mathrm{q}_w]^T)^T = [\mathrm{q}_x, \mathrm{q}_y, \mathrm{q}_z]^T$ and utilized as a small angle approximation for the orientation difference between $\mathbf{z}_i$ and $\hat{\mathbf{z}}_i$.

The i^{th} measurement Jacobian, $\mathbf{H}_i$, is calculated by applying small angle approximations and taking the partial derivatives of the i^{th} single-robot residual with respect to the error-state. The non-zero entries are shown below:

$$\begin{aligned} \mathbf{r}_i &\simeq \mathbf{H}_i\tilde{\mathbf{x}} \\ \mathbf{H}_i &= \begin{bmatrix} -{}^C_G\hat{\mathbf{R}} & \mathbf{0} & \mathbf{0} & \dots & {}^C_G\hat{\mathbf{R}} & \mathbf{0} & \mathbf{0} & \dots \\ {}^C_G\hat{\mathbf{R}}\lfloor({}^B_G\hat{\mathbf{p}}_i - {}^O_G\hat{\mathbf{p}}_O - {}^C_O\mathbf{p})\times\rfloor & -{}^C_G\hat{\mathbf{R}} & \mathbf{0} & \dots & \mathbf{0} & {}^C_G\hat{\mathbf{R}} & \mathbf{0} & \dots \end{bmatrix} \\ \tilde{\mathbf{x}} &= \begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_o & {}^G\tilde{\mathbf{p}}_o & {}^G\tilde{\mathbf{v}}_o & \dots & {}^G\tilde{\boldsymbol{\theta}}_i & {}^G\tilde{\mathbf{p}}_i & {}^G\tilde{\mathbf{v}}_i & \dots \end{bmatrix} \end{aligned} \tag{5.13}$$

where ${}^C_G\hat{\mathbf{R}} = {}^C_O\mathbf{R}\,{}^O_G\hat{\mathbf{R}}$ and $\lfloor\mathbf{q}\times\rfloor$ is the quaternion skew operator from Equation 5.6. The higher order and cross terms are dropped from $\mathbf{H}_i$ to satisfy the linear requirement of the EKF.

The states of all picket robots become correlated with the observer robot through the measurement Jacobian. This enables an individual pose estimate of a picket robot to improve the state estimate of each picket robot. The correlation is essential to localizing the observer robot because it is unable to observe itself directly from camera imagery.

5.2.5 EKF Update

From the camera measurement model the EKF update is performed. To utilize the standard equations, the overall measurement Jacobian is calculated by vertically stacking the single-robot measurement Jacobians from the camera measurement model in Equation 5.13:

$$\mathbf{H} = [\mathbf{H}_1^T, \mathbf{H}_2^T, \dots \mathbf{H}_{\mathrm{n}}^T]^T \in \mathbb{R}^{6n \times 16(n+1)} \tag{5.14}$$

Accordingly the measurements, $\mathbf{z}_i$, are stacked identically:

$$\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T, \dots \mathbf{z}_{\mathrm{n}}^T]^T \in \mathbb{R}^{7n \times 1} \tag{5.15}$$

The corresponding overall observation noise is calculated by diagonalizing the uncorrelated relative pose estimate covariances from Equation 5.7:

$$\mathbf{Q} = \mathrm{diag}(\mathbf{Q}_1, \mathbf{Q}_2, \dots \mathbf{Q}_n) \in \mathbb{R}^{6n \times 6n} \tag{5.16}$$

From Equations 5.14, 5.15, and 5.16, the procedure to update an EKF with quaternion states is described in [51] [38].

Table 5.1: Pose Tracker Comparison

	Faessler et al. [21]	Our System
Resolution (pixels^2)	752x480	640x480
Baseline Radius (cm)	10.9	10.6
LEDs/Robot	5	5
LED Type	Infared	RGB
$\approx$ error at 2 m depth	5 cm, 1-2 deg	5-8 cm, 1-4 deg

5.3 Results

5.3.1 Experimental Approach

We apply the localization technique described above to data collected from a team of three small, low-cost, mobile robots. The Zumy robot[2] is a decimeter-scale tracked robot running ROS on board a Linux computing system with networking and vision processing capabilities. The observer Zumy supports a Microsoft Lifecam 3000 camera with 640$\times$ 480 pixels^2 at 30 Hz, InvenSense MPU-6050 MEMS IMU at 30 Hz, and supports WiFi wireless communication. This robot is designed to be easily built from commercially available off-the-shelf parts for a total cost of $\approx$ \$350.

The robotic team consists of one observer and two picket robots shown in Figure 5.2. A Zumy robot with a camera serves as the observer, and to represent the inexpensive and less capable picket robots, we use Zumy robots without cameras. Each picket robot is outfitted with an LED "hat" so that it can be visually tracked by the observer robot. Infrared markers are also attached to each Zumy in order to obtain ground truth from a VICON motion capture system. The robots are manually driven for these datasets.

5.3.2 Planar Base Case

The baseline experimental task was a cooperative U-turn in planar 3-DOF with one observer and two pickets. The robots were manually driven in the dark. Although the dataset was recorded in a 3-DOF environment, the filter was not constrained with environmental priors. A direct comparison between the LED pose tracker system setup in [21] and our system setup is in Table 5.1.

In Figures 5.4 and 5.5, we show the resulting trajectories from the localization of all team members during this U-turn dataset and we compare to ground truth. We plot the results of using only one picket while discarding the measurements from other, and then the results of using both pickets. Note that the observer trajectory is not as smooth as that of picket-1 or picket-2, because the motion of the observer has unmodeled vibration effects that cause motion blur and temporary changes to the "static" camera transform.

[2] `https://wiki.eecs.berkeley.edu/biomimetics/Main/Zumy`

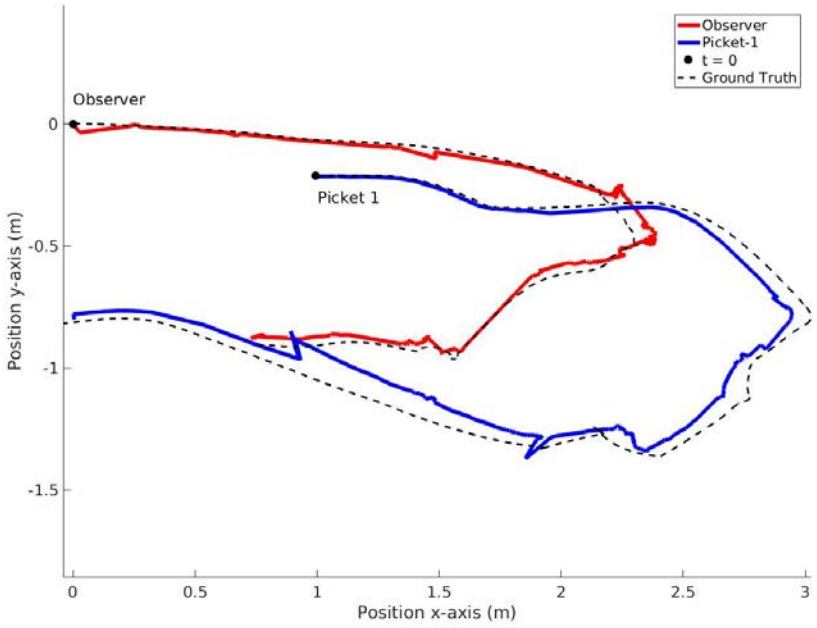


Figure 5.4: A plot of the XY projection of the team's pose estimates from the EKF along with the ground truth trajectories. Shown for the base case of the planar U-turn where a single picket is used to perform localization.

Table 5.2: Planar Drift Analysis

		Camera-only Two Pickets	Fusion One Picket	Fusion Two Pickets
Observer	x (cm)	-8.14	**0.67**	-0.80
	y (cm)	16.64	2.58	**-1.46**
	z (cm)	4.09	-5.42	**3.07**
	Angle (°)	9.33	1.89	**1.54**
Picket-1	x (cm)	-13.38	**-4.25**	-6.23
	y (cm)	28.97	-1.79	**-1.70**
	z (cm)	4.53	-7.91	**4.39**
	Angle (°)	4.72	2.64	**1.36**
Picket-2	x (cm)	**-1.35**	-	9.92
	y (cm)	25.40	-	**-5.65**
	z (cm)	4.56	-	**5.18**
	Angle (°)	4.56	-	**1.88**

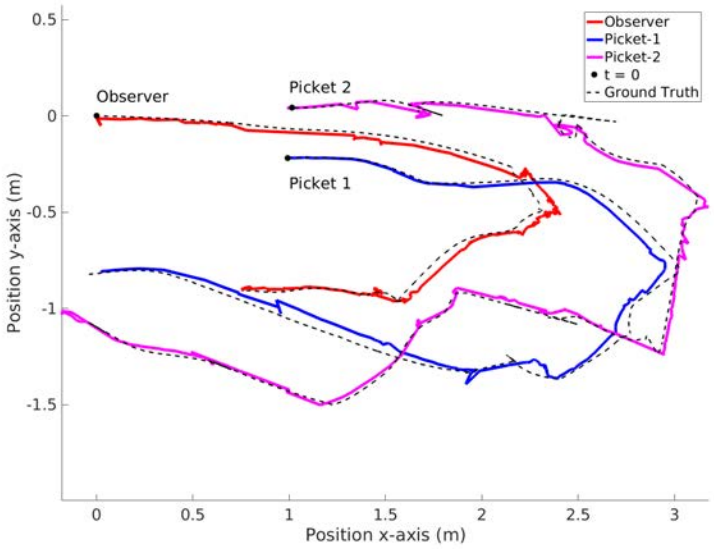


Figure 5.5: A plot of the XY projection of the team's pose estimates from the EKF along with the ground truth trajectories. Shown for the base case of the planar U-turn where both pickets are used to perform localization.

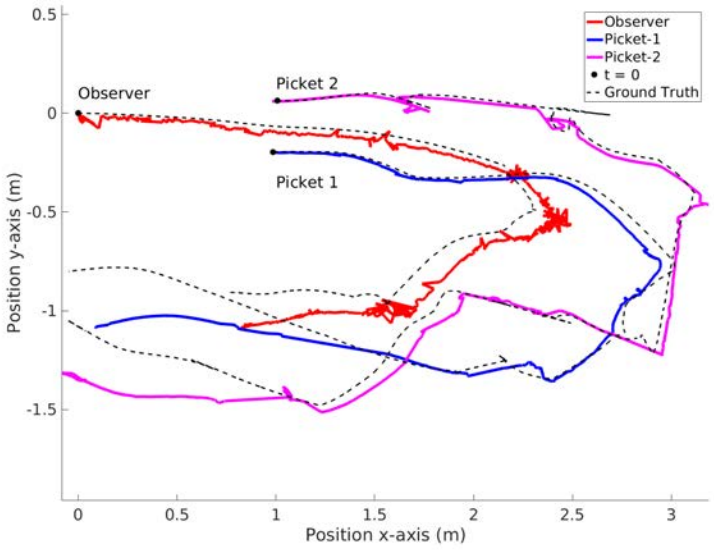


Figure 5.6: Camera-only approach: A plot of the XY projection of the team's pose estimates along with the ground truth trajectories, using a camera-only approach. Performs notably worse in yaw drift than the IMU-camera fusion approach shown in Figures 5.4, 5.5.

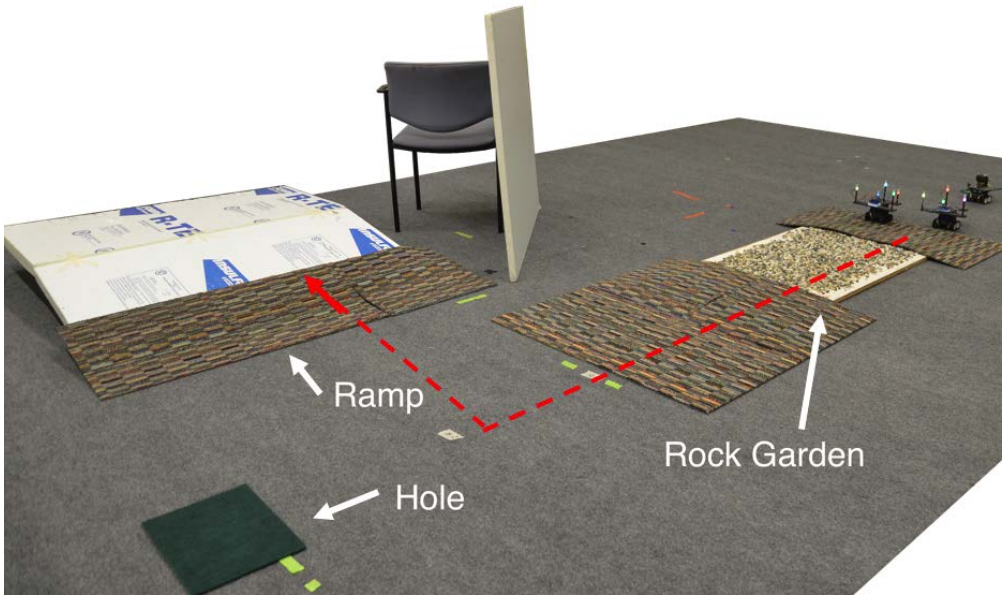


Figure 5.7: 6-DOF environment for testing: Robot team is on the right, rock garden is center-right), a "hole" is shown on the bottom-left, and the ramp is on top left.

This dataset consisted of 10 inchworm increments. An inchworm increment is defined as the minimal set of team motions where each robots has moved once. End position drift for using one versus two pickets is shown in Table 5.2. The angular drift, which causes propagates error into the future, for the two picket fusion case was less than the one picket case. Although unconstrained to a plane, the angular drift was almost exclusively in yaw. Performing right or left turns with the robot team introduces more rotational drift than forward or backwards motions. Without external features or global correction, the yaw errors persist until the end of experiment, but adding more picket robots helps to mitigate these effects. The jagged regions of the trajectory correspond to the observer and picket robots starting or stopping motion, and they are due to the LED mounts and the robots shaking during these transient motions.

A camera-only filtering approach was evaluated in Figure 5.6 as a baseline. The camera-only approach uses the same formulation of the measurement derived in Section 5.2.4 but without a motion model. This method performed significantly worse with four times as much yaw drift than the IMU plus camera fusion approach. Without the gyroscope, the inchworm localization performs significantly worse in orientation estimation.

5.3.3 Non-planar Terrain with Ramp

The second experiment was conducted in an environment featuring non-planar terrain, obstacles, and occlusions. The robots were manually driven in the environment shown in Figure 5.7. The 6-DOF non-planar dataset consisted of 10 inchworm increments: 3 for the

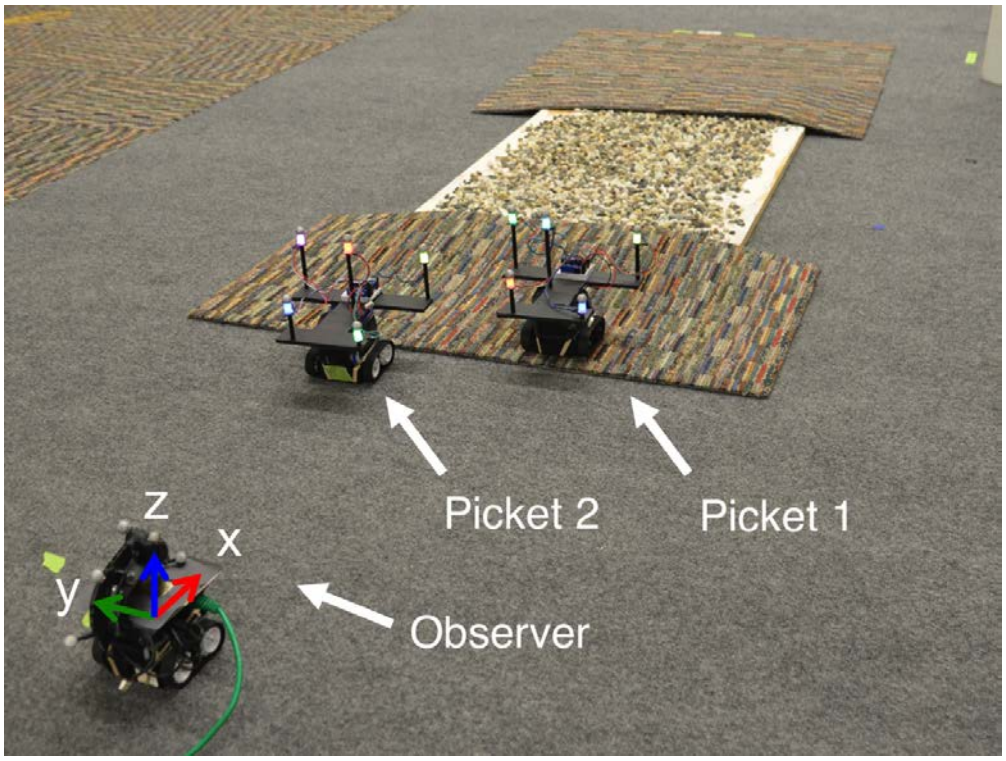


Figure 5.8: Starting position of the robot team with view of the rock garden section. The origin is defined as the starting position of the observer.

rock garden and 7 for the right turn and ramp. Temporary line of sight failure of both pickets occurred during the rock garden because the pose estimator failed to converge, as the observer was moving on the rocks. Wheel slippage also occurred during the rock garden section. After the rock garden, picket-2 was deliberately left behind to simulate a hole in the environment and a loss of a robot.

The ground truth trajectories and the EKF pose estimates of the dataset are shown in Figures 5.9 and 5.10. The end point drift analysis is shown in Table 5.3 with a comparison against a camera-only approach. The fusion approach outperformed the camera-only for the observer and picket-1. The most critical improvement is the orientation error of the observer, which persists without correction. Picket-2 traveled mostly in a straight line except during the rock garden, and the endpoint errors of both approaches are almost identical. The drift is predominantly in pitch for each robot. We hypothesize that this is due to a higher covariance in the estimate of the relative pitch angles between robots, as the distribution of markers on the picket robots is wider than it is tall from the perspective of the observer. Further investigation is necessary to be certain of the cause of this drift. Temporal plots with ground truth are in Figures 5.11 and 5.12.

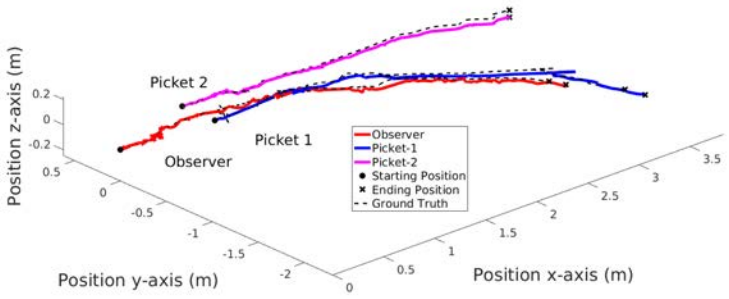


Figure 5.9: Ground truth trajectories of the multi-robot team are compared against the estimates of the EKF for the non-planar environment. Axes are scaled equally

.

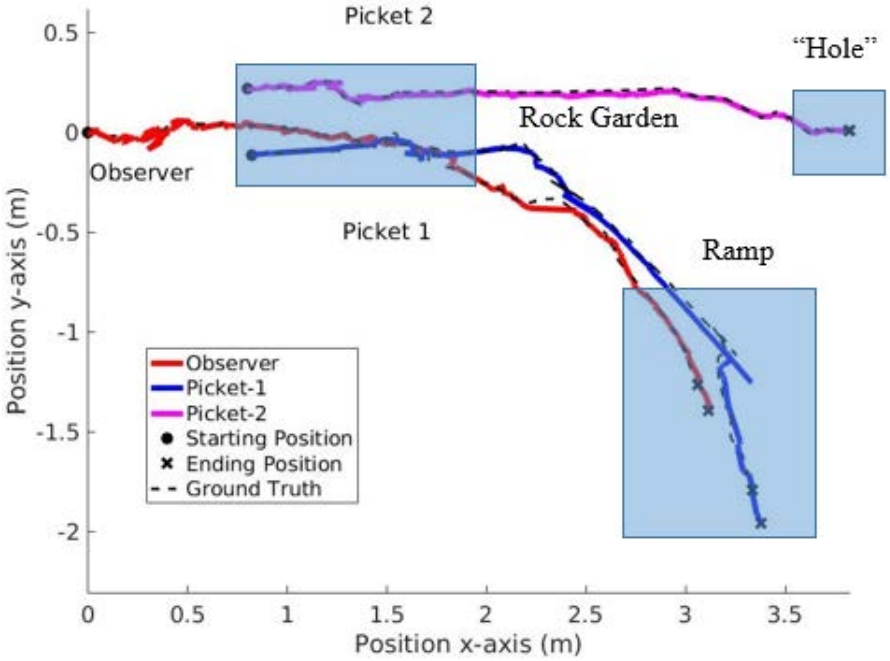


Figure 5.10: 2D projection of ground truth trajectories of the multi-robot team are compared against the estimates of the EKF for the non-planar environment.

Table 5.3: Non-Planar Drift Analysis

		Camera-only Two Pickets	Fusion Two Pickets
Observer	x (cm)	**-4.03**	-5.35
	y (cm)	13.36	**12.67**
	z (cm)	1.01	**0.04**
	Angle (°)	3.12	**2.18**
Picket-1	x (cm)	**-2.11**	-4.48
	y (cm)	17.9	**16.76**
	z (cm)	1.72	**0.10**
	Angle (°)	6.22	**4.29**
Picket-2	x (cm)	**0.28**	0.32
	y (cm)	-0.20	**0.15**
	z (cm)	**5.37**	**5.37**
	Angle (°)	**3.58**	3.59

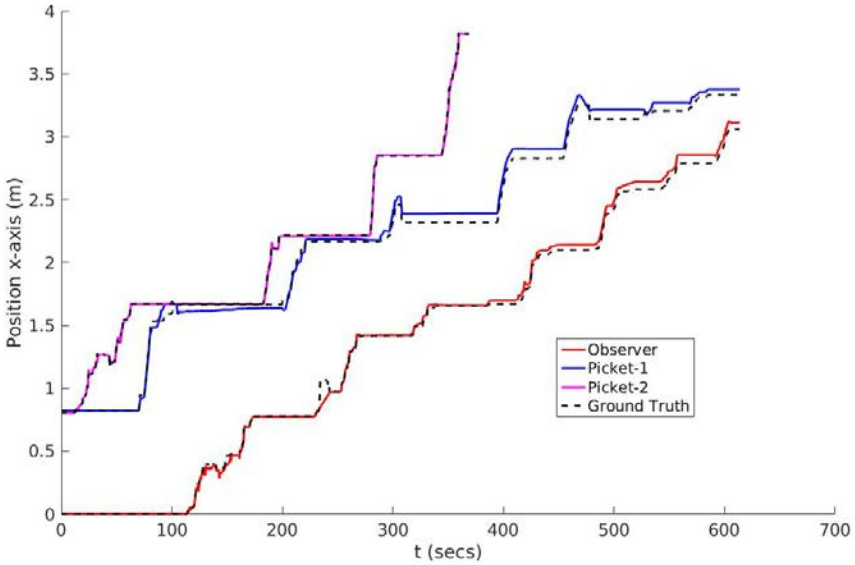


Figure 5.11: Position along the x-axis versus time.

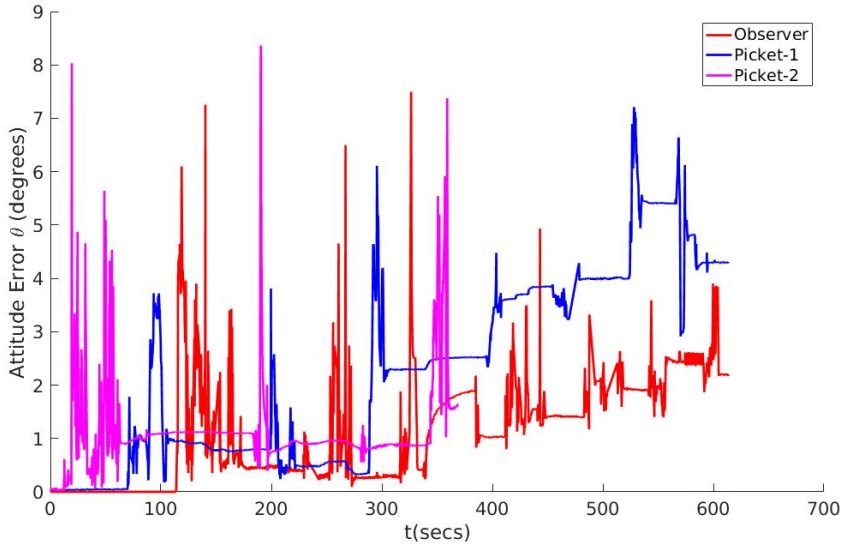


Figure 5.12: Orientation error versus time.

5.4 Conclusion

A heterogeneous team which consists of a single observer and multiple picket robots is able to navigate a visually featureless, unknown, non-planar environment as a unit, using only relative pose observations and IMU measurements to estimate the motion of the entire team. The IMU and camera fusion approach presented in this work has clear advantages over a simpler camera-only approach (Figures 5.4, 5.5, and 5.6) and its benefits outweigh the cost of having asynchronous communication. Although calibration of an IMU adds many complications, it is a natural choice for environments in which wheel encoders are unreliable. A camera-only approach heavily relies on line of sight at all times, which is restricting and potentially impractical to maintain. In addition, motion-model based approaches for EKF propagation allow for the rejection of errant camera pose estimates from faulty LED detections or P3P correspondence matching. Most importantly, with a camera-only approach, each inchworm increment has an associated positional and rotational drift in 6-DOF. The calibration of the IMU allows the fusion based approach of using the stationary robots' gravity vectors to reduce and bound the pitch and roll drift leading to drift in only 4-DOF. While the full state of the robotic team with respect to the environment is fundamentally unobservable without any environment sensing, this approach has been shown to be at least as effective as commonly used odometry methods in other SLAM work. Moreover, this approach works with noisy robot motion (due to dynamics or environment), and where wheel encoder approaches that assume no slippage would fail.

In the future, we will create exploration strategies for larger robot teams of more than 10 robots. With this increasing number of robots, autonomous control is far more effective

than manual driving. An advanced control scheme that factors in terrain, obstacles, and collaboration of robots will be developed for effective exploration in hazardous environments. It is clear that growing a linear state space to more robots will not scale well at some point. To grow to larger (10-100 robot) teams, methods that can effectively leverage sparse observations between team members should be explored. Factor-graph SLAM approaches are a popular and active field of research that would address this issue [14].

Chapter 6

Energetic Cost of Cooperative Range Finding

6.1 Introduction

In this chapter, we show preliminary work towards the addition of a simple scanning laser beam to the robot team used in Chapter 5 to enable the team to measure range via triangulation in overlapping camera Field of Views (FOVs) volumes. This range sensing capability is the last critical component of a full SLAM approach on low-cost hardware, as it would allow the robotic team to map the surrounding environment, and make decisions about how to proceed with exploration. A basic technique for scanning and then navigating an unknown environment is presented, with considerations for how the FOV size affects the rate at which a region can be scanned. We demonstrate the efficacy of this approach using a simulation developed in the V-REP Simulation Framework [65], which shows point cloud generation for a team of robots using only visual features. Finally we show progress towards miniaturizing the hardware for use on crawling millirobots.

This approach is motivated primarily by focusing on scanning hardware that is as low-cost, lightweight, and low-power as possible. Almost all range finding methods rely on emitting light or sound energy, and measuring the bearing and distance to a point of reflection of that energy in the environment. The choice of sensing modality affects trade-offs between resolution, energy used, and rate of volume scanned. In general, more energy can be used up to a point to collect information from a farther range, or take more measurements at a time. Sensor interference, ambient noise, scattering, and reflection inefficiencies all contribute to the fundamental limits of these approaches. Bulk scanning approaches, such as structured light or Time of Flight (ToF) depth cameras, also tend to be energetically wasteful, as they emit a significant amount of energy in regions that have already been sensed.

To understand these trade-offs at a first-principles level, we study an approach here that takes exactly one range measurement at a time. Akin to geographical surveying methods, we position robots with known relative pose so that their FOVs are overlapping, emit a

visible laser beam from one robot, detect the reflected point from at least two perspectives, and triangulate the position of the reflected point. Scanning the bearing of the laser beam through the full FOV of the co-located camera allows the system to collect a depth-map of the intersecting volume. Details such as reflectance angle and occlusion will reduce the effective sensing FOV, but this approach provides a simple proof-of-concept 3D range finding that can be extended and integrated with SLAM frameworks, all while being plausibly adaptable for use within the power, weight, size, and computational constraints of low-cost robots.

Section 6.2.1 presents a theory of moving through and scanning a volume with a team of robots with scanning lasers while respecting the constraints of Inchworm Localization outlined in Chapters 4 and 5. In order to provide an objective framework for estimating the energy used and information gained by a mapping approach, Sections 6.2.2 and 6.2.3 respectively define an energy and information model that can capture up to quadratic costs. Section 6.3 shows progress on a laser scanning mechanism and active marker frame that can be adapted for use on Zumy (and ultimately VelociRoACH-scale) robots to enable this cooperative range finding approach. This hardware is used to derive the operational energy parameters of the energy model. Section 6.4 shows the results of simulating a pair of robots moving and scanning a simple environment via the technique in Section 6.2.1. The derived energy and information models are used to produce a graph showing the trajectory of energy consumption and information gain of the team.

Ultimately, this chapter is mostly speculative towards a pipeline that could be used to compare various robotic platforms, sensors, sensing techniques, and exploration algorithms. The energy and informational models here have issues with generality and scalability, but mostly serve as placeholders to show what a future technique could address in terms of comparing the informational efficiency of mapping approaches.

6.2 Theory

Assuming the technique of monocular pose estimation established in Chapters 4 and 5, we have at least two robots with cameras with known relative pose. The pose estimation technique in Chapter 4 conveniently provides a software framework for identifying the bearing of points of light within a camera frame, so we can leverage this output to measure the bearing of reflected laser points in an environment. In Section 6.2.1 we first establish a navigation strategy for exploring an environment while sensing the environment via cooperative triangulation. We then outline a simple energetic model that can predict the approximate energy used for robots navigating with active markers while laser scanning with this technique in Section 6.2.2. Lastly, Section 6.2.3 provides a simple way of quantifying the information gained during a mapping trajectory using the volume swept out by scan rays. Together, these methods lay a theoretical framework foundation that could be leveraged by future work to implement a complete, on-line SLAM approach that can reason about energetic and information gain trade-offs.

6.2.1 Navigation

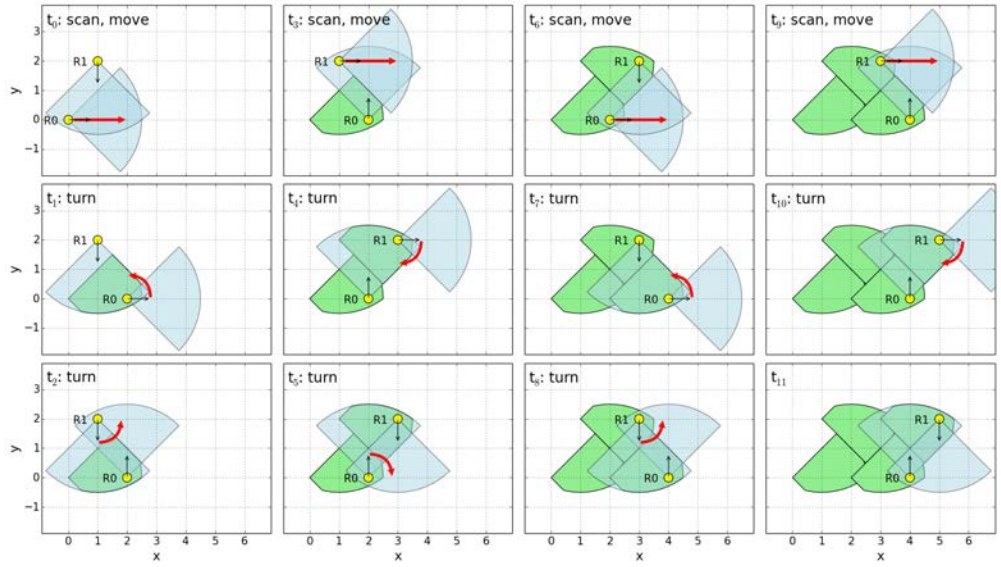
Given a laser emanating from one of the robots, our goal is to identify the most likely point relative to the robot frame to have generated the pair of bearing measurements. Hartley and Sturm [31] outline and compare several techniques for triangulating a point from two perspectives. For the proof-of-concept implementation in this work, we have implemented the simplest "Midpoint Method.' This method produces an estimate of the 3D coordinates of a detected laser reflection point by calculating the midpoint of the common perpendicular to the two rays from the camera centers to the detected point. In the case of no noise in the detected point bearing observations, this result is exact, and it is robust to small perturbations in the detected bearing. Any of the described methods in Hartley and Sturm may be used; we chose the Midpoint Method for its intuitive interpretation and ease of implementation.

An important condition for this technique to work is that the relative pose of the cameras is known. We can maintain this invariant using the inchworm technique developed in Chapter 5, with the additional condition that we only move into regions that have been scanned. Ideally, the entire volume surrounding the robots would be scanned allowing motion in the most convenient or informationally rewarding direction. However, common low-cost cameras have fixed diagonal FOV usually $\sim \frac{\pi}{2}$. We also assume that the resolution of the camera and power of the active markers and laser emitter imposes some maximum range of sensing, which we will approximate as a fixed radius. This motivates choosing a sequence of robot positions that first grow the sensed region in the desired direction of travel, moves one robot to the edge of that region, and then switches the roles of the robots so that the second one can be moved.

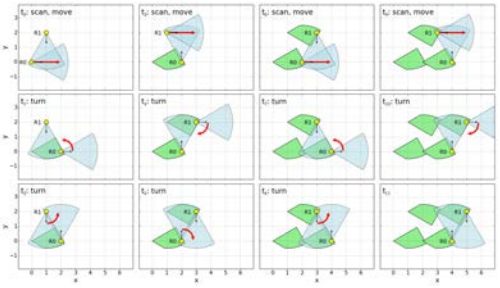
Figure 6.1 shows an overhead projection of this basic two-robot mapping and moving sequence, with differing Fields of View (FOVs). Each sequence shows the position of robots as yellow circles, their FOVs as blue arcs, and the growing mapped region as green segments. The goal is to travel as a team in the positive x direction.

Figure 6.1a shows one of the more efficient cases (in terms of total area scanned per step) when the FOV is $\frac{\pi}{2}$ radians. At t_0, the upper robot (R1) is facing in the negative y direction, with the lower robot (R0) at the origin, and left edge of R1's FOV facing in the positive x direction. This configuration nearly maximizes the scannable region, with some buffer to allow for the size of the robot, and uncertainties in pose and scan density. The robots scan at t_0, which results in the mapped area in t_1. R1 then moves to the right edge of this mapped region. By t_2, R0 has rotated to have R1 in its FOV. After a similar rotation by R1 to face left by t_3, the robots now have a new overlapping region ready for scanning.

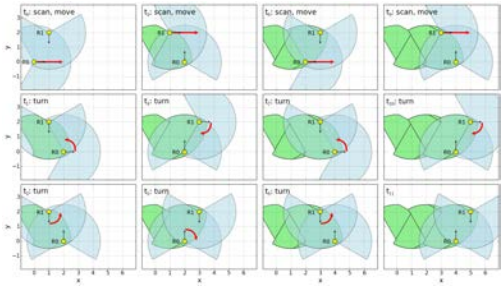
This configuration is symmetric to t_0, so the remaining plots show the repetition of the process, growing the mapped region in each of the upper frames (t_0, t_3, t_6, t_9). This process can continue as long as there is no obstacle blocking the path of the robots, which ostensibly will be sensed by the mapping process. In the case of an obstacle, it is straightforward to reorient the robots within the already scanned region such that they can start moving in a new direction.



(a) Mapping motion with $\frac{\pi}{2}$ FOV.



(b) Mapping motion with $\frac{\pi}{3}$ FOV.



(c) Mapping motion with $\frac{4\pi}{3}$ FOV.

Figure 6.1: Comparison of mapping motion with different Field of View.

Figures 6.1b and 6.1c show the same sequence of motions, but with smaller and larger FOVs respectively. With the smaller FOV, some of the region between the robots is left unmapped. If the only goal is to move quickly this may be an acceptable outcome. If coverage is desired, an intermediate rotation of R1 facing R0 can be used to map this region. For the larger FOV, we see that there is an initially much larger region scanned at the first step, but the region available for mapping in subsequent steps is not much larger than in Figure 6.1a. As was shown in Chapter 4, growing the FOV of a camera without maintaining spatial resolution can have a detrimental affect on effective range, so this further supports that a midrange FOV of $\sim \frac{\pi}{2}$ may be the most practical.

6.2.2 Energy Model

Considering the power/information metric space we used to categorize popular SLAM sensors and systems in Figure 1.2, we would like to develop a power cost and information gain metric for this proposed cooperative range finding approach that can allow us to compare the overall informational efficiency to popular and state of the art approaches.

The total energy used in Joules for cooperative range finding is a pragmatic cost metric, especially for resource constrained robots. To be useful for predicting actual performance, the model must be expressive enough to capture the relevant dominating physics of motion in the environment, be it viscous drag for fluid-navigating robots (aerial or underwater), moving on sloped ground, or any combination thereof. If used in an optimization framework for exploration, we should also have a reasonable expectation that the energy model can be designed as a robust, conservative upper bound for the actual energy used. For convenience in adapting to different types of robots, or even customizing for robots of the same type with variance in operational parameters between individual robots or variation over time, we would like that the energy model can be quickly derived in a data-driven manner (similar to the motivation in Chapter 3). Finally, for practicality in exploration optimization, where hundreds to tens of thousands of exploration trajectories may need to be evaluated quickly, we would like that the energy metric is lightweight computationally.

Given these motivating properties, here we evaluate a quadratic matrix form as a cost metric. It has the ability to capture important physics that cannot be expressed as a simpler weighted sum, such as moving on sloped ground, which requires a product of velocity and surface normal. Given a trajectory of a robot representative in the physics we are trying to learn and the real power consumption, it is straightforward to regress a quadratic power model that minimizes the error in predicted energy. Finally, calculating the quadratic power estimation metric is simply two matrix multiplications, and thus very fast to compute.

This model will take the basic matrix quadratic form of a cost function used by Linear-Quadratic Regulator (LQR) theory [15] to predict the instantaneous power $p(t)$ in Watts given a robot state $\mathbf{x}(t)$:

$$p(t) = \mathbf{x}^{\mathrm{T}}(t)\ M_p\ \mathbf{x}(t) \tag{6.1}$$

The constant quadratic cost matrix M_p allows us to capture physical phenomena proportional to squared state variables (such as viscous friction dominating losses in constant velocity wheel motion), or products of two variables (such as gravitational power being a product of velocity and elevation angle). In this chapter we will show an example power matrix for differential drive terrestrial robots derived from basic physics, but this form of power model is easily extensible to other kinds of robots such as the underwater μAUVs in Chapter 4. It could also be extended with an experimental data-driven technique as in Chapter 3 to learn the matrix parameters from motion and energy data.

Assuming the power model form above, the total energy used at a given point in time in Joules $e(t)$ is the integral of power:

$$e(t_i) = \int_0^{t_i} p(t)\,\mathrm{d}t \tag{6.2}$$

If we can assume that trajectories in this power space are a sequence of states $\{\mathbf{x}_i\} = \{\mathbf{x}(t_i)\}$, and assume piecewise constant velocity approximations between states (where $t_j - t_i = \Delta t_{ij}$ and $\mathbf{x}_j - \mathbf{x}_i = \Delta \mathbf{x}_{ij}$):

$$\mathbf{x}(t)|_{t\in[t_i,t_j)} = \mathbf{x}_i + \frac{t - t_i}{\Delta t_{ij}}\Delta \mathbf{x}_{ij} \tag{6.3}$$

This allows us to write the velocity between the two states:

$$\dot{\mathbf{x}}(t)|_{t\in[t_i,t_j)} = \dot{\mathbf{x}}_{ij} = \frac{1}{\Delta t_{ij}}\Delta \mathbf{x}_{ij} \tag{6.4}$$

Then the energy used on a constant velocity path e_{ij} is a function of the start and end states, as well as the time between them:

$$\begin{aligned} e_{ij}(\Delta t_{ij}, \mathbf{x}_i, \mathbf{x}_j) &= \int_{t_i}^{t_j} \mathbf{x}^{\mathrm{T}}(t) M_p \mathbf{x}(t)\,\mathrm{d}t \\ &= \frac{\Delta t_{ij}}{3}(\mathbf{x}_i^{\mathrm{T}} M_p \mathbf{x}_i + \mathbf{x}_i^{\mathrm{T}} M_p \mathbf{x}_j + \mathbf{x}_j^{\mathrm{T}} M_p \mathbf{x}_j) \end{aligned} \tag{6.5}$$

This makes it easy to predict the energy used by any trajectory of states X_i as just the sum of the energies used for each trajectory segment $x_{i,i+1}$, provided we know the Δt between them. Future work will need to evaluate whether or not this approximation holds for trajectories useful in cooperative range finding, and then if it is useful for improving the computation time required to estimate the energetic cost of a motion or sensing action. For now, in Section 6.4 we use numerical integration of the power model to calculate total energy used, as we have the energy state at each time step and are only computing one trajectory.

In our cooperative triangulation application, we assume that each robot uses energy in four main categories: computation, motion, active markers, and laser scanning. Thus the state $\mathbf{x}$ is a column vector with first and second order variables representing the pose and velocity of the robot, the motion and illumination of the laser scanner, and the illumination state of the LED markers. For convenience, we associate the computational energetic cost with a generic "robot" energy state $\mathbf{x}_r$. We now assume the energy used in the robot, marker $\mathbf{x}_m$, and scanning $\mathbf{x}_s$ states is independent of the other states, allowing us to rewrite the power cost as:

$$p(t) = \begin{bmatrix}\mathbf{x}_r^{\mathrm{T}}(t) & \mathbf{x}_m^{\mathrm{T}}(t) & \mathbf{x}_s^{\mathrm{T}}(t)\end{bmatrix} \begin{bmatrix} M_{p,r} & 0 & 0 \\ 0 & M_{p,m} & 0 \\ 0 & 0 & M_{p,s} \end{bmatrix} \begin{bmatrix}\mathbf{x}_r(t) \\ \mathbf{x}_m(t) \\ \mathbf{x}_s(t)\end{bmatrix} \tag{6.6}$$

This allows us to describe the substate elements and power cost matrices independently. To let us express quadratic and linear terms, each substate vector is a collection of one

or more time-varying scalar quantities $\mathbf{x}_{k,N}(t)$, concatated with a constant 1 in the last coordinate.

$$\mathbf{x}_k(t) = \begin{bmatrix} x_{k,0}(t) \\ \vdots \\ x_{k,n}(t) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{k,N}(t) \\ 1 \end{bmatrix} \tag{6.7}$$

This intuitively organizes each substate power matrix into quadratic (A_k), linear ($\mathbf{b}_k$), and constant (c_k) elements:

$$\begin{aligned} p_k(t) = \mathbf{x}_k^{\mathrm{T}}(t)\ M_{p,k}\ \mathbf{x}_k(t) &= \begin{bmatrix} \mathbf{x}_{k,N}^{\mathrm{T}}(t) & 1 \end{bmatrix} \begin{bmatrix} A_k & \frac{1}{2}\mathbf{b}_k \\ \frac{1}{2}\mathbf{b}_k^{\mathrm{T}} & c_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k,N}(t) \\ 1 \end{bmatrix} \\ &= \mathbf{x}_{k,N}^{\mathrm{T}}(t)\ A_k\ \mathbf{x}_{k,N}(t) + \mathbf{b}_k^{\mathrm{T}}\ \mathbf{x}_{k,N}(t) + c_k \end{aligned} \tag{6.8}$$

For the motion aspects we assume that our robots essentially exhibit differential drive dynamics in a 3D environment. Chapter 3 shows that legged robots can be used under these assumptions, and the Zumy robots used in Chapters 2 and 5 also exhibit noisy differential drive dynamics. In this modeling work, we assume that frictional losses and gravitational work dominate the energy expenditure. For reasons we explore at the end of this section, we also assume that robots move throughout the environment in constant velocity trajectories, with brief periods of acceleration near waypoints that do not contribute significantly to the bulk energy consumption. Using the same reference frame and variables as in Figure 3.5 is used, with the addition that the robot may be on a sloped plane, the scalar state variables of the robot energy state $\mathbf{x}_r(t)$ are

$$\mathbf{x}_r(t) = \begin{bmatrix} v(t) \\ \omega(t) \\ \hat{x}_z(t) \\ 1 \end{bmatrix} \tag{6.9}$$

where $v(t)$ is the linear velocity in m/s, $\omega(t)$ the angular velocity in rad/s, and $\hat{x}_z(t)$ is the projection of the robot x-axis on the world z-axis.

We suppose that the energy loss in left and right wheel, leg set, or track of a ground robot is proportional to its angular velocity squared $\dot{\alpha}^2(t)$ via a friction coefficient a_w. Using Equation 3.2 to convert between the robot and wheel velocity space, the wheel power $p_w(t)$ is written as:

$$p_w(t) = \begin{bmatrix} \dot{\alpha}_l & \dot{\alpha}_r \end{bmatrix} \begin{bmatrix} a_w & 0 \\ 0 & a_w \end{bmatrix} \begin{bmatrix} \dot{\alpha}_l \\ \dot{\alpha}_r \end{bmatrix}, \begin{bmatrix} \dot{\alpha}_l \\ \dot{\alpha}_r \end{bmatrix} = \begin{bmatrix} 1/r & -d/2r \\ 1/r & d/2r \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{6.10}$$

$$\Rightarrow p_w(t) = \begin{bmatrix} v(t) & \omega(t) \end{bmatrix} \begin{bmatrix} 2a_w/r^2 & 0 \\ 0 & a_w d^2/2r^2 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{6.11}$$

Assuming that the gravitational work done is simply the rate of potential energy gain $m_r\, g\, v(t)\, \hat{x}_z(t)$, where g is the magnitude of gravitational acceleration, we can write the full quadratic portion of the robot power matrix:

$$A_r = \begin{bmatrix} {}^{2a_w}/_{r^2} & 0 & {}^{m_r\, g}/_{2} \\ 0 & {}^{a_w d^2}/_{2r^2} & 0 \\ {}^{m_r\, g}/_{2} & 0 & 0 \end{bmatrix} \tag{6.12}$$

There are no linear components for the robot power matrix ($\mathbf{b}_r = \mathbf{0}$), but we do include the constant c_r to account for the power that the robot draws without moving. Thus the full robot power matrix $M_{p,r}$ is:

$$M_{p,r} = \begin{bmatrix} {}^{2a_w}/_{r^2} & 0 & {}^{m_r\, g}/_{2} & 0 \\ 0 & {}^{a_w d^2}/_{2r^2} & 0 & 0 \\ {}^{m_r\, g}/_{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_r \end{bmatrix} \tag{6.13}$$

The power state for the active markers only has one variable component $m(t)$ which is the sum of the red, green, and blue LED values ranging from 0 to 255 for each active marker. The $M_{p,m}$ matrix has only the linear component b_m, which is the scalar that scales an LED value to power consumption:

$$\mathbf{x}_m(t) = \begin{bmatrix} m(t) \\ 1 \end{bmatrix},\ M_{m,r} = \begin{bmatrix} 0 & {}^{b_m}/_{2} \\ {}^{b_m}/_{2} & 0 \end{bmatrix} \tag{6.14}$$

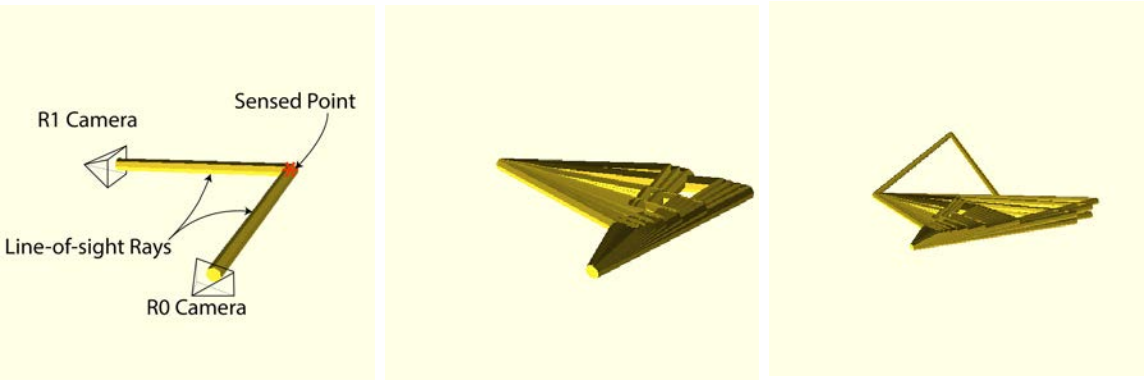
Finally, the laser scanning component of this energetic model captures the energy used to illuminate a projected laser beam, and energy used to steer the laser to different points in the camera FOV. These two components can be represented by the laser command $l(t)$ and the steering command $s(t)$. For now we assume these quantities can be calculated such that linear coefficients b_l and b_s can be used to scale to power consumed, such that the scanning power state and matrix are as follows:

$$\mathbf{x}_s(t) = \begin{bmatrix} l(t) \\ s(t) \\ 1 \end{bmatrix},\ M_{p,s} = \begin{bmatrix} 0 & 0 & {}^{b_l}/_{2} \\ 0 & 0 & {}^{b_s}/_{2} \\ {}^{b_l}/_{2} & {}^{b_s}/_{2} & 0 \end{bmatrix} \tag{6.15}$$

6.2.3 Information Model

In order to quantify and compare the informational value of different actions in a mapping context, we need an information gain metric that can be calculated from a mapping action. Traditional SLAM approaches usually define an information metric of a map $h(m)$ in terms of the entropy of the distribution representing the uncertainty in the map [72]

As a surrogate for a probabilistic information gain metric traditionally used in SLAM approaches, we propose that the volume swept out by cylinders along the line-of-sight scan rays can approximate information gained in a mapping context. This is a more conservative



(a) First scanned point of t_0, $h_v = 21.3$.

(b) Scan volume part way through t_0 scan, $h_v = 4650$.

(c) First scan volume after move and beginning second scan of t_3, $h_v = 10800$.

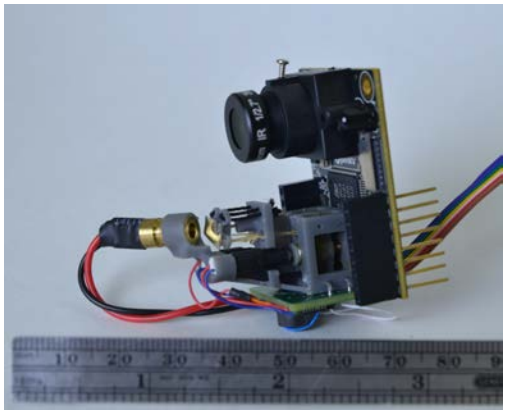
Figure 6.2: Sequence of scan volumes showing the increasing information gain metric.

bound than using cones, as it allows that scans of far away surfaces may have gaps of uncertain geometry between sensed points. This volume would be proportional to the number of voxels necessary to represent the scan geometry, and thus proportional to the number of bits needed to represent a binary occupancy space. For this work, a cylinder with the diameter of the robot is swept from each camera to a triangulated point in space. Figure 6.2 shows a sequence of these scan volume cylinders being added to a scan volume. Importantly, the union of new scans with the existing scan volume is taken at each time step so that only newly sensed areas contribute to the information metric. This geometric calculation is computationally expensive, and would not scale well to on-line implementations.

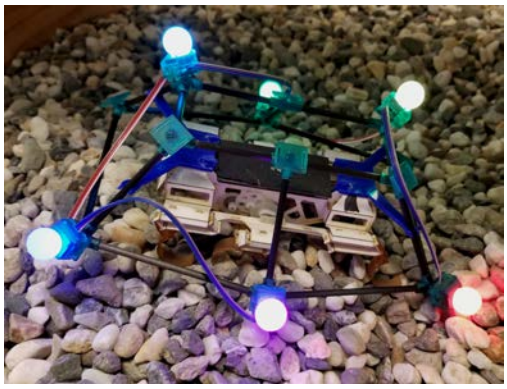
Most well-studied mapping representations would do better at approximating this information metric, and would likely be a fruitful direction for future implementation and study. Probabilistic extensions are also more natural with the addition of uncertainty measures at each sensed point. This would work particularly well with the recently published AtomMap by Fridovich-Keil et al. [23], which represents a scanned volume with a collection of spheres (or "Atoms") in the environment. This approach avoids volumetric quantization errors intrinsic to voxel-based approaches by representing surfaces implicitly as a Signed Distance Field (SDF) estimate at each scan point. All of these features would make it an excellent candidate for future work extensions of cooperative range finding, and ultimately SLAM with small scale robots.

6.3 Hardware

Figure 6.3 shows efforts to create and miniaturize scanning hardware that could be used on a bio-inspired millirobot. Figure 6.3a shows a scanning laser system attached to the OpenMV Cam M4 V2 [56]. This camera platform pairs an image sensor with a 180MHz ARM Cortex



(a) Miniature camera and laser scanner.



(b) Addressable active marker hat on VelociRoACH.

Figure 6.3: Pose estimation and scanning hardware.

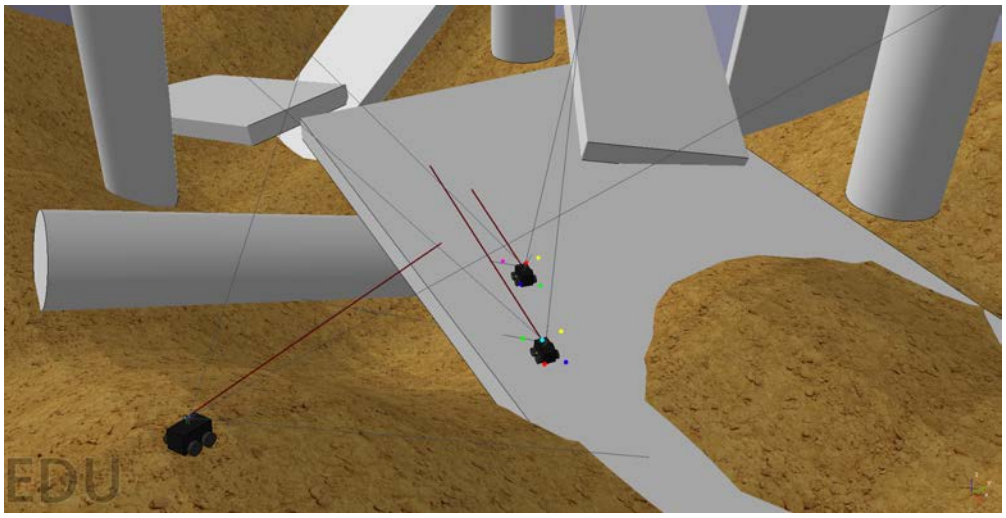
M4 embedded processor, and draws 462mW of power when active. The software environment is designed to easily port software developed with the OpenCV libraries, which was used almost exclusively in the blob extraction pipeline.

The laser scanning rig uses two small gear motors and worm drives to tilt a spring-loaded mirror positioned in front of a laser diode module. The range of motion of the mirror allows up to $\frac{\pi}{4}$ deflection from center in each axis. The scanning motors draw 297mW when moving the mirror, and the laser uses 14mW at maximum illumination.

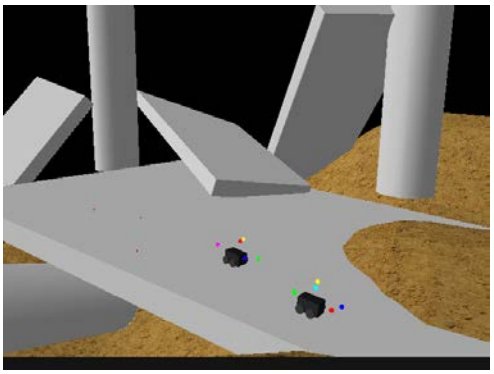
Figure 6.3b shows a lighter and reconfigurable version of the active marker hats used in Chapter 5. The hat frame uses carbon fiber spars and 3D-printed node connectors with exchangeable LED and diffuser clips that allows LEDs to be placed at any of 12 node locations. The LED clips are a series of programmable RGB modules, so that each LED can be assigned any color and brightness at run time. At maximum power (all six LEDs programmed to white) the markers can draw 1.5W. Using the default red, yellow, green, blue, cyan, magenta coloring at powers needed to establish a pose estimate at $\sim$1m, only 224mW are used.

6.4 Simulation

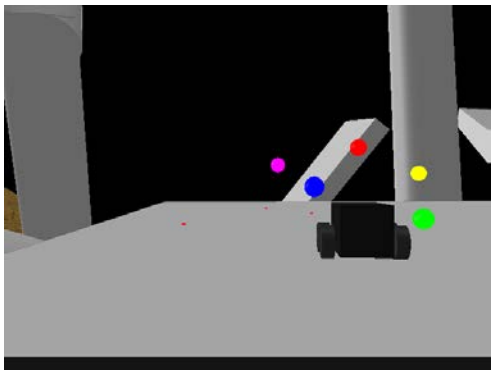
To test the cooperative range finding with vision only sensing, a simulation was developed in the V-REP Framework. This framework offers a selection of full physics engines, including Bullet and ODE, as well as API and plugin in support for ROS and Python interfaces. This simulation was designed to be as drop-in compatible as possible with the hardware as possible, in that it takes the same commands as a Zumy robot for commanded velocity, scan, and active marker colors, and produces the same sensor messages in terms of images, IMU readings, and odometry. The modularity afforded by the ROS/V-REP model allows components of the pipeline to be easily substituted. For example, the initial work from Chapter 2



(a) Team and environment.



(b) "Observer" perspective.



(c) "Picket" perspective.

Figure 6.4: Complex V-REP simulation environment.

with AR tags used for safe exploration was briefly re-implemented in V-REP before adapting the vision system to active markers. Much of this work for V-REP simulation was inspired by the Gazebo-based simulation available for the PixHawk development environment, which was used to test the initial adaptation of the colored active markers to the HippoCampus platform used in Chapter 4.

Figure 6.4 shows several simulated views in a complex rubble environment with three robots. Figure 6.4a shows an overhead perspective view of the whole team of robots in a rolling hill environment with pillar and slab concrete obstacles that could be expected in a collapsed building USAR situation. Figures 6.4b and 6.4c respectively show the views from the perspective of the rear observer robot camera, and one of the picket robots on the wall section. These perspectives clearly show the active markers on board the picket robots, which allow pose estimation between team members as per the techniques in Chapters 4 and 5. We can also see the dark red lines representing the rays of steerable lasers, and close

Table 6.1: Simulation parameters.

Environment	**Team**	**Robot**
• geometry • surface friction properties • visual textures • ambient lighting • particulate occlusion	• number of robots • initial pose of robots	• body geometry • motion modality • number and position of cameras • FOV of cameras and scanners • power model parameters

inspection of the observer and picket perspective will show the laser reflection points on the slab. Currently, the texturing and high polygon count of the environments lead to high pre-compute times and less-than-on-line execution rates for the simulation, but with some work on optimizing the environment for visual rendering efficiency we believe this framework could serve as a general-purpose simulator to produce sensor trajectories for arbitrary teams of robots in arbitrary environments.

This flexibility in simulation would be key to showing the expected performance of exploration and mapping techniques. Table 6.1 lists some examples of useful simulation parameters that could be explored with this simulation framework in the physical environment, configuration of the team of robots, and configuration of a single robot, supporting the generality of the simulation.

Figure 6.5 shows outline generic system software architecture allowing substitution of hardware and simulation. As of yet only the upper algorithmic modules (control, environment, and feature extraction) have been tested. Future work will hopefully leverage the simulation environment as convenient way of developing a full SLAM stack, as it is much easier develop complex environments, and scale to large heterogeneous teams of robots.

As Section 6.2.2 outlined, we can use a quadratic form to simulate the power used at a very fine granularity in simulation. For the purposes of estimating the power used by the simulation motion and scanning trajectory, the full power state $\mathbf{x}(t)$ can be calculated at each simulation step. Since the simulation proceeds at an exact regular time interval Δt, the total energy used is just the sum of the individual power costs multiplied by the time step:

$$e(t_N) = \Delta t \sum_{i \in N} \mathbf{x}_i^{\mathrm{T}} M_p \mathbf{x}_i \tag{6.16}$$

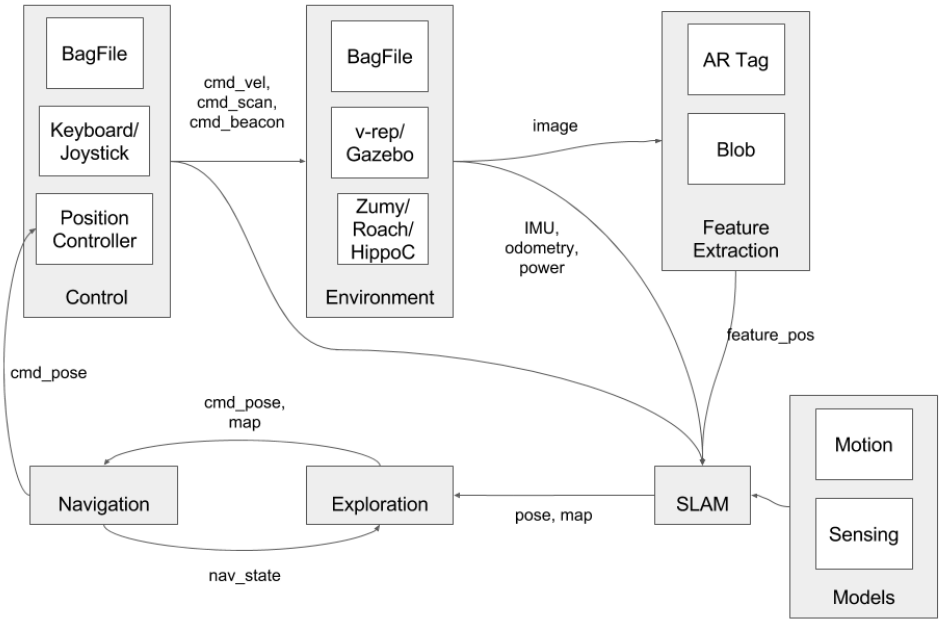


Figure 6.5: SLAM system overview.

The hardware developed in Section 6.3 was tested under basic motion, scanning, and marking conditions to arrive at the following values for the power matrix components:

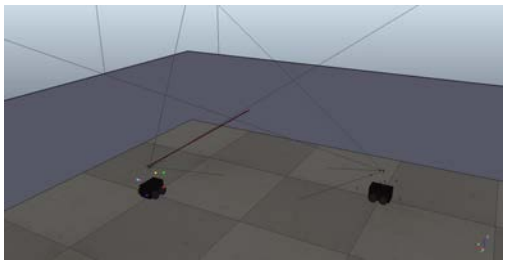
$$M_{p,r} = \begin{bmatrix} 14.0 & 0 & 1.68 & 0 \\ 0 & 0.035 & 0 & 0 \\ 1.68 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.56 \end{bmatrix} \tag{6.17}$$

$$M_{p,m} = \begin{bmatrix} 0 & 2 \times 10^{-4} \\ 2 \times 10^{-4} & 0 \end{bmatrix}, \; M_{p,s} = \begin{bmatrix} 0 & 0 & 0.25 \\ 0 & 0 & 0.025 \\ 0.25 & 0.025 & 0 \end{bmatrix} \tag{6.18}$$

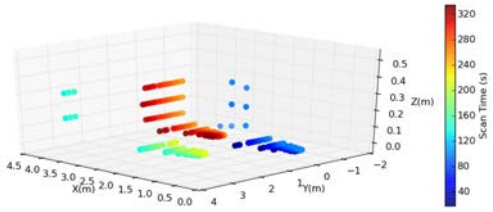
These values were used to calculate the power used trajectory in Section 6.5.

6.5 Results

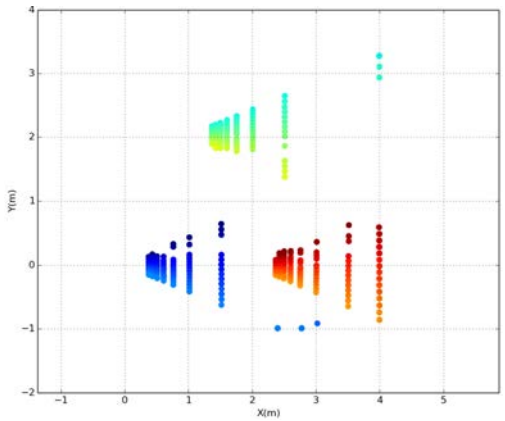
To test extracting the geometry of the environment, two simulated robots were placed in a static configuration on flat ground near a wall corner, as shown in Figure 6.6a. One robot is in the FOV of the other, nominally where the first robot would travel forward after mapping as in Figure 6.1. A portion of the floor and wall are in the intersection FOV of the two



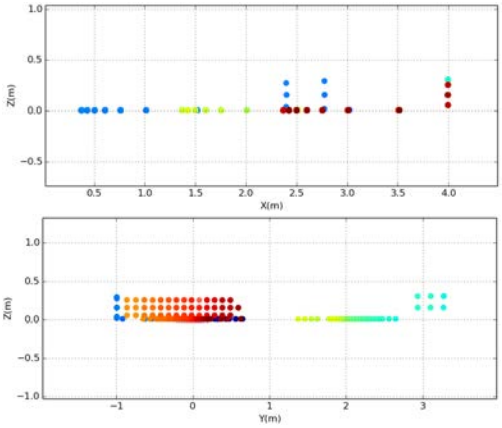
(a) Simulated environment with two robots.



(b) Resulting point cloud.



(c) Point cloud top view.



(d) Point cloud front and side views.

Figure 6.6: Cooperative mapping in simulated environment.

robots, shown as light gray lines in Figure 6.6a. The first robot has six colored markers that are used via the camera image exactly as in Chapters 4 and 5. Also depicted is the bearing of the simulated laser beam, which is only visible at the point of reflection in the images from each camera.

In the simulation, the active markers are first flashed to establish the pose between the robots. Then the laser is scanned in a regular grid with respect to the first robot's FOV in a boustrophedon pattern. For each simultaneous detection, the intersection of the bearing measurement from each camera is triangulated. Figure 6.6b shows the resulting point cloud, which clearly shows ground and walls in front of and to the right of the first robot. The points in Figures 6.6c and 6.6d use the same color scaling as Figure 6.6b to show the time at which each point was sensed. Due to image noise/aliasing and timing synchronization issues, some scan points are missed.

After initial triangulation was tested, the sequence of alternating scans and moves was implemented in simulation up to time $t6$ as shown in Figure 6.1. Using the energy model and information metric discussed in Section 6.2.2, the total energy used and information gained throughout the trajectory was simulated, shown in Figure 6.7. The graph shows the three distinct scans, with power use due to motion and active marker pose estimation

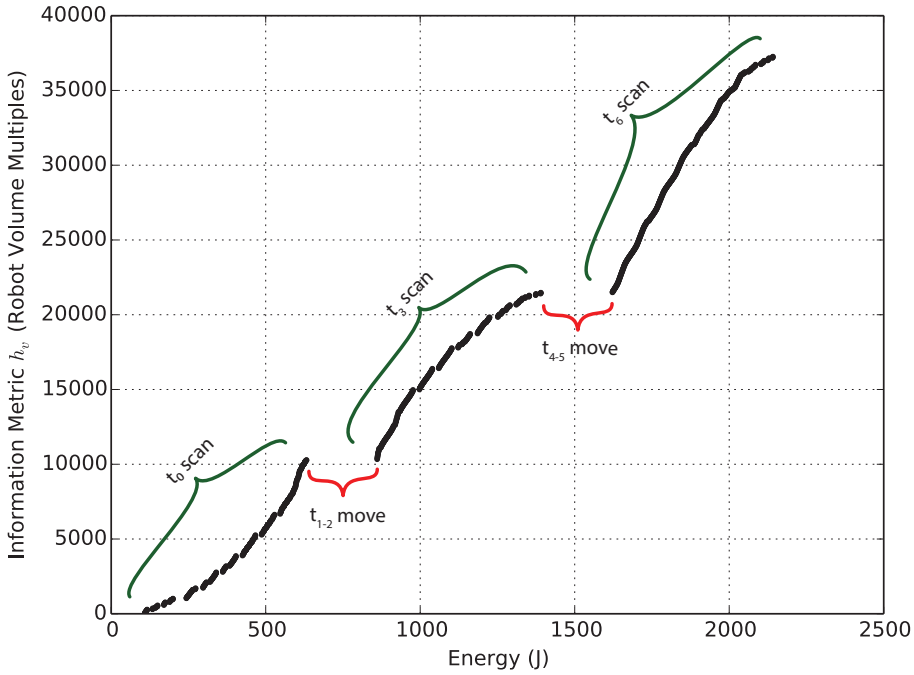


Figure 6.7: Information gain versus energy used for two simulated robots.

accounting for the gaps in between the dense scanning arcs. We see from this graph that the expected general increasing trend with diminishing returns in information can be seen in the second and third scans, but the first scan does not show this trend. For this simulation the increasing information trend at the end of the first scan is likely due to scanning from left to right, and measuring a region that is farther away from the robots but still in the FOV of both. This shows the weakness of this metric for quantifying the expected decreased certainty in farther scans, and highlights the need for further development of a probabilistic informational metric.

Some important takeaways from Figure 6.7 are that the energetic cost of sensing is greater than the robot platform motion. While this will not always be true, we expect that it would be the case for very resource constrained robots such as the VelociRoACH, and it is important that our model can express this. In the case of the simulation here based on the Zumy robots, this trend could be largely due to the slow frame rate of the simulated camera (20Hz), conservative wait times for inter-robot camera synchronization, and the high resting cost of the computational platform, but it does show that this approach can highlight the trade-offs between cost of sensing and cost of motion.

6.6 Conclusion

While there is still quite a bit of future work to be done to achieve a full SLAM solution at millirobot scales, we have at least shown a proof of concept implementation of point cloud generation via triangulation between two robots with cameras. With some consideration for maintaining line-of-sight when exploring, we have shown that the inchworm localization technique can be modified to allow two robot to scan an unknown region, and then make progress in a desired direction with this technique. A full physics simulation in V-REP has shown this triangulation technique to be viable with vision-only pose and bearing measurements. Hardware has been developed to enable vision and a steerable laser system at millirobot scales of power, weight, and size.

Chapter 7

Conclusion

7.1 Discussion

Throughout this thesis we have seen one approach to enabling low-cost bio-inspired millirobots with sensing and estimation techniques that could eventually be used in a cooperative SLAM application. On the dual fronts of modeling the motion and sensing properties of these robots, we have shown progress in modeling the behavior that informs their use in a SLAM pipeline.

First of all, we have shown that the low-cost design of the robots can allow for unique strategies in exploring terrains that could leverage disposability of robots for robust, safe exploration of dangerous environments. With regards to motion modeling, we have shown that even the highly dynamic and stochastic nature of the robot motion can be approximated mostly by differential drive assumptions. Future work can improve the results of learning motion models via data-driven techniques. With more complex models that can be designed to use on-board sensing for improved motion estimation.

Even with the challenges in modeling the motion of robots, we have developed a pose estimation technique that can compensate for the noisiness of this motion. By using small on-board monocular vision and active markers carried with the robots, a full 6-DOF pose estimate can be extracted between robots that have line of sight maintained. This technique can be effective in underwater environments, which simulate the feature-poor environment of a USAR deployment.

Ultimately, the monocular pose estimate can also be leveraged with additional laser scanning hardware to cooperatively triangulate surfaces in the environment, paving the way for a full mapping implementation. Initial simulations with two robots cooperatively localizing and triangulating portions of the environment. A basic energy model allows the energy used for each motion, sense, and scanning action to be simulated, which could serve as a starting point for a predictive model of SLAM active perception. We also showed a first approximation of an information gain metric relative to sensed environment volume, which would serve as the productivity metric in an exploration framework.

7.2 Future Directions

In order to see a usable and robot SLAM implementation on these robots, future work will need to proceed along several fronts.

For motion modeling, the data driven model identification framework could show promise with more complex models, and better on-line sensing integration. At the extreme of data-driven approaches would be unsupervised machine learning methods that assume no structure in the motion model. Progress in the complexity and accuracy of machine learning modeling techniques is continuous, but the especially non-linear and stochastic interaction of underactuated legged robots with unstructured terrain may still provide challenges in learning predictive models. Being able to learn and quantify probabilistic bounds on the uncertainty in dynamic modes of the robots will also be critical to effectively using the models in an estimation framework. After incorporation with energetic models of stochastic motion, exploration frameworks could effectively evaluate optimal choices in trajectories through a space to maximize information gain and minimize cost of motion in the process.

For monocular pose estimation, it would also be beneficial to develop more accurate uncertainty models of optically sensed poses. The current technique can offer a baseline Gaussian approximation of the pose uncertainty, but is based on a fairly rudimentary assumption of single pixel variance in sensing. This assumption does not truly capture the nature of noise in the pose estimates, and also does not deal with the problems brought on by separating adjacent or occluded markers. A data-driven approach of actually measuring the noise and bias in pose estimate using various configurations of markers and cameras would shed light on the structure of the noise in this method of pose estimation. The pose estimation technique would also benefit from more fine-grained power modeling in conjunction with the noise characterization. Understanding the trade-offs between energy used in the markers and the quality of the pose estimate should allow the energy to be modulated to the minimum necessary to achieve a desired quality of pose estimate.

There are several interesting venues for future work on cooperative localization and mapping. Obviously the scanning laser technique for cooperative triangulation needs to be tested on real systems to understand how well the assumptions about noise and energy apply to the non-idealities of the implementation. The simulation environment was originally developed to allow simulations of large-scale environments and teams of robots. It would be valuable to prove out automated exploration algorithms in simulation while varying parameters of the exploration team (camera FOV, marker size, laser power, speed, location, etc.). This would illuminate how sensitive the exploration technique is to each of these parameters, and allow optimization of the team and algorithm in terms of speed of mapping, accuracy, or power efficiency.

Appendix A

SLAM Sensors

This appendix details the data collected for the SLAM sensor and system comparison in Figure 1.2. Table A.1 shows the unique name used for each sensor, and the associated Part Number, Manufacturer and sensing technology category. The name corresponds with labeled points in Figure 1.2, and the technology category is shown as the disk color.

Table A.2 shows the physical characteristics of these sensors. They are the raw maximum bandwidth produced while sensing in bits per second, the power used while sensing in Watts, the sensor unit cost in US Dollars, the weight in grams, and the size in cubed millimeters. Where possible the bandwidth is calculated as the size in bits of a sensor read datagram multiplied by the sense rate. In cases where an exact datagram format was not available the bits needed to represent a sensor reading was approximated as $\left\lceil \log_2 \left(\frac{\text{range}}{\text{resolution}} \right) \right\rceil$.

Table A.3 lists all of the associated website URLs and publication references used to collect these data.

Table A.1: SLAM Sensor Technologies.

Name	Part Number	Manufacturer	Technology
SF-MPU-9250	MPU-9250	InvenSense	IMU
AMS-AS504	AS5040	AMS AG	Encoder
ALM-A3213	A3213	Allegro MicroSystems	Encoder
USD-E4T	E4T	US Digital	Encoder
PO-2591	Pololu2591	Pololu	Encoder
SH-GP2Y0A02	GP2Y0A02	Sharp	Triangulation
ST-VL6180X	VL6180X	STMicroelectronics	IR-ToF
Low-Power-SL	Low Power SL	Mertz	Structured-Light
PL-LIDAR-Lite	LIDAR Lite	PulsedLight	ToF
MI-SR4500	SR4500	Mesa Imaging	ToF
MI-SR4000	SR4000	Mesa Imaging	ToF
SK-DS325	DS325	SoftKinetic	ToF
NUX-Pico	CamBoard Pico S	NimbleUX	ToF
RS-RPLIDAR	RPLIDAR-A2	RoboPeak	Scanning-2D
HK-30LX	UTM-30LX	Hokuyo	Scanning-2D
SI-LM100	LM100	Sick	Scanning-2D
VD-Puck	Puck LITE	Velodyne	Scanning-3D
Line-Camera	TSL1401CL	TAOS	Gray-Rolling
OV-GG	OV6211	OmniVision	Gray-Global
OV-RC	OVM9724	OmniVision	Color-Rolling
CMUCam5	CMUcam5 Pixy	Charmed Labs	Color-Rolling
MS-LifeCam	T3H-00011	Microsoft	Color-Rolling
PG-CG-1	FMVU-03MTC-CS	PointGrey	Color-Global
PG-CG-0	FL3-GE-03S1C-C	PointGrey	Color-Global
PG-XB3	BBX3	PointGrey	Stereo
LM-LeapMotion	Leap Motion	Leap Motion	Stereo
PG-08S2	BB2-08S2	PointGrey	Stereo
PG-HICOL	LD2-HICOL-KIT	PointGrey	Spherical
MS-Kinect	L6M-00001	Microsoft	DepthImage
MS-XB1-Kinect	B00INAX3Q2	Microsoft	DepthImage
Laser-Scanner	Laser-Scanner	Buchan	SLAM
V-Roach-Cam	V-Roach-Cam	Bermudez	SLAM
SB-VI	VI-Sensor	SkyBotics	SLAM
Backpack	Backpack	Zakhor	SLAM
V-Roach	VelociRoACH	Haldane	SLAM

Table A.2: SLAM Sensors Properties.

Name	Bandwidth bit/s	Power W	Cost USD	Weight g	Size mm^3
SF-MPU-9250	2.40×10^5	9.25×10^{-3}	5.44	4.37×10^{-2}	9.00
AMS-AS504	1.00×10^5	5.28×10^{-2}	5.00	5.57×10^{-2}	96.7
ALM-A3213	4.00×10^3	8.25×10^{-4}	1.30	1.42×10^{-2}	8.64
USD-E4T	6.00×10^6	0.125	20.0	19.0	5.85×10^3
PO-2591	3.00×10^4	7.92×10^{-2}	7.00	1.50	222
SH-GP2Y0A02	400	0.165	9.75	5.00	1.83×10^4
ST-VL6180X	240	4.76×10^{-3}	25.0	0.500	13.4
Low-Power-SL	1.11×10^9	6.50	800	500	2.00×10^6
PL-LIDAR-Lite	1.20×10^3	0.650	150	22.0	3.84×10^4
MI-SR4500	1.01×10^7	12.0	4.65×10^3	850	6.16×10^5
MI-SR4000	6.08×10^6	9.60	4.29×10^3	500	3.21×10^5
SK-DS325	1.36×10^9	2.50	260	-	8.79×10^4
NUX-Pico	6.91×10^6	2.00	690	-	9.08×10^3
RS-RPLIDAR	3.20×10^4	1.15	500	340	1.84×10^5
HK-30LX	6.91×10^5	8.00	1.10×10^3	370	3.13×10^5
SI-LM100	2.70×10^5	12.0	5.00×10^3	1.10×10^3	1.63×10^6
VD-Puck	7.20×10^6	8.00	5.60×10^3	590	5.88×10^5
Line-Camera	3.07×10^4	1.25×10^{-2}	8.77	5.00×10^{-2}	33.8
OV-GG	5.76×10^8	0.238	136	3.46×10^{-2}	20.2
OV-RC	8.29×10^8	0.154	5.08	0.500	35.8
CMUCam5	1.23×10^9	0.700	60.0	27.0	1.80×10^4
MS-LifeCam	2.46×10^8	0.550	40.0	5.00	9.00×10^3
PG-CG-1	6.50×10^8	1.00	300	37.0	3.65×10^4
PG-CG-0	1.11×10^9	2.50	600	38.0	2.52×10^4
PG-XB3	1.42×10^9	4.00	4.02×10^3	505	4.28×10^5
LM-LeapMotion	9.22×10^7	1.00	80.0	32.0	1.28×10^4
PG-08S2	1.15×10^9	2.50	4.02×10^3	342	2.68×10^5
PG-HICOL	4.25×10^8	11.2	430	1.19×10^3	1.55×10^6
MS-Kinect	3.69×10^8	2.25	150	547	8.11×10^5
MS-XB1-Kinect	1.49×10^9	17.6	43.1	1.29×10^3	6.42×10^5
Laser-Scanner	7.37×10^7	0.889	30.0	11.4	1.44×10^5
V-Roach-Cam	1.48×10^8	3.03	210	46.0	1.92×10^5
SB-VI	1.73×10^8	10.0	4.62×10^3	130	5.41×10^5
Backpack	7.50×10^7	125	2.00×10^4	1.50×10^4	2.83×10^7
V-Roach	1.64×10^5	2.22	200	45.0	1.92×10^5

Table A.3: SLAM Sensor References.

Name	References
SF-MPU-9250	https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/ https://www.sparkfun.com/products/13762
AMS-AS504	https://ams.com/kor/content/download/1285/7214/file/AS5040_Datasheet_EN_v2.pdf
ALM-A3213	http://www.allegromicro.com/~/media/Files/Datasheets/A3213-4-Datasheet.ashx
USD-E4T	http://www.usdigital.com/products/encoders/incremental/rotary/kit/E4T
PO-2591	https://www.pololu.com/product/2591/specs
SH-GP2Y0A02	https://www.pololu.com/product/1137/specs
ST-VL6180X	https://cdn.sparkfun.com/datasheets/Sensors/Proximity/DM00112632.pdf https://www.pololu.com/product/2489
Low-Power-SL	Mertz [49] http://www.picopros.com/article/battery-life-testing-microvision-showwx http://www.ptgrey.com/flea3-03-mp-color-gige-vision-sony-icx618-camera%E2%84%A2
PL-LIDAR-Lite	http://velodynelidar.com/lidar/hdlproducts/vlp16.aspx
MI-SR4500	http://www.mesa-imaging.ch/products/sr4500/
MI-SR4000	http://www.mesa-imaging.ch/products/sr4000/
SK-DS325	http://www.softkinetic.com/Store/ProductID/6 http://www.softkinetic.com/Portals/0/Download/WEB_20120907_SK_DS325_Datasheet_V2.1.pdf
NUX-Pico	http://pmdtec.com/html/pdf/PMD_RD_Brief_CB_pico_71.19k_V0103.pdf
RS-RPLIDAR	http://www.robotshop.com/media/files/pdf/datasheet-rplidar.pdf
HK-30LX	http://www.hizook.com/blog/2009/03/03/new-sick-laser-range finder-lms-100-designed-compete-hokuyo-utm-30lx
SI-LM100	http://www.hizook.com/blog/2009/03/03/new-sick-laser-range finder-lms-100-designed-compete-hokuyo-utm-30lx
VD-Puck	http://velodynelidar.com/vlp-16-lite.html
Line-Camera	http://www.mouser.com/catalog/specsheets/TSL1401CL.pdf https://www.digikey.com/product-detail/en/ams/TSL1401CL/TSL1401CLCT-ND/3095283
OV-GG	http://www.ovt.com/download_document.php?type=sensor&sensorid=147

	https://www.digikey.com/product-detail/en/omnivision-technologies-inc/OV06211-EAAA-AA0A/OV06211-EAAA-AA0A-ND/5022490
OV-RC	http://www.ovt.com/download_document.php?type=sensor&sensorid=144
	https://www.digikey.com/product-detail/en/omnivision-technologies-inc/OVM9724-RYDA/OVM9724-RYDA-ND/4377197
CMUCam5	http://charmedlabs.com/default/pixy-cmucam5/
	http://cmucam.org/projects/cmucam5
MS-LifeCam	http://www.microsoft.com/hardware/en-us/p/lifecam-hd-3000
PG-CG-1	http://www.ptgrey.com/firefly-mv-03mp-color-usb-20-micron-mt9v022
	http://www.trossenrobotics.com/fireflyMV
PG-CG-0	http://www.ptgrey.com/flea3-03-mp-color-gige-vision-sony-icx618-camera
	http://www.iceinspace.com.au/forum/archive/index.php/t-73527.html
PG-XB3	http://www.ptgrey.com/bumblebee-xb3-stereo-vision-13-mp-color-firewire-1394b-38mm-sony-icx445-camera
	http://www.ptgrey.com/support/downloads/10132
LM-LeapMotion	https://www.leapmotion.com/product
PG-08S2	http://ebuying365.com/Product/21925291896/
PG-HICOL	http://www.ptgrey.com/ladybug2-48-mp-firewire-1394b-spherical-digital-video-camera-system
	https://www.govdeals.com/index.cfm?fa=Main.Item&itemid=105&acctid=6542
MS-Kinect	http://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/
MS-XB1-Kinect	http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1
	https://www.amazon.com/Xbox-One-Kinect-Sensor/dp/B00INAX3Q2
Laser-Scanner	Chapter 6
V-Roach-Cam	Bermudez [4]
SB-VI	http://www.skybotix.com/skybotix-wordpress/wp-content/uploads/2013/12/VISensor_Factsheet_web.pdf
	http://www.ros.org/news/2014/05/skybotix-opens-vi-sensor-early-adopter-program.html
Backpack	https://www.voanews.com/a/new-tool-maps-buildings-energy-efficiency/2660519.html
V-Roach	Haldane [28]

Appendix B

Hardware Designs

B.1 VelociRoACH

URL : `https://www.github.com/biomimetics/imageproc_pcb`

The design of the VelociRoACH robot embedded control board is available at the above repository. Details on the processor type, radio, IMU, encoder control, and energy sensing can be found within the EAGLE design files.

B.2 Zumy

URL : `https://wiki.eecs.berkeley.edu/biomimetics/Main/Zumy`

The Zumy robot used in Chapters 2, 5, and 6 was designed to be an open-source low-cost robotic platform. As such, much of the design documentation can be found at the public documentation Wiki link above. This site includes the most recent bill of materials, Printed Circuit Board designs, and assembly instructions.

B.3 Active Marker Hat

The active marker hat designed to be used on the VelociRoACH and Zumy platforms pictured in Figure 6.3b is a configurable mount for a collection of programmable RGB LED units with diffusers. It is constructed as collection of carbon-fiber spars held together with cyanoacrylate glue at 3D printed nodes to roughly form a hexagonal base frustum. Each node has a slotted face for accepting LED clip-on units, allowing markers to be placed at any node. The marker unit holds a Worldsemi WS2812 RGB LED unit next to a translucent 1cm diameter polypropylene sphere as a diffuser. A chain of six of these LED units are wired together as per the WS2812 datasheet for data-in data-out chaining, allowing the LED brightnesses to be programmed with a single serial data line.

The geometry files for the full assembly and 3D printed parts can be found as the same file server referenced in Appendix D, in folder `public/coop_slam/mechanical_designs/hat/vroach/space_frame`. Most files are available as a SolidWorks part file (SLDPRT) and STereoLithography (STL) file for use in 3D printing.

B.3.1 Files:

- `vroach_space_frame.SLDASM`: SolidWorks assembly of entire space frame hat.
- `vroach_strut_*.SLDPRT`: part files showing lengths of carbon fiber spars.
- `vroach_node_*.SLDPRT/STL`: part files for each frame node piece.
- `vroach_clip.SLDPRT/STL`: part files for clip to hold frame on VelociRoACH robot.

The geometry files for the marker clips are in folder `public/coop_slam/mechanical_designs/hat/vroach/space_frame`:

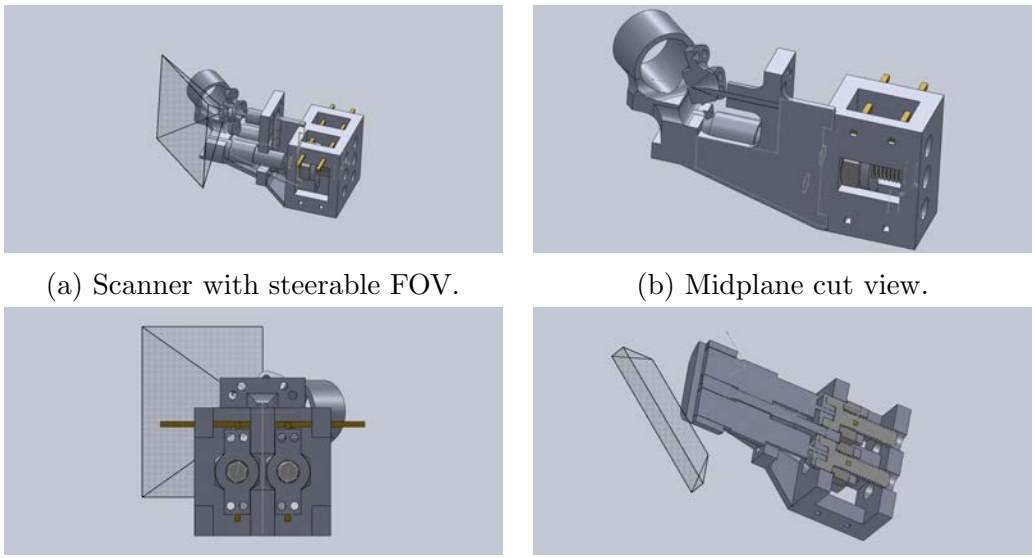
- `marker_thin.SLDASM`: assembly of entire clip, retainer, LED, and diffuser.
- `led_clip_thin.SLDPRT/STL`: housing of LED clip.
- `led_retainer_thin.SLDPRT/STL`: retainer to hold LED in to clip.

B.4 Laser Scanner

The laser scanning system developed in Chapter 6 supports a steerable section of mirror in front of a laser to allow the laser to be projected anywhere in a 90° pyramid in front of the scanner. It uses two DC motors with planetary gear reductions connected to machine screws to move a pair of sliders attached to the elastic-spring loaded mirror pivot to achieve control over two-degree-of-freedom angular displacement. Metallic stops for the sliders are also used to sense when either slider is at the extreme of its motion. Figure B.1 shows several renders of the complete mechanism with section views from various illustrative planes. The geometry files for 3D printing are available on the public file server in folder `public/coop_slam/mechanical_designs/hat/vroach/space_frame`.

B.4.1 Files:

- `scanner.SLDASM`: full assembly of scanner mechanism.
- `guide.SLDPRT/STL`: the guide box that contains the sliders and interfaces with the mount.
- `slide.SLDPRT/STL`: one of two sliders that are actuated via the machine screws and nylon nut.



(a) Scanner with steerable FOV.

(b) Midplane cut view.

(c) Rear cut view.

(d) Top cut view.

Figure B.1: Renders of laser scanner hardware

- `mount.SLDPRT/STL`: front mount that abuts guide box, and supports pin pivot to mirror plate.
- `mirror_plate.SLDPRT/STL`: support plate for mirror segment with thread holes for elastic bands and pull strings to sliders.
- `M2_adapter.SLDPRT/STL`: adapter from gearbox output to hexagonal socket of machine screw.

B.4.2 Control Board

Figure B.2 shows the schematic for the laser scanner control board. The full schematic, printed circuit board file, and part libraries are available in the repository at `https://github.com/biomimetics/modular_robotics_pcb/tree/master/laser_scanner/board`.

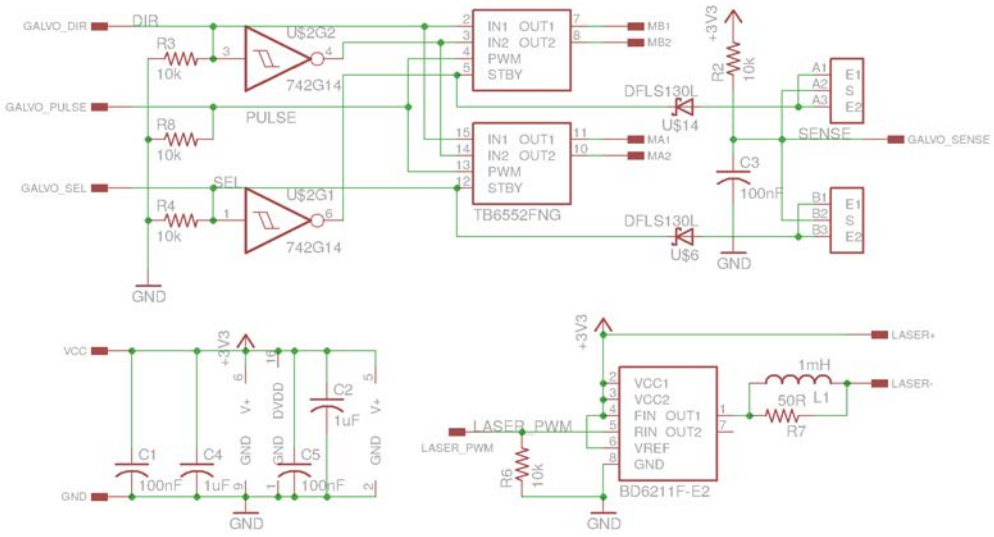


Figure B.2: Laser scanner schematic.

Appendix C

Software

C.1 Automatic PWA Model Identification

URL : `https://www.github.com/biomimetics/auto_pwa`

This repository includes software used in Chapter 3 to filter motion capture and IMU data into a consistent state and state derivative trajectory, and create Piecewise Affine motion models from the resulting data for each of the model type discussed in the chapter.

C.2 VelociRoACH Embedded

URL : `https://www.github.com/biomimetics/roach`

This repository includes the embedded software running on the VelociRoACH main board used in Chapters 2 and 3. It is responsible for on-board leg velocity control, radio command processing, and sensor measurement, recording, and streaming.

C.3 Zumy Embedded and ROS

Embedded URL : `https://www.github.com/biomimetics/mbed_zumy`

Platform URL : `https://www.github.com/biomimetics/ros_zumy`

Remote Launch URL : `https://www.github.com/biomimetics/odroid_machine`

Joystick URL : `https://www.github.com/biomimetics/man_joy_override`

This repositories contain all of the code required to run the Zumy robots used in Chapters 2, 5, and 6. The Embedded repository is the software for the LPC1768 embedded microcontroller on-board the Zumy that coordinates sensor measurements, track velocity

control, and interfacing with sensor or actuator extensions such as the active markers or scanning laser system. The "packet" branch is the one used for this research.

The Platform repository is the software running on the Zumy ODROID embedded platform that handles communication with the LPC1768, and interfacing with ROS communication layers. It publishes all sensor measurements as topics and relays commands for velocity control and exteneral sensors from input topics or services. The "packet" branch is the one used for this research.

The Remote Launch repository is responsible for launching control software on Zumy robots from a host machine over a network via ROS infrastructure.

The Joystick repository provides a convenience utility for using a gamepad joystick to control multiple robots.

C.4 Monocular Pose Estimation

URL : `https://www.github.com/biomimetics/rpg_monocular_pose_estimator`

This repository has the pose estimation software adapted from [21] to use hue information and scale to multiple robots as used in Chapters 4, 5, and 6. The "master" branch was used for the work in Chapter 4, the "zumy" branch for Chapter **??**, and the "factored" branch was used for the work in Chapter 6.

C.5 V-REP Simulation

V-REP URL : `https://www.github.com/biomimetics/bml_vrep`

V-REP Model Server Folder : `vrep_mapping/coop_slam`

Exploration URL : `https://www.github.com/biomimetics/exploration`

These repositories contain code necessary to support the simulation of robots in V-REP doing monocular pose estimation and cooperative range finding as discussed in Chapter 6. The V-REP repository contains code to configure a team of robots and an environment within the V-REP simulation, with a ROS-based interface for control and sensing. Robot and environment model files are available on the server referenced in Appendix D, under the Model Server Folder listed above. The Exploration repository contains software for creating energy estimates from simulation, and information metric calculation.

Appendix D

Datasets

The following datasets can be reached by logging in to the following public file server:

URL: `https://biomimetics.eecs.berkeley.edu`

Username: `public_user`

Password: `NfUPxBONWrl3B`

Each entry below details a folder in this file server interface under "`public/coop_slam`."

D.1 Cooperative Exploration

The folder "`cooperative_exploration`" contains a video and ROS bag files logging all ROS topics used in Chapter 2.

D.1.1 Files:

- `cooperative_exploration.mp4`: Video showing safe exploration and granular media, as well as mapping results.
- `manual.bag`: ROS bag file for manual drive experiments.
- `random.bag`: ROS bag file for random drive experiments.

D.2 Automatic PWA Model Identification

The folder "`model_identification`" contains datasets of the VelociRoACH running on a treadmill and videos used to create the models in Chapter 3.

D.2.1 Files:

- `Trial2013_03_08_21_26_53_front.mov`: Video file showing front view of treadmill running.
- `Trial2013_03_08_21_26_53_side.mov`: Video file showing side view of treadmill running.
- `Trial_*_clip.mat`: MATLAB format data file with state trajectory and derivative.
- `Trial_*_clip_stats.mat`: MATLAB format data file with statistics of each type of model for corresponding state trajectory file.
- `Trial_*_clip_models.mat`: MATLAB format data file with learned models for corresponding state trajectory file.

D.3 Monocular Pose Estimation

The folder "`monocular_pose_estimation`" contains ROS bag files used to create the pose estimate calibration data and pose trajectory plot in Chapter 4.

D.3.1 Files:

- `calibration*.bag`: ROS bag file showing angle or axis sweep for estimation accuracy calibration.
- `circles_and_shaking.bag`: ROS bag file for trajectory of pose estimation.

D.4 Inchworm Localization

The folder "`inchworm_localization`" contains ROS bag files and a video showing a team of Zumy robots navigating an environment with Inchworm localization as in Chapter 5.

D.4.1 Files:

- `updated_coop_slam.mp4`: Video showing robots navigating environment, and results.
- `maze.bag`: ROS bag file with information from robot team (video, IMU, control).
- `vicon_maze.bag`: ROS bag file with Vicon motion tracking ground truth.

D.5 V-REP Mapping Simulation

The folder "`vrep_mapping`" contains a ROS bag file and video showing two robots mapping a simulated environment in V-REP, as detailed in Chapter 6.

D.5.1 Files:

- `move_map.avi`: Video file showing perspective view of robots moving and mapping corner in V-REP simulation.
- `move_map.avi`: ROS bag file with resulting data from navigation and mapping.

Bibliography

[1] Aamir Ahmad et al. "Cooperative robot localization and target tracking based on least squares minimization". In: *IEEE Int. Conf. on Robotics and Automation, 2013*, pp. 5696–5701.

[2] R. Altendorfer, Daniel E. Koditschek, and Philip J Holmes. "Stability Analysis of a Clock-Driven Rigid-Body SLIP Model for RHex". In: *The International Journal of Robotics Research* 23.10-11 (Oct. 2004), pp. 1001–1012. ISSN: 0278-3649. DOI: 10.1177/0278364904047390.

[3] Kostas E Bekris, Max Glick, and Lydia E Kavraki. "Evaluation of algorithms for bearing-only SLAM". In: *IEEE Int. Conf. on Robotics and Automation, 2006*, pp. 1937–1943.

[4] Fernando Luis Garcia Bermudez. *Compensation for camera motion on unsteady robots for optical flow*. University of California, Berkeley, 2013.

[5] Paul Birkmeyer, Kevin Peterson, and Ronald S. Fearing. "DASH : A dynamic 16g hexapedal robot". In: *IEEE Int. Conf. on Intelligent Robots and Systems* (2009), pp. 2683–2689.

[6] Michael Bloesch et al. "Kinematic Batch Calibration for Legged Robots". In: *IEEE Int. Conf. on Robotics and Automation* 3 (2013), pp. 2527–2532.

[7] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[8] Urban Brändström. *The ALIS Imager: Some basic concepts*. 2004. URL: http://www.irf.se/~urban/avh/html/node13.html (visited on 02/22/2017).

[9] Andreas Breitenmoser, Laurent Kneip, and Roland Siegwart. "A monocular vision-based system for 6D relative robot localization". In: *IEEE Int. Conf. on Intelligent Robots and Systems, 2011*, pp. 79–85.

[10] Rodney A Brooks and Anita M Flynn. *Fast, cheap and out of control*. Tech. rep. Massachusetts Institute of Technology Artificial Intelligence Lab, 1989.

[11] Austin D Buchan, Duncan W Haldane, and Ronald S Fearing. "Automatic identification of dynamic piecewise affine models for a running robot". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 5600–5607.

[12] Austin D. Buchan et al. "Low-Cost Monocular Localization with Active Markers for Micro Autonomous Underwater Vehicles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (accepted)*. IEEE. 2017.

[13] Samuel Burden, Shai Revzen, and SS Sastry. "Dimension reduction near periodic orbits of hybrid systems". In: *IEEE Conf. on Decision and Control* (2011), pp. 6116–6121. arXiv: arXiv:1109.1780v1.

[14] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[15] Frank M. Callier and Charles A. Desoer. *Linear System Theory*. London, UK, UK: Springer-Verlag, 1991. ISBN: 0-387-97573-X.

[16] Luca Carlone et al. "Rao-Blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication". In: *IEEE Int. Conf. on Robotics and Automation, 2010*, pp. 243–249.

[17] Philippe Cavalier. *Quatro 290 - 1M CAD Model*. 2016. URL: https://grabcad.com/library/quatro-290-1m-1 (visited on 02/15/2017).

[18] Aleksandr Dikarev et al. "Combined multiuser acoustic communication and localisation system for μAUVs operating in confined underwater environments". In: *IFAC-PapersOnLine* 48.2 (2015), pp. 161–166.

[19] A. Dirafzoon et al. "Mapping of unknown environments using minimal sensing from a stochastic swarm". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. Sept. 2014, pp. 3842–3849. DOI: 10.1109/IROS.2014.6943102.

[20] FR Fabresse, F Caballero, and A Ollero. "Decentralized simultaneous localization and mapping for multiple aerial vehicles using range-only sensors". In: *IEEE Int. Conf. on Robotics and Automation, 2015*, pp. 6408–6414.

[21] Matthias Faessler et al. "A Monocular Pose Estimation System based on Infrared LEDs". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 907–913.

[22] Giancarlo Ferrari-Trecate et al. "A clustering technique for the identification of piecewise affine systems". In: *Automatica* 39.2 (Feb. 2003), pp. 205–217. ISSN: 00051098. DOI: 10.1016/S0005-1098(02)00224-8.

[23] D. Fridovich-Keil, E. Nelson, and A. Zakhor. "AtomMap: A probabilistic amorphous 3D map representation for robotics and surface reconstruction". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 3110–3117. DOI: 10.1109/ICRA.2017.7989355.

[24] Andreas R Geist et al. "Towards a Hyperbolic Acoustic One-Way Localization System for Underwater Swarm Robotics". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4551–4556.

[25] Robert Grabowski et al. "Heterogeneous Teams of Modular Robots for Mapping and Exploration". en. In: *Autonomous Robots* 8.3 (June 2000), pp. 293–308. ISSN: 0929-5593, 1573-7527. DOI: 10.1023/A:1008933826411. URL: http://link.springer.com/article/10.1023/A:1008933826411 (visited on 09/12/2015).

[26] Arron Griffiths et al. "AVEXIS-Aqua Vehicle Explorer for In-Situ Sensing". In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 282–287.

[27] Axel Hackbarth, Edwin Kreuzer, and Eugen Solowjow. "HippoCampus: A Micro Underwater Vehicle for Swarm Applications". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 2258–2263.

[28] Duncan W Haldane et al. "Animal-inspired Design and Aerodynamic Stabilization of a Hexapedal Millirobot". In: *IEEE Int. Conf. on Robotics and Automation* (2013), pp. 3264–3271.

[29] Duncan W. Haldane et al. "Detection of slippery terrain with a heterogeneous team of legged robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 4576–4581. DOI: 10.1109/ICRA.2014.6907527.

[30] P.E. Hart, N.J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (July 1968), pp. 100–107. ISSN: 0536-1567. DOI: 10.1109/TSSC.1968.300136.

[31] Richard I Hartley and Peter Sturm. "Triangulation". In: *Computer vision and image understanding* 68.2 (1997), pp. 146–157.

[32] Constantin Herbst et al. *Safe Navigation with a Heterogeneous Team of Legged Robots*. 2013. URL: http://students.asl.ethz.ch/upl_pdf/511-report.pdf (visited on 09/12/2015).

[33] *Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder - RobotShop*. URL: http://www.robotshop.com/en/hokuyo-urg-04lx-ug01-scanning-laser-rangefinder.html.

[34] Philip Holmes et al. "The Dynamics of Legged Locomotion: Models, Analyses, and Challenges". In: *SIAM Review* 48.2 (Jan. 2006), pp. 207–304. ISSN: 0036-1445. DOI: 10.1137/S0036144504445133.

[35] Aaron M. Hoover et al. "Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot". In: *3rd IEEE RAS and EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob)*. Sept. 2010, pp. 869–876.

[36] A.M. Hoover and R.S. Fearing. "Fast scale prototyping for folded millirobots". In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. May 2008, pp. 1777–1778. DOI: 10.1109/ROBOT.2008.4543462.

[37] Andrew Howard. "Multi-robot simultaneous localization and mapping using particle filters". In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1243–1256.

[38] Guoquan Huang, Michael Kaess, and John J Leonard. "Towards consistent visual-inertial navigation". In: *IEEE Int. Conf. on Robotics and Automation, 2014*, pp. 4926–4933.

[39] J. Huang et al. "Efficient, generalized indoor WiFi GraphSLAM". In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 1038–1043. DOI: 10.1109/ICRA.2011.5979643.

[40] Vadim Indelman et al. "Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization". In: *IEEE Int. Conf. on Robotics and Automation, 2014*, pp. 593–600.

[41] George H Joblove and Donald Greenberg. "Color Spaces for Computer Graphics". In: *ACM SIGGRAPH Computer Graphics*. Vol. 12. 3. ACM. 1978, pp. 20–25.

[42] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 2969–2976.

[43] Haldun Komsuoglu et al. "A physical model for dynamical arthropod running on level ground". In: *Int. Symposium on Experimental Robotics* (2008), pp. 303–317.

[44] Haldun Komsuoglu et al. "Characterization of Dynamic Behaviors in a Hexapod Robot". In: *Int. Symposium on Experimental Robotics* (2010).

[45] Ryo Kurazume and Shigeo Hirose. "An experimental study of a cooperative positioning system". In: *Autonomous Robots* 8.1 (2000), pp. 43–52.

[46] Raj Madhavan, Kingsley Fregene, and Lynne E Parker. "Distributed heterogeneous outdoor multi-robot localization". In: *IEEE Int. Conf. on Robotics and Automation, 2002*. Vol. 1, pp. 374–381.

[47] Agostino Martinelli, Frederic Pont, and Roland Siegwart. "Multi-robot localization using relative observations". In: *IEEE Int. Conf. on Robotics and Automation, 2005*, pp. 2797–2802.

[48] Allison Mathis et al. "Autonomous Navigation of a 5 Gram Crawling Millirobot in a Complex Environment". In: Baltimore, 2012, pp. 23–26. URL: https://robotics.eecs.berkeley.edu/~ronf/PAPERS/MEDIC-clawar12.pdf (visited on 09/12/2015).

[49] Christoph Mertz et al. "A low-power structured light sensor for outdoor scene reconstruction and dominant material identification". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012, pp. 15–22.

[50] Rudolph Van Der Merwe, Eric A Wan, and Simon I Julier. "Nonlinear Estimation and Sensor-Fusion - Applications to Integrated Navigation -". In: *Proc. AIAA Guidance Navigation and Controls Conf* (2004), pp. 1–30.

[51] Anastasios I Mourikis and Stergios I Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation". In: *IEEE Int. Conf. on Robotics and Automation, 2007*, pp. 3565–3572.

[52] R.R. Murphy. "Marsupial and shape-shifting robots for urban search and rescue". In: *IEEE Intelligent Systems and their Applications* 15.2 (Mar. 2000), pp. 14–19. ISSN: 1094-7167. DOI: 10.1109/5254.850822.

[53] Luis E. Navarro-Serment, Christiaan J. J. Paredis, and Pradeep K. Khosla. "A Beacon System for the Localization of Distributed Robotic Teams". In: *In Proceedings of the International Conference on Field and Service Robotics*. 1999, pp. 232–237.

[54] Brian E Nemsick et al. "Cooperative inchworm localization with a low cost team". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 6323–6330.

[55] Jorge Nocedal and Stephen J Wright. *Numerical Optimization, 2nd Ed.* Springer, 2006.

[56] *OpenMV Cam M4 V2 — OpenMV*. URL: https://openmv.io/products/openmv-cam.

[57] OpenMV LLC. *OpenMV Lenses*. 2016. URL: https://openmv.io/collections/lenses (visited on 02/28/2017).

[58] Daegil Park et al. "3D Underwater Localization Scheme using EM Wave Attenuation with a Depth Sensor". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 2631–2636.

[59] Robin M Pope and Edward S Fry. "Absorption spectrum (380–700 nm) of pure water. II. Integrating cavity measurements". In: *Applied optics* 36.33 (1997), pp. 8710–8723.

[60] Amanda Prorok, Alexander Bahr, and Alcherio Martinoli. "Low-cost collaborative localization for large-scale multi-robot systems". In: *IEEE Int. Conf. on Robotics and Automation, 2012*, pp. 4236–4241.

[61] Long Quan and Zhongdan Lan. "Linear n-point camera pose determination". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8 (1999), pp. 774–780.

[62] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 2009. URL: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf (visited on 09/12/2015).

[63] H. E. Rauch, C. T. Striebel, and F. Tung. "Maximum likelihood estimates of linear dynamic systems". In: *AIAA Journal* 3.8 (Aug. 1965), pp. 1445–1450. ISSN: 0001-1452. DOI: 10.2514/3.3166. URL: http://arc.aiaa.org/doi/abs/10.2514/3.3166.

[64] Shai Revzen and John M Guckenheimer. "Finding the dimension of slow dynamics in a rhythmic system." In: *Journal of the Royal Society, Interface / the Royal Society* 9.70 (May 2012), pp. 957–71. ISSN: 1742-5662. DOI: 10.1098/rsif.2011.0431.

[65] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *Intelligent Robots and Systems (IROS), 2013 IEEE / RSJ International Conference on*. IEEE. 2013, pp. 1321–1326.

[66] Cameron J Rose, Parsa Mahmoudieh, and Ronald S Fearing. "Modeling and Control of an Ornithopter for Diving". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 957–964.

[67] Stergios I Roumeliotis and George A Bekey. "Distributed multirobot localization". In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 781–795.

[68] Sumant Sharma and Simone D'Amico. "Comparative assessment of techniques for initial pose estimation using monocular vision". In: *Acta Astronautica* 123 (2016), pp. 435–445.

[69] E Sontag. "Nonlinear regulation: The piecewise linear approach". In: *IEEE Transactions on Automatic Control* (1981), pp. 346–358.

[70] Dennis Strelow and Sanjiv Singh. "Motion estimation from image and inertial measurements". In: *The International Journal of Robotics Research* 23.12 (2004), pp. 1157–1195.

[71] R. Tedrake et al. "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification". In: *The International Journal of Robotics Research* 29.8 (Apr. 2010), pp. 1038–1052. ISSN: 0278-3649. DOI: 10.1177/0278364910369189.

[72] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. 2005.

[73] Stephen Tully, George Kantor, and Howie Choset. "Leap-frog path design for multi-robot cooperative localization". In: *Int. Conf. on Field and Service Robotics*. Springer. 2010, pp. 307–317.

[74] Eric Turner, Peter Cheng, and Avideh Zakhor. "Fast, automated, scalable generation of textured 3d models of indoor environments". In: *IEEE Journal of Selected Topics in Signal Processing* 9.3 (2015), pp. 409–421.

[75] C. Urmson and R. Simmons. "Approaches for heuristically biasing RRT growth". In: *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings*. Vol. 2. Oct. 2003, 1178–1183 vol.2. DOI: 10.1109/IROS.2003.1248805.

[76] Thumeera R Wanasinghe, George KI Mann, and Raymond G Gosine. "Distributed collaborative localization for a heterogeneous multi-robot system". In: *27th IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE), 2014*, pp. 1–6.

[77] Thumeera R Wanasinghe, George KI Mann, and Raymond G Gosine. "Distributed Leader-Assistive Localization Method for a Heterogeneous Multirobotic System". In: *IEEE Transactions on Automation Science and Engineering* 12.3 (2015), pp. 795–809.

[78] Karl E Wenzel, Andreas Masselli, and Andreas Zell. "Visual tracking and following of a quadrocopter by another quadrocopter". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4993–4998.