University of California
Los Angeles

An Embedded Autonomous Multi-Vehicle Testbed for Development of Cooperative Control Algorithms

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical Engineering

by

Yuan Rick Huang

2006

The thesis of Yuan Rick Huang is approved.

William J. Kaiser

Andrea L. Bertozzi, Committee Co-chair

Rajeev Jain, Committee Co-chair

University of California, Los Angeles

2006

To my dearest grandma Huorong and grandma Tackjing,
whose love continues to give me courage and strength,
and I miss them ...

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I want to express my deepest gratitude to my advisers: Prof. Andrea Bertozzi, for providing me a genuinely exciting and thoroughly rewarding research experiences in the UCLA Applied Math Lab (AML), and Prof. Rajeev Jain, for guiding me through the writing process with pointed advice and rare patience (I just love to listen to you guys chat about stuff. It was both view-broadening and thought-provoking). I also want to thank Prof. William Kaiser for kindly serving on my committee and being a truly wonderful teacher, and Prof. Emilio Frazzoli for the inception of the testbed and sharing his vast knowledge.

My highest appreciation also goes to fellow members of AML robotic research group: Chung Hsieh for leadership and and Kevin Leung for partnership. Without their enthusiasm and dedication, this project would not have been possible. Abhijeet Joshi and Vlad Voroninski had also contributed significantly. I am grateful for their hard work and friendship.

I am very thankful for my other collaborators, including Yao-Li Chuang, Maria D'Orsogna, Yanina Landa, Jenny Treanor, David Galkowski, G. Malla, and Carmeliza Navasca. It was total pleasure to work with them.

I am also grateful for the support provided by ARO MURI grant 50363-MA-MUR, ARO grant W911NF-05-1-0112, and ONR grant N000140610059.

I am indebted to Bernie Yip, Joanna Liao, Alan Liao, Kevin Leung (again), Jason Fang, Jinghui Gao, Ya-Leng Lin, Jenny To, Janice Hsu, and Pen Lin. They are my family in Los Angeles. It was their care and support that accompanied me through many ups and downs of my UCLA journey.

My gratitude also goes to the family of my three uncles, especially Uncle Philip and Aunt Jane, without whom it would have been a miracle that I could even

set foot in UCLA.

Last but not least, I want to thank my parents for their endless love and care.

Abstract of the Thesis

An Embedded Autonomous Multi-Vehicle Testbed for Development of Cooperative Control Algorithms

by

Yuan Rick Huang

Master of Science in Electrical Engineering

University of California, Los Angeles, 2006

Professor Rajeev Jain, Co-chair

Professor Andrea L. Bertozzi, Co-chair

An autonomous multi-vehicle testbed is an important development tool for the cooperative control algorithms required in the deployment of an autonomous multi-vehicle system, that can operate in geographically complex terrains. Although many such testbeds exists, the cost and size per vehicle is too large for multi-vehicle studies with more than five vehicles. To address this issue, we have developed a low-cost ($160 per vehicle) and space-efficient (1/64th scale vehicles) autonomous multi-vehicle testbed by leveraging embedded systems technologies with off-the-shelf devices and open-source software. We were able to implement cooperative control algorithms like UAV-routing and multi-vehicle flocking based on pairwise interacting potentials. This thesis focuses on the system design, subsystem device selection and embedded firmware development of the testbed. We also present application examples for dynamic obstacle avoidance and environ-

ment visibility exploration, which are implementable with on-board proximity sensors.

CHAPTER 1

Introduction

1.1 Motivation

1.1.1 Need for Autonomous Multi-Vehicle Systems

An autonomous vehicle can move from location to location and perform tasks all on its own without the need for a human operator. Once fully realized, autonomous vehicles can bring enormous benefit to human society, where many hazardous tasks still rely on human operation. First, dangerous missions can be carried out without risking human lives. Second, human operator payload can be eliminated resulting in higher economic and energy efficiency. Third, car accidents and the resulting casualty can be almost eliminated as the absolute majority are caused by human driver errors (e.g. drunk driving). Fourth, convenience can be brought to many, who are either not able or simply not willing to drive: from visually impaired to elderly and children, to commuters who would rather spend time on reading and people, who just cannot handle parallel parking. Autonomous vehicles can also assist in improving traffic congestion by computerized adaptive navigation that allows cars to travel faster and closer together.

Many tasks cannot be accomplished by a single vehicle (e.g. war zone combat and rescue), while some others can gain efficiency from simultaneously deploying a

group or even an army of vehicles, i.e. a multi-vehicle system (e.g. environmental exploration and search). One can see the effectiveness of a multi-agent system in a scene of the movie "The Minority Report", in which a swarm of spider-like robots effectively search an apartment building for a criminal. Furthermore, a multi-vehicle system can easily resolve the problem of single point failure as vehicles can replan their motion to make up for the loss of one. A multi-vehicle system is irreplaceable by a vehicle on its own, however sophisticated the vehicle might be. The ability to share resource, tasks, and information among constituents can lead to more effective solutions for problems in environmental monitoring, surveillance, military mission, and many other areas.

1.1.2 Cooperative Control Algorithms

A prerequisite to successful deployment of an autonomous multi-vehicle system is an effective control algorithm or planning strategy that targets its intended application. In order to use an autonomous vehicle system to perform a certain task, high-level goals and specifications have to be first translated into low-level commands and signals. In other words, the vehicles have to be told how to move. To achieve such a goal, path and motion planning algorithms are developed and applied.

For multi-vehicle systems, the planning algorithms involved are mostly aimed at achieving cooperation and optimized interaction among constituents. Some of the algorithms are inspired by natural self-organizing biological systems like a school of fish, ant colonies, or an army of grasshoppers. Such algorithms bear the objective to controlling the vehicles to maintain a cohesive formation, while avoiding collisions between them. Such capability is of high demand in applications like formation flying and combat systems. Other multi-vehicle algorithms target

applications like environmental monitoring, search, surveillance, and war zone rescue missions, where a series of tasks are shared based on spatial characteristics. A well known example in this area is the multi-vehicle "Traveling Salesperson Problem", in which a team of unmanned vehicles service stochastically-defined targets and minimize the average waiting time.

1.1.3 Control Algorithm Development: an Emulation Testbed

A testbed is desirable to emulate and validate the effectiveness and real-time practicality of the control algorithms before directly applying them to the expensive actual vehicles. To validate the effectiveness of an algorithm, computer simulations are extensively used. However, many simplification or approximation are usually made in a computer simulation, thus the algorithm can only be deemed valid under very ideal conditions. The implementation of such algorithms in a real environment can pose some unanticipated problems and challenges. Since the ultimate goal of any control algorithm is real environment feasibility, a multi-vehicle testbed becomes very necessary. In addition, such a testbed can also provide an invaluable learning tool for education on many subjects including control theories, embedded system, system engineering, networked sensing, communication systems, and others. Students would be able to appreciate the theories better and be more motivated with the opportunity to experience real-world implementation. This thesis will describe the development of an economical, compact-size multi-vehicle indoor testbed, which can truly emulate the real world implementation of a similar system.

1.2 Related Work

As technology advances in computers, wireless communications, embedded sensing, and miniaturization of electromechanical systems, many researchers have developed autonomous multi-vehicle testbeds [6 - 17].

At Brigham Young University, the MAGICC (Multiple AGent Intelligent Coordination & Control) lab focuses its effort on developing algorithms and architectures for cooperative control of multiple UAV (unmanned aerial vehicle) ([14]). Since the beginning of their research in 1999, they have built a ground vehicle testbed with wheeled robots and overhead positioning system (Figure 1.1), and a UAV testbed featuring several types of airplanes and flying-wing UAV (Figure 1.2).

Figure 1.1: MAGICC lab nonholonomic wheeled robots. Photo courtesy of the MAGICC Lab, Brigham Young University.

The RoboCup competition has attracted much attention and participation around the world since its inception in 1997. Notably, researchers at Cornell University and Carnegie Mellon University have developed testbeds and built RoboSoccer teams, of which the members are essentially autonomous vehicles (Figure 1.3). Through these testbeds and competitions, technologies in area like multi-agent collaboration, strategy acquisition, and sensor fusion have been developed, tested, and improved ([13], [15]). These technologies are also expected

Figure 1.2: Top left: the MAV 06 Champion (wingspan 16 in); top right: the Unicorn (wingspan 48 in), a commercially available plane from Procerus technologies (www.procerusuav.com); bottom: the Bus (wingspan 60 in). Photo courtesy of the MAGICC Lab, Brigham Young University.

to be used in search and rescue missions in large scale disasters. The RoboCup competition features small (maximum diameter 180mm) to medium (occupying more than 900cm^2 of field area) size wheeled vehicles, four-legged robots (e.g. Sony Aibo dog-like robots), and humanoid robots (human-like in terms of body and senses). Its ultimate goal is set at winning the human soccer world champions with a team of humanoid robots by the year of 2050. Thus, besides optimal path planning and cooperative strategies, high speed and other sport related capabilities are also features pursued by the RoboSoccer testbeds.

The UIUC HoTDeC (HOvercraft Testbed for DEcentralized Control) features hovercraft as their primary vehicles (Figure 1.4), which has much less open-loop stability than wheeled vehicles ([11]). This feature allows for simulations of models that can be extended for aerial vehicles, while the implementation is done indoor on the floor thus reducing the risk of damaging vehicles. Another useful feature of the HoTDeC is its capability of remote operation, as the user can change controller parameters and watch web-cam real-time capture via Internet. The autonomous recharging system allows the testbed to run unattended continuously.

The Caltech MVWT (Multi-Vehicle Wireless Testbed) was first developed in 2002. It features overhead video tracking, off-field vision processing system, and a team of fan-driven vehicles that rest on casters (Figure 1.5). The vehicle chassis is rectangle shaped at $37cmL \times 27cmW$. The unit costs more than $2000 [6, 7]. In 2004, the Caltech testbed was upgraded to MVWT-II, introducing a second generation of vehicles that are based on hovercraft, while the size and cost were both significantly reduced. The vehicle body is now a round disk with diameter of $20cm$. In addition to overhead tracking for indoor positioning, differential GPS is added so that the vehicle can operate outdoor as well.

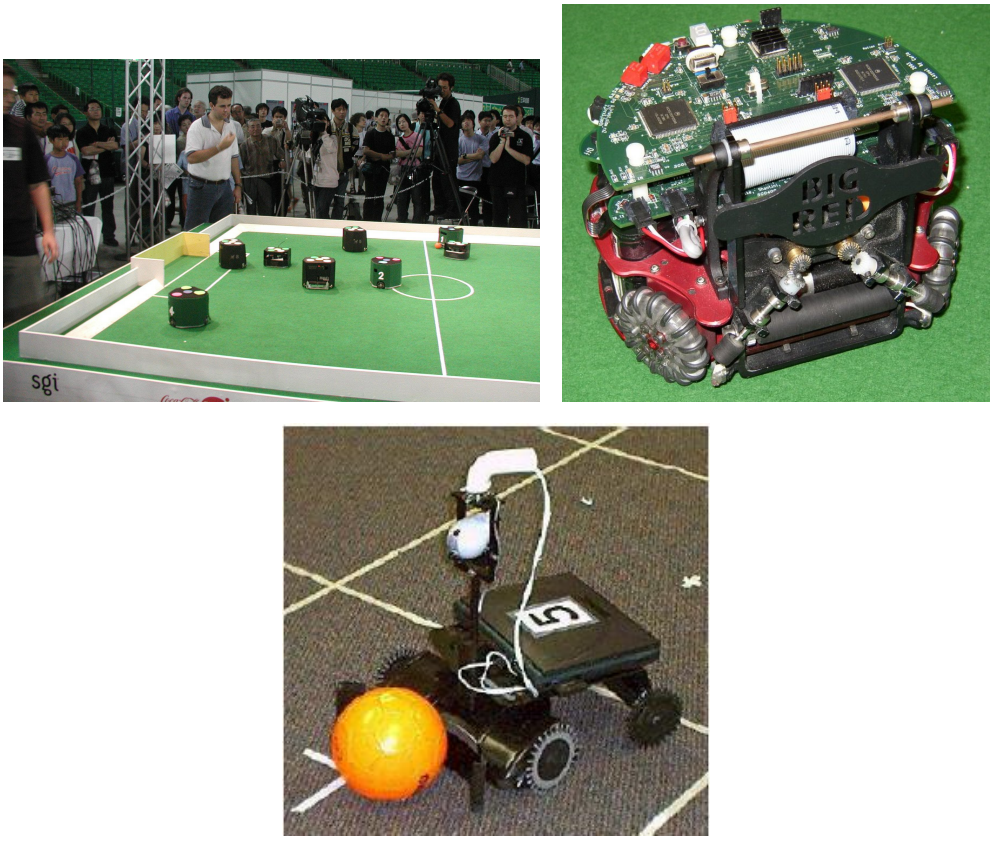

Figure 1.3: Top left: Cornell RoboSoccer Field with robot soccer players; top right: a Cornell RoboSoccer robot without casing. Photo courtesy of Cornell University RoboCup Team. Bottom: a robot from CMU the Minnow Project. Photo courtesy of the Minnow Project within the CORAL research group at Carnegie Mellon University.

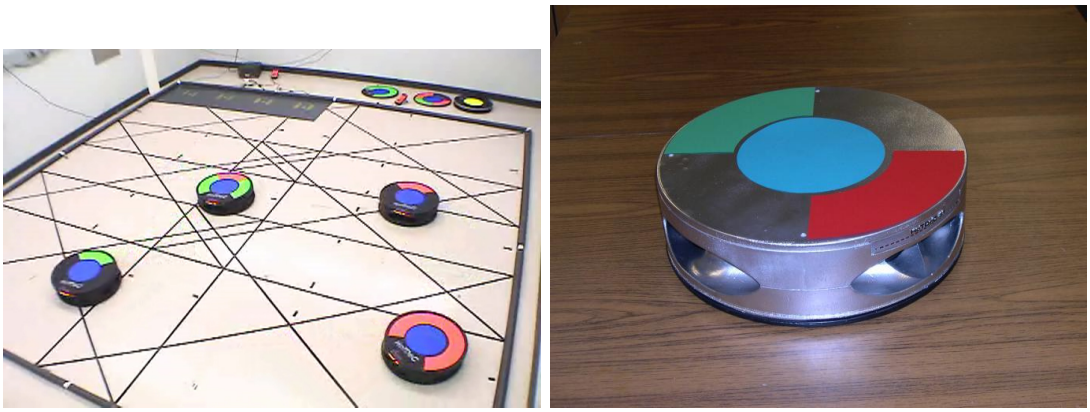

Figure 1.4: Left: a HoTDeC hovercraft; right: the HoTDeC testbed. Photo courtesy of University of Illinois at Urbana-Champaign Hovercraft Testbed for Decentralized Control.

Figure 1.5: Left: a MVWT Kelly II; right: the MVWT first generation testbed. Photo courtesy of California Institute of Technology Multi-Vehicle Wireless Testbed.

.

Testbed (developer)	featured vehicle	vehicle size
MAGICC testbed (BYU)	UAVs	wingspan 610 mm or greater autopilot $89mm \times 51mm$
MVWT (CalTech)	fan-driven caster platform	$370mm \times 270mm$
MVWT-II (CalTech)	Hovercraft	$200mm$ in diameter
CMU Hammerheads	wheeled	laptop size ground area
HoTDeC (UIUC)	Hovercraft	$406mm$ in diameter

Table 1.1: Comparison of Testbed Vehicles and Sizes

Table 1.1 compares the physical dimensions of the vehicles. Although many of the existing testbeds are implemented in a laboratory setting, they tend to demand a significant amount of space due to the size of the vehicle and the fact that the testbed is intended to allow multiple vehicle operating simultaneously. However, many researchers or educators can only afford a limited amount of lab space. Thus, a small-area testbed (that can be fit into any lab or classroom) featuring sub-palm size vehicles would be of great value. In addition, with vehicles at such small scale, the testbed would be more portable than the existing ones thus making remote demonstration possible. Many testbeds do not provide details about their cost, but we estimate them to be on the order of thousands of dollars per vehicle, which makes it prohibitive to have large number of them. For UAVs, the damage of one would be relatively costly, too. Hence a low-cost (no more than a couple hundreds of dollars) testbed is also highly desirable. Table 1.2 and 1.3 list devices or solutions deployed in the existing testbeds. In the research project we leveraged embedded system technology to develop a low cost ($160/vehicle), small scale (100 sq. ft) testbed.

Testbed (developer)	tracking/positioning	processing
MAGICC testbed (BYU)	GPS	29 MHz Rabbit
MVWT (CalTech)	60 Hz Overhead cameras	Intel 700 MHz
MVWT-II (CalTech)	Overhead cameras, DGPS or IR	Strong ARM 206 MHz, 16 MHz Atmel
CMU Hammerheads	on-board camera	P3 700MHz
HoTDeC (UIUC)	30 Hz Overhead cameras	x86 compatible, MCU

Table 1.2: Comparison of Testbed System Components I

Testbed (developer)	embedded sensor	communication
MAGICC testbed (BYU)	gyros	900 MHz wireless modem
MVWT (CalTech)	gyros, ultrasonic	802.11b
MVWT-II (CalTech)	gyro, accelerometer, magnetic heading sensor	802.11b
CMU Hammerheads		802.11
HoTDeC (UIUC)	inertial sensor, hall effect sensor, on-board vision	802.11 or Bluetooth

Table 1.3: Comparison of Testbed System Components II

1.3 Outline of the Rest of the Thesis

The rest of this thesis is organized as follow: In Chapter 2 we first introduce a first generation testbed based on desktop PCs and its functionality and shortcomings. To improve upon the described shortcomings, we migrated the computing burden from a remote PC to embedded processors mounted on the vehicles. In Chapter 3 we describe the selection of devices and the design of the system hardware. In Chapter 4 the embedded firmware is described. To demonstrate how various cooperative control algorithms can be implemented and how theories can be validated using the testbed, we then provide examples of the implementation in Chapter 5. In the end, we conclude by speculating how a real-environment autonomous micro-size multi-vehicle testbed can be realized based on the tested-and-true components of the testbed.

This research project is the fruit of many individuals' intelligence, dedication, and collaboration. The overhead video tracking system, which is mainly inherited from the first generation testbed, is developed by Chung Hsieh and improved with sub-pixel accuracy by Vlad Voroninski. Yao-Li Chuang developed the mathematical model for the first generation vehicle (car) motion. Vlad Voroninski modified the model to describe the second generation car motion and provided a model for the tanks as well. The selection of vehicle subsystem devices is the brainchild of Chung Hsieh, Kevin Leung, and Yuan Huang. Consulted with Chung Hsieh and assisted by Yuan Huang, Kevin Leung put together the vehicle electronic systems, including circuit design, PCB layout, and mechanical integration. The embedded firmware is a joint effort as well by Chung Hsieh (scheduler), Kevin Leung (lower level motion controller and sensor measurement acquisition), and Yuan Huang (communication driver), while the control algorithms are adapted and implemented by Kevin Leung (homotopy, dynamic obstacle detection and

avoidance, and UAV routing), Abhijeet Joshi(circle tracker and dynamic obstacle detection and avoidance), Vlad Voroninski (dynamic obstacle detection and avoidance), Chung Hsieh (UAV routing), and Yuan Huang (UAV routing, multi-vehicle flocking, visibility exploration, and pursuit-evasion). The IR sensor is calibrated by Jennifer Treanor, Abhijeet Joshi, and Yuan Huang. Yana Landa, David Galkowski, Christine Lee, and G. Malla also contributed enormously to the realization of visibility exploration. Last but not least, it was Prof. Andrea Bertozzi's guidance and advice that connected all the dots and elevated the result to be more than the sum of its pieces.

CHAPTER 2

Desktop Based Micro-Car Testbed

2.1 System Overview

In the summer of 2005, a first generation micro-car testbed was constructed at the UCLA Applied Mathematics Lab [1]. This testbed includes a $1.5m \times 2.0m$ arena on the floor, two monochrome CCD cameras mounted on the ceiling providing overhead imaging, a desktop computer as image-processor that calculates the position and orientation for the cars, another desktop computer running control algorithms to provide motion decisions, and radio-controlled cars that execute wirelessly-transmitted decisions on the arena.

Featuring a very space-efficient arena and sub-palm size vehicles, the testbed fits very well into a lab setting of even very modest-size (The UCLA Applied Math Lab has a wall-to-wall width of only 12 ft). With the use of many off-the-shelf components and open source software, the total material cost was kept as low as $4000 including the two PCs and four working vehicles. With the effort of two masters and one undergraduate electrical engineering students working full-time, the testbed was completed in the period of 12 weeks. The testbed has been used for implementation of point-to-point control, arbitrary path following, tri-vehicle region patrolling, and tri-vehicle flocking, which are described in Chapter 5. Figure 2.1 shows a system overview of the testbed.

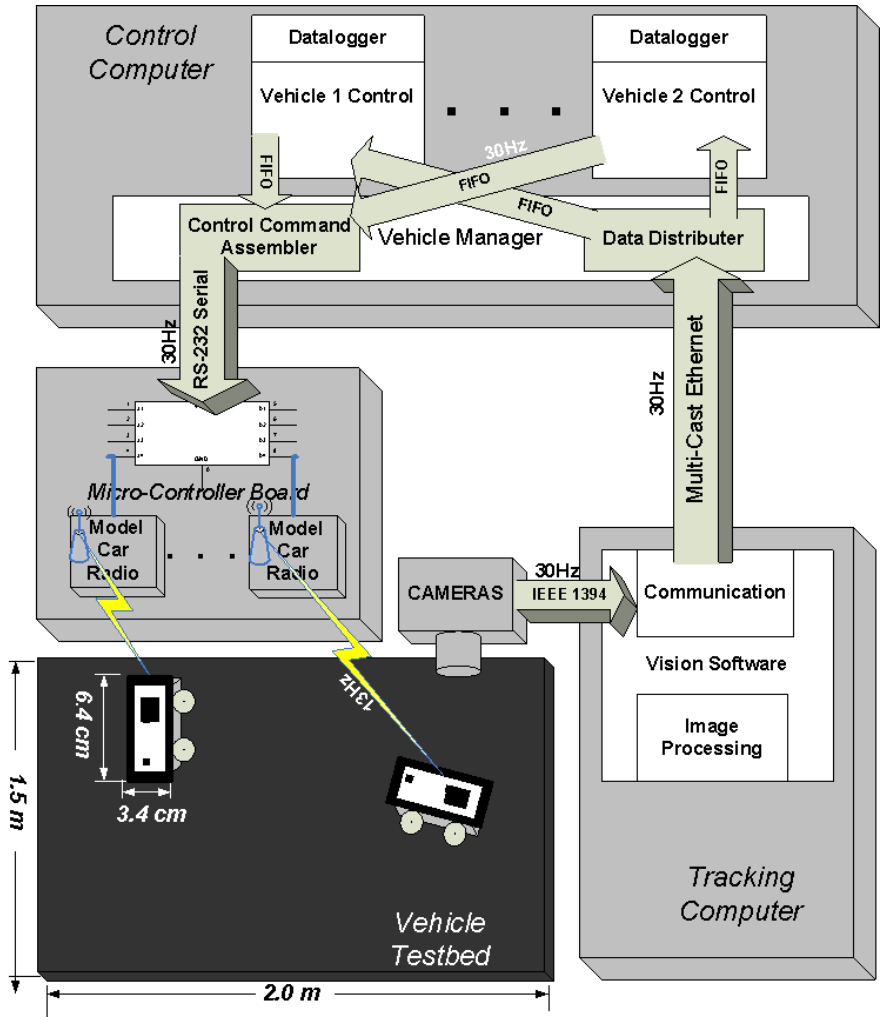

Figure 2.1: A system overview of the first generation testbed. 3-D objects represent embedded components that are implemented on the vehicles, while 2-D objects represent off-board components. The arrows indicate communication connection and direction.

2.2 Vehicle Information

We re-engineered off-the-shelf products suitable for the size and scope of the test bed. We chose Hobbico microsizers radio control (RC) cars (1:64) for their low cost, compactness, weight, minimal modification requirements, and ease of purchase. These cars operate on two carrier frequencies 27MHz and 49MHz using a Time Division Multiple Access (TDMA) transmission scheme. The commands are updated at 13Hz. Each channel is capable of controlling three cars simultaneously to allow up to six vehicles. We made slight modifications to both the vehicle and the radio transmitter to match the needs of the test bed. Out of the box, the vehicles have only 5 minutes of continuous run time. To extend the run time to 30 minutes, we replaced the original battery with two AAA batteries secured to the top of the vehicle. We also replaced the transmitter's button switches with electronic switches in order to incorporate this existing radio control architecture into our testbed. Figure 2.2 shows two of the first generation cars. Table 2.1

Figure 2.2: Cars after integration with added battery rack and marker tag for overhead positioning.

shows physical characteristics of the cars.

Dimension ($L \times W \times H$)	$(6.4 \times 3.4 \times 3.2)cm$
Weight (with batteries)	50g
Steady State Speed	60 cm/s
Acceleration (from zero)	78 cm/s^2
Turning Radius	11-13 cm
Angular velocity	3.3 rad /s

Table 2.1: Specifications of First Generation Vehicles

2.3 Tracking System

The vehicle arena floor has a top layer of asphalt felt paper, which provides a uniform, non-glossy, black background for imaging. An overhead position tracking system is composed of two Imaging Source DMK 21F04 1/4" Monochrome CCD cameras with progressive scan of 30 images/sec at a resolution of 640 x 480 pixels. The cameras are connected via an IEEE 1394a (firewire) interface to a 3.0GHz PC with IEEE1394 adapter card for image processing. The camera lenses are Pentax H612A(KA) with focal length of $6.0mm$ and 56° horizontal angle of view. The cameras are mounted 2.6 meters above the testbed resulting in an observable area for each camera of $1.5m \times 1.1m$, with each pixel representing a $2.4mm \times 2.4mm$ area. The cameras have a 0.144 m image overlap resulting in a total visible arena of $1.5m \times 2m$. Because the whole vehicle must be visible for tracking, the actual trackable region is slightly smaller, $1.4m \times 1.9m$. This setup can be expanded easily by adding more cameras and more image processing computers.

The Vehicle Tracking System software utilizes Intel's computer vision software (OpenCV [19]). We detect the vehicle position and identification using a black and white two dimensional bar code marking similar to that used on the CalTech

Multi-Vehicle Wireless Testbed [6, 7]. The vehicle information are found with an OpenCV contour searching function and the following algorithm.

1. Record all contrasting shapes detected on the platform and bound them within a rectangle.
2. Sort the rectangles into three groups by size.
3. Assign the large rectangles as possible car object.
4. Go through medium sized rectangles:
    - if within bound of a large rectangle, mark as car's heading symbol.
5. Go through small sized rectangles:
    - if within bound of a large rectangle, mark as car's ID symbol.
6. Once all rectangles have been sorted and placed, evaluate the car object:
    - Position : The center of the large rectangle.
    - Heading : the vector formed by the center of the large and medium rectangle.
    - Identification : the distance formed by the center of the medium and small rectangles. The distance determines the bit position.

The processed data is then multi-casted via ethernet to the controller computers to close the feedback loop. The whole cycle of identifying and transmitting data for 10 different markers averages 15ms per frame.

2.4 Control System

As shown in Fig. 2.1, the control computer determines the motion of each vehicle. The first generation testbed architecture has off-board computation, with all n-vehicle's control computed using a 3.0GHz PC. However, each vehicle runs an independent process on the computer, and for inter-process communications we used standard named pipes. There is an additional manager program responsible for receiving and distributing the tracking information to each of the n vehicle programs for control command calculation. Once all vehicle programs generate their respected commands, the manager packages the commands and send it via RS-232 cable to the Micro-Controller Board. The Micro-Controller will unpack the received commands and distribute them to the radio transmitters of the corresponding vehicles.

2.5 System Limitations

The remote PC based testbed comes with a few shortcomings due to the size, budget, and development time constraints:

1. The positioning and the feedback control are PC-based which gives rise to a highly centralized system, while distributed control are more desirable in real world implementation.

2. We directly utilize the radio transmitters that come with the RC cars for ease of integration, which results in a system update rate of only 13 Hz, significantly lower than camera frame rate. Hence merely 13 out of every 30 control decisions generated are being executed by the cars.

3. The cars also have limited range of motion, namely they can only perform

one of the six possible state when running: left forward/backward, straight forward/backward, and right forward/backward. This prevents the car from emulating some of the more sophisticated control algorithms.

To address these issues, we develop a second generation testbed which has the following features:

1. The vehicles are equipped with on-board processing and proximity sensing capabilities.

2. The wireless communication solution can handle the transmission of positioning information at 30 frame/sec.

3. The cars are capable of variable steering.

4. In addition to the existing Ackerman steered platform (cars), add a differential drive platform (e.g. tanks).

We will present details about the development in the next two chapters.

CHAPTER 3

Embedded Control Based Testbed: Vehicle Systems and Components

For embedded emulation testbed [3], we designed a system as shown in Figure 3.1, where the algorithm processing and motion control is managed by an embedded processor mounted on the vehicles instead on a remote PC as in the first generation testbed. The major modification takes place at the vehicles, while the positioning system and the testbed arena is largely unchanged. The embedded system results in the following advantages:

1. Enabling integration of embedded sensors so that we can implement more dynamic algorithms
2. Reduced dependency on non-mobile remote devices

This chapter will focus on the system specification and device selection of four key subsystems of the second generation vehicles:

1. Chassis
2. Micro-controller
3. Wireless communication
4. Embedded range sensor

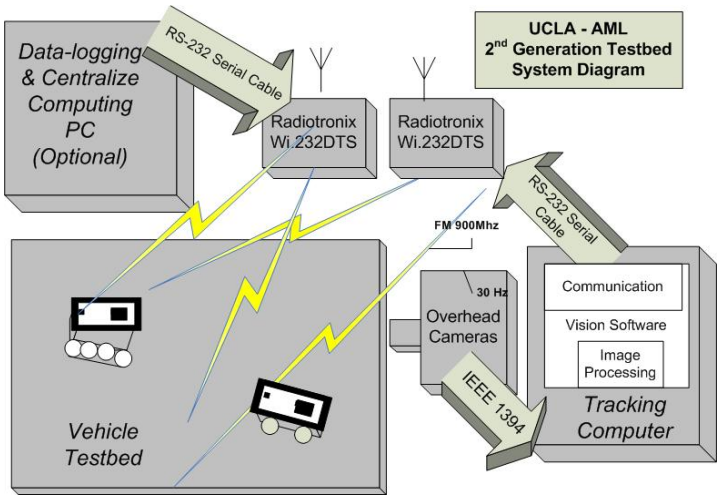

Figure 3.1: Testbed system diagram.

3.1 System Specification

We will first specify the requirements that the vehicles and their various subsystems need to satisfy. These requirements include physical dimensions imposed by the size and scale of the testbed and electrical specifications regarding the interface and communication between subsystems. In addition, we will list some desirable features that we wish to see on the corresponding device candidates.

3.1.1 Physical Requirements

The testbed size constraint requires that individual vehicle should not be too large, especially the length and the width, otherwise a team of vehicles would crowd the arena thus limiting the task capacity. It would be ideal to have vehicles of 1:64 scale just like the first generation vehicles. With such a scale, vehicles should be operable at the speeds range of $8.9 \sim 31.1 cm/s$, which corresponds to the speeds range of $20 \sim 70 mile/hr$ for actual size vehicles. We think that these are reasonable speeds at which an autonomous vehicle can be expected to run

Variables	dynamic range	unit	byte/frame	bandwidth
x-coordinate	$0 \sim 640$	pixels	2	480 bps
y-coordinate	$0 \sim 900$	pixels	2	480 bps
Heading angle	$0 \sim 6280$	Milli-radian	2	480 bps
speed	$0 \sim 120$	pixel/sec	1	240 bps
Application Specific Data			2	480 bps
sum			9	2160 bps

Table 3.1: Tracking Data to Be Communicated to Vehicles

when carrying out a mission. This requirement in turn puts a constraint on the height of the vehicle center of gravity to guarantee the vehicles lateral stability (the vehicle would not tip over sideways) when turning at a speed within the range mentioned above. Hence stacking devices one on top of another is not a viable approach, and we need to optimize the space to be consumed by each subsystem and make sure the size is close to the first generation vehicles while providing more decentralized capabilities.

3.1.2 Communication Requirements

We need real-time communication between the MCU and the image processing PC. The output of the image processing program includes the x, y coordinates, heading angle, vehicle speed, and possibly other data that is required by a specific application. Table 3.1 lists the variables and their required bandwidth. The sum of 2160 bps is the bandwidth required for a single vehicle without any overhead. With a system capacity designed at 10 vehicles and adding an estimate of 2 byte of overhead per frame, we have a bandwidth requirement at about 22.1 Kbps. Some applications might demand two way communication for data logging or

tracking feedback, thus the bandwidth requirement should be doubled to 44.2 Kbps. Finally, to leave a safety margin for possible retransmission or other issues, we require the communication module to handle more than 50 Kbps of effective data throughput. A vehicle can be as far as 5m away from the tracking PC when operating on the testbed. Thus we require the wireless module to have an indoor range of at least 10m to leave some margin.

3.1.3 Desirable Features

As mentioned in Chapter 1, the testbed is intended for validation of path planning algorithms and a prototyping tool for developing real-environment autonomous multi-vehicle system. However, there exist a broad range of path planning algorithms with various objectives. Some algorithms focus on high level strategies, namely the generation of discrete way-points without paying attention to how the intended vehicles travel to each waypoint. Some others take also the lower level motion planning into account but are geared toward one particular type or a subcategory of vehicles. By type of vehicles, we are referring to their kinematic designs (some refer to it as mobility configuration), which directly limits how the vehicle moves. Some common examples for ground vehicles are Differential Drive, Tricycle Drive, Ackerman Steering, Synchro Drive, Omnidirectional Drive, MDOF (Multi-Degree-of-Freedom) Vehicles, and Tracked Vehicles ([21]). Among these configurations, Ackerman steered vehicles (generally four-wheeled vehicles, where the inside front wheel is rotated to a sharper angle than the outside front wheel to prevent tire slippage when turning) are predominantly used in automotive industry; tracked vehicles (a special case of differential drive that is also know as skid steering, e.g. tanks) are also widely used in military applications. Thus these two types of ground vehicles are the most likely candidates for out-

door autonomous vehicles. Hence we want to include both of these two vehicle chassis and design a single electronic system that can be used on both types of vehicles

The Microsizer chassis used in our first generation testbed is Ackerman steered, but has a limited amount of steering freedom (the vehicle can only steer either extreme left or extreme right). Such limitation can induce large and prolonged error between the optimal path designed by a certain algorithm and the actual path the vehicle is capable of following. Hence variable steering is a crucial feature for an Ackerman steered vehicle so that an algorithm performance can be accurately reflected.

In addition, outdoor environment is more complex than an area of laboratory floor, thus a proximity sensor for the purpose of short-range obstacle detection is desired to enable users applying algorithms in signal processing, estimation, and dynamic path planning. To enable the DC motors to rotate both forwards or backwards, a motor driver circuit like the H-bridge is required. Thus we would prefer a micro-controller unit (MCU) with a built-in H-bridge as space can be saved for other peripherals.

3.2 Subsystem Specification and Device Selection

To minimize development time, we choose to integrate existing modules instead of building the subsystems from discrete components. Due to constraints on the testbed size, we adopt a design flow as follows. To prioritize the vehicle size, we would first choose the chassis without much consideration about the other three subsystems. Next we find a micro-controller module that is mountable on the selected chassis. With a chosen micro-controller, we then pick a wireless commu-

nication module that the micro-controller can support. Finally, we figure out the amount of space that is left and find a proximity sensor that can fit the available space. Aside from size, the selected modules should also have matching physical and electrical interfaces and satisfy the cost target. For each subsystem, we will first specify what we want to achieve with the selection of the corresponding modules, then compare the candidate selections we consider and discuss the decision we arrive at.

3.2.1 Chassis

As pointed out in previous sections, the chassis in the first generation testbed, namely the Hobbico Microsizers, come with a major shortcoming, which is the lack of proportional steering. But we are very satisfied with their size, which is 1:64 scale and about 6 cm in length, and their cost, which is $30 each. For the second generation of car chassis, we aim at finding one with similar size and cost but capable of variable steering.

Reference [27] is a hobbyist site that covers a variety of small-size radio controlled toy cars. As it turns out, there are only a few models that satisfy all above requirements. We ended up choosing the ZipZaps micro RC Special Edition car as our Ackerman steered vehicle chassis because it can be easily purchased at a low cost of $12.47. It comes with a 21,500 RPM motor and 12:1 gearing. It has rear wheel drive in either a forward or backward direction. It uses a potentiometer steering gauge that provides feedback to the steering controller. For the tracked vehicle, we chose the Ecoman R/C mini tank at the price of $10. The tank comes with 96:1 gearing and the ability to climb a 38° slope (however, with added electronic system, the tank may topple over at this angle due to higher center of gravity).

Vehicle	Dimensions $(L \times W \times H)$
first generation car	$(58 \times 30 \times 20)mm$
second generation car	$(68 \times 32 \times 18)mm$
second generation tank	$(51 \times 38 \times 20)mm$

Table 3.2: Physical Dimensions of the Vehicle Chassis

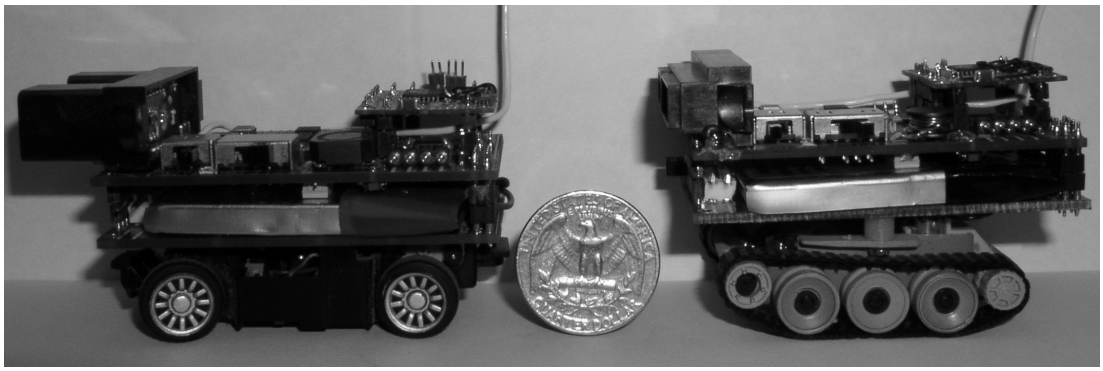

Figure 3.2: Left: car based vehicle; right: tank based vehicle.

Both vehicle platforms have exactly two degree of freedom (namely two motors; the car has one for actuation and the other for steering; the tank uses one for controlling the left belt and the other for the right belt) and come in similar size. We were able to design a unified electronic system that can be mounted on top of both thus reducing development cost and time. Table 3.2 compares the physical dimensions of chassis of the first, second generation cars and the tank. Figure 3.2 shows side images for a pair of assembled second generation car and tank.

3.2.2 Micro-Controller Module

Given the dimensions of the chassis chosen, we have a rough idea of the upper bound on size of the micro-controller module. Ideally, it should be as small as

Micro-controller	Intel Gumstix basix 400xm-bt	RabbitCore RCM 3410	MEGAbitty
Processor	Intel XScale PXA255	Rabbit 3000	Atmel ATmega8
Integer Range	32 bit	8 bit	8 bit
Size	$(80 \times 20)mm$	$(34 \times 29)mm$	$(23 \times 23)mm$
Retail Price	$169	$59	$39
Speed	400MHz	29.7MHz	16MHz
Data Storage	64MB SDRAM	256KB SRAM	1KB SRAM, 512B EEPROM
Program Storage	16MB Flash	256KB Flash	8KB Flash
Peripherals	many	8 channel ADC	8 channel ADC
Communication	Bluetooth, USB, UART I^2C, etc	5 serial ports	serial UART

Table 3.3: Comparison of Processing Module Candidates

possible in order to reduce the difficulty of placing peripherals around it. Additionally, low cost and ease of integration are also crucial factors in our decision-making process.

We consider a range of processing modules from general purpose mobile systems (e.g. PDA motherboards) to modules designed solely for customized embedded applications (e.g. washing machines). Table 3.3 compares major parameters of three of these modules. The Intel Gumstix has built-in Bluetooth capability [22], so it does not need an additional wireless communication module to transmit and receive data. However the motherboard has a length of $80mm$, which is longer than the vehicle chassis. Thus it becomes difficult to add a proximity

sensor as the sensor has to face the vehicle front. The retail price of $169 is actually very reasonable for all the features it has, but the amount of processing power and memory are an overkill for the purpose of this testbed. Nonetheless, the Gumstix could be a suitable choice in the future if the testbed is eventually upgraded to be based on either a larger indoor or a completely outdoor arena, where larger vehicle chassis and more complex control problems are considered.

The final choice is between the RCM 3410 and the MEGAbitty. The RCM 3410 is the smallest module from the RabbitCore micro-controller family by Rabbit Semiconductor [23]. Compared to the MEGAbitty, which is developed by the Portland Area Robotics Society [24], the RCM 3410 is almost twice as fast and comes with adequate memory for both data and programs. However, it covers an area that is almost twice as large as the MEGAbitty and costs about 50% more.

The MEGAbitty requires less area and costs less than the RCM 3410 and Gumstix. It is designed with motor-driven embedded application in mind as it includes two built-in 500mA H-bridge motor drivers. In conclusion, we choose the MEGAbitty as our processing module for its compact size and low cost. However, its limited memory would require the users to optimize the firmware. For control algorithms of interest to us, the memory consumed by C embedded compiler code was within the MEGAbitty capability.

3.2.3 Wireless Communication Module

The wireless communication module should both transmit and receive as we cannot fit two separate communication modules on one vehicle. Hence we will refer to it as the transceiver module. To select the right transceiver module, it is important to keep in mind the two devices, which the module would directly communicate with, namely the MEGAbitty micro-controller at one end and the

PC that processes the overhead images at the other end. The transceiver module should be able to interface with both.

For data communication interfaces, the MEGAbitty comes with a Serial Peripheral Interface (SPI) port, a byte-oriented Two-Wire serial Interface (TWI), and a Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART). On the other hand, most modern PCs generally come with USB ports, Ethernet ports, and phone-line Modems. Serial ports are gradually being replaced by USB ports on laptop computers, but they will continue to be supported in PCs since they cost minimally. Laptops without a built-in Serial port can use a USB-Serial dongle to gain RS232 access. Thus the simplest way to communicate between the MEGAbitty and a PC is via RS232 interfaces. Hence we are essentially looking for a transceiver module that can

1. Function as a wireless RS232 replacement;
2. Fit within our cost and size budget;
3. Achieve at least 50Kbps data throughput;
4. Achieve range of no less than 10m indoor.

Furthermore, we prefer the wireless module to use a frequency channel that would have less indoor interference from common laboratory electronic devices like microwave oven (2.4GHz), cordless telephones (900MHz phones are replaced by 2.4 and 5.8GHz phones), cellular phones (some use 900MHz), and wireless LAN router (mostly 2.4GHz).

Table 3.4 compares two candidate modules that satisfy the above requirements. The retail prices are quotes from a major on-line reseller, Mouser Electronics. The major trade-off here is one between size and price, as the two

Transceiver Modules	MaxStream [26] XBee^{TM} ZigBee OEM RF Module	Radiotronix [25] Wi.232DTS^{TM}
Size	$(24 \times 28)mm$	$(24 \times 21)mm$
Retail price	$22.50	$27.90
Frequency band	$2400 \sim 2484$MHz	$902 \sim 928$MHz
Receive current consumption	50mA (at 3.3V)	20mA (at 3.3V)
Transmit current consumption	45mA (at 3.3V)	57mA (at 3.3V)
Receiver sensitivity	-92 dBm	-100 dBm
Indoor range	up to 30 m	not specified
Outdoor range	up to 100 m	up to 400 m
Channel rapacity	16	32
RF data rate	up to 250Kbps	up to 152Kbps
Serial Interface	3V CMOS UART	3V CMOS UART
Interface data rate	up to 115.2Kbps	up to 115.2Kbps

Table 3.4: Comparison of Wireless Transceiver Module Candidates

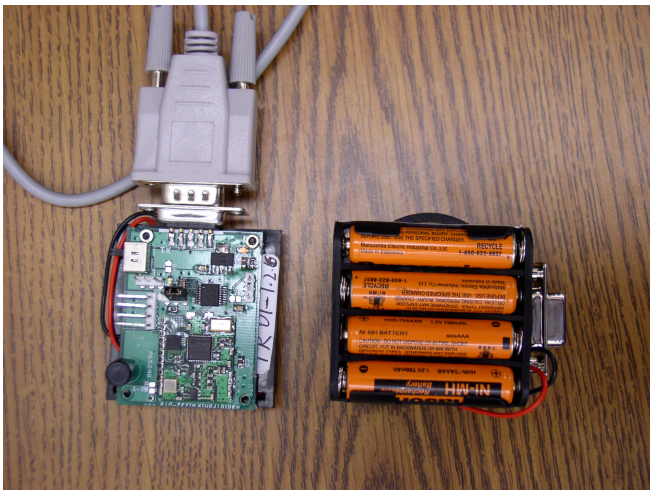

Figure 3.3: Front and back of the Wi.232DTS-integrated transceiver module connected to a RS232 cable

modules either satisfy or exceed our requirements. The Wi.232DTS [25] is 24% more expensive, while the XBee occupies 33% more area. In the end, we prefer the Wi.232DTS because the space is a hard constraint imposed by the selected chassis. Moreover, the 900MHz band used by Wi.232DTS has arguably less interference than the 2.4GHz band used by XBee.

We built a simple conversion circuit that connects the Wi.232DTS to a RS232 cable from our tracking PC, which can be seen in Figure 3.3. Our solution only introduces extra peripherals at the PC end of the system, where no space constraint exists.

3.2.4 Embedded Proximity Sensor

An embedded sensor is desired for providing proximity sensing thus enabling the vehicle to be operable in arenas with obstacles. The most desired sensor feature should be its compactness. Specifically, its longest dimension should be strictly less than the width of the vehicle, so that we can at least put it on top

Sensor	Devantech SRF10 Ultrasonic Range Finder	SHARP GP2Y0A02YK Distance Measuring Sensor
Type	Sonar	Optical
Size	$(32 \times 15 \times 10)mm$	$(29.5 \times 13 \times 21.6)mm$
Price	\$54	\$10.25
Range	$3 \sim 600$ cm	$20 \sim 150$ cm
Connection	Standard I^2C Bus	regular ADC
Operating Period	65 ms	40 ms

Table 3.5: Comparison of Ranger Sensor Candidates

of the vehicle without having part of it sticking out, which would cause problems when multiple vehicles are asked to run shoulder-to-shoulder. The cost, ease of integration, and sensor signal quality are lower priority than size, however they are very important. If a sensor can fit on the vehicle, while being relatively low-cost and easy to integrate, we would be willing to compromise on sensing performance and develop firmware that can compensate for any loss in signal quality.

Given the size and cost constraints, we found two candidate range detectors. Table 3.5 compares the basic parameters of the two sensors [24]: We can see that the sonar sensor has a much longer range than the IR sensor. However the sonar sensor is required to be mounted with a certain elevation from the ground or pointing upwards to avoid reflections from the carpet or ridges in a concrete floor. In the case of the SRF10, the manufacturer recommends $30cm$ from the ground, which would be impractical as our vehicles are only $4.3cm$ tall. Also the SRF10 features a relatively wide beam pattern as seen in Figure 3.4. This would not only cause ambiguity in terms of direction of response source but would also generate response from the floor due to its conical beam shape. Although there

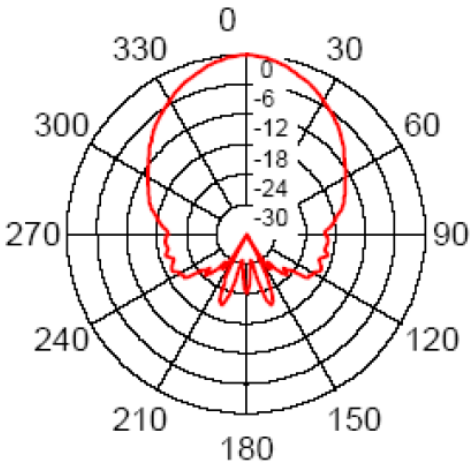

Figure 3.4: SRF 10 ultrasonic sensor beam pattern. The vertical axis has unit of dB.

are methods that can narrow the beam width, we would prefer a sensor that provides a simpler solution.

On the other hand, the IR sensor is equipped with a PSD (position sensing device) onto which the light is focused [4]. IR EM radiation is emitted via LED at the front of the sensor. The wavelength range in use is 850 nm$\pm$70 nm. The half intensity angle of the device is 1.5°. Figure 3.5 shows a schematic of the sensor layout. The IR sensor can respond to an obstacle in a shorter amount of time, which is beneficial for real-time obstacle detection and avoidance. It also features two models with different ranges and the GP2Y0A21YK described in Table 3.5 is the longer range one. The other model has a range of $10 \sim 80cm$ but higher accuracy. Both models have the same connector, which allows us to just plug in a sensor suitable for a specific application. Both long range and short range IR sensors have an analog voltage between 0 and 5V as output. Section 5.2.2 presents more information about the output characteristics and calibration.

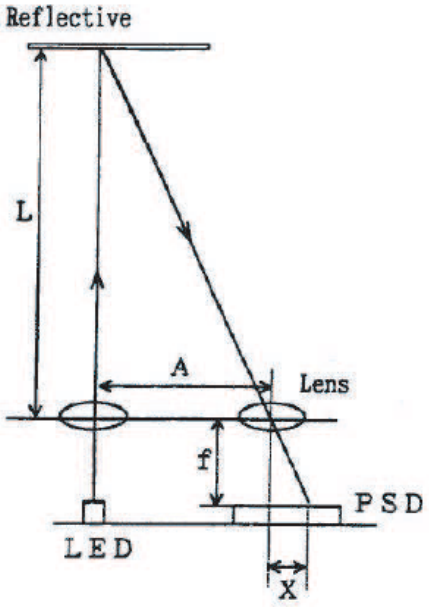

Figure 3.5: Schematic sensor layout and ray patterns.

3.3 Summary and System Integration

Table 3.6 summarizes the selection of each subsystem with its pros and cons. In essence, we favor size, cost, and practicability over other features like speed, memory, and signal quality. One may argue that such an approach will limit the capability of the testbed. We want to re-emphasize that the purpose of the testbed is not the novelty of the autonomous vehicles but rather in exploring what is manageable with such a compact package of technology.

Subsystem	Selection	Pros	Cons
Processing	MEGAbitty	size, cost	low memory
Communication	Wi.232DTS	size, freq band	cost, raw data rate
Range Sensor	SHARP IR sensor	size, cost, speed	range, signal quality

Table 3.6: List of Pros and Cons of Subsystem Selection

We design a two-deck (two printed circuit boards) structure to hold all the electronic subsystems (see Figure 3.2). The lower deck is directly mounted on the chassis. It holds the mechanics mount between the chassis and the upper deck. The main power reservoir, a single cell 3.7V 740 mAh Lithium polymer battery, is sandwiched between the upper and lower decks. Similar to the first generation, a fully charged battery can provide about 30 min of run-time. The upper deck holds the major component of the robot, the MEGAbitty, the communication module, the infrared sensor, power structure, and connectors. We utilize the step up regulator to provide an 8V power rail such that a low drop out 5V and 3.3V regulator can maintain stability for the communication unit and the processor module. Figure 3.6 shows a diagram of the car's on-board electronic system [3].

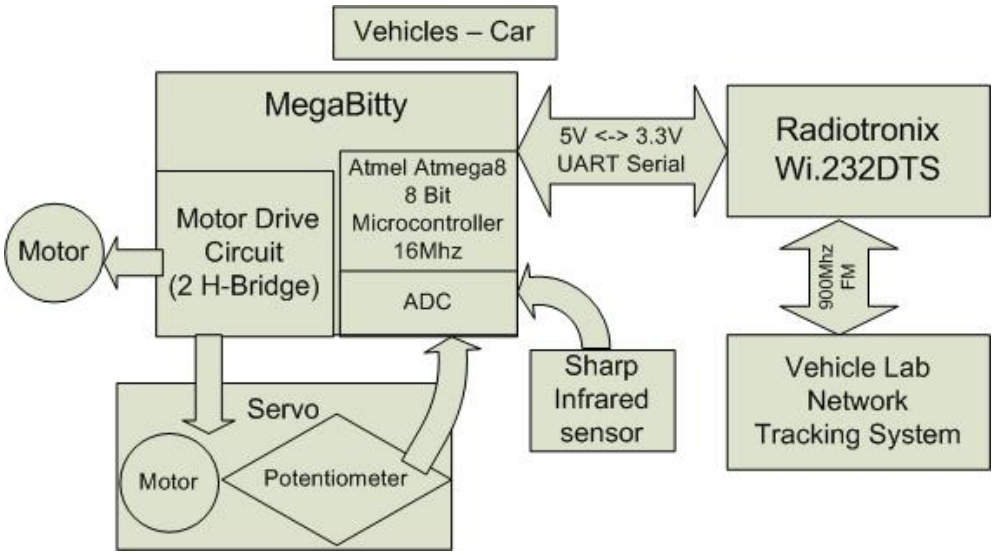

Figure 3.6: Diagram of vehicle on-board system. The tank system diagram is the same as above, except that it has another motor instead of the servo.

CHAPTER 4

Embedded Control Based Testbed: Embedded Firmware

Figure 4.1 shows the design our embedded firmware [3]. The scheduler runs periodic routines like a lower level motion controller and an IR sensor measurement acquisition, at a fixed frequency specified by the user. The interrupt-driven communication driver executes upon the receipt of a new frame of data. Both the communication driver and the IR measurement acquisition routine updates an array of global variables at the end of their run-time. The main routine reads the updated variables into the control algorithm and outputs the corresponding control decisions, which get written to some other global variables. These variables will be read by the lower level motion controller, and the lowest level of control values result.

4.1 Scheduler

A simple task scheduler regulates the update rate of the steering, motor drive control, and sensor readings. On start up, each task registers with the scheduler with its priority, update rate, and a callback function. Since the primary purpose of the scheduler is to update the various local control systems, none of the tasks are allowed to run for longer than the scheduler resolution of one ms.

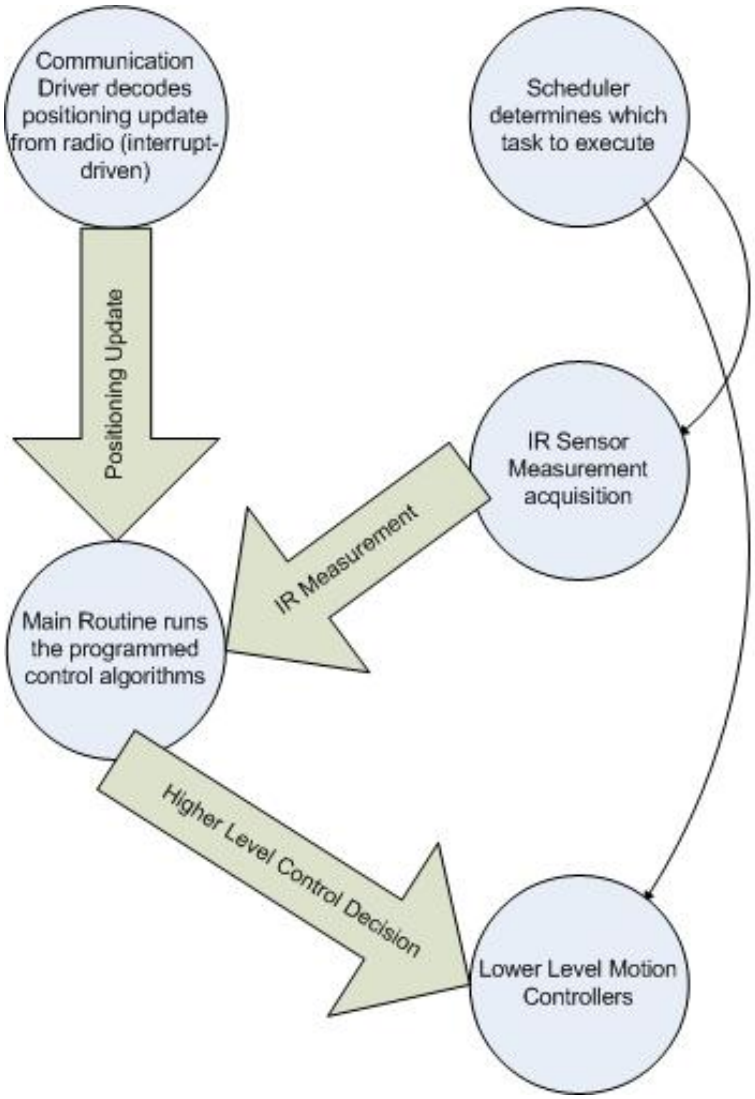

Figure 4.1: Embedded firmware flowchart.

The scheduled task can not perform any blocking calls; if the task is waiting for additional resources, it is required to reschedule itself to run again later. Scheduled task conflicts are resolved via priorities, if two tasks have the same priority, the tasks will be executed in the order of initial registration. One user task can be scheduled, but it has the lowest priority and can be preempted by any of the controller tasks to ensure correct operation of the vehicles.

4.2 Lower Level Motion Controllers

4.2.1 Basic Motion Controller on the Car.

Both the steering and speed motor are controlled by two pulse width modulation (PWM) channels via two H-Bridges. The speed motor is simply controlled by altering the pulse width, while the steering wheel control requires task scheduling for closed loop feedback control. A potentiometer feeds the analog voltage to the analog to digital converter (ADC) on the micro processor. Task scheduling allows the ADC call to release processing time to other jobs, while waiting for the conversion to be completed. The returned ADC value feeds into the classical proportional derivative controller operating at 250Hz. The steering angle has a total of 51°, 50° for far left, 25° for center and 0° for right, with an accuracy of one degree. The settling time for a 25° offset is 0.18s. Figure 4.2 illustrates the performance of the steering controller.

4.2.2 Basic Motion Controller on the Tank

The tank drives two belts independently, resulting in turns of arbitrary radius, while moving forward and backward. In practice we find it simpler to construct paths composed of either straight line motion or turning in place. A state machine

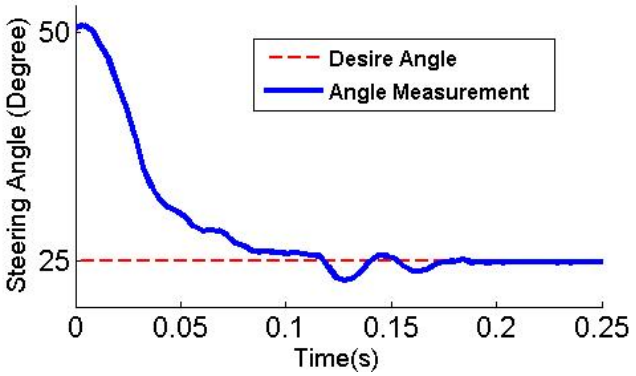

Figure 4.2: Steering response, from far left (50°) to center (25°), as measured by the potentiometer.

is responsible for the execution sequence of the two maneuvers. Both the speed and direction are the input parameters. We assume the direction has higher priority than speed. The complete motion state sequence is described in table 4.1. Both the straight line and turning maneuvers rely on the tracking system

	Current heading direction	New heading direction Requested
$Speed = 0$	No motion	1)Turn to desired direction 2) Stand
$Speed > 0$	Move Forward	1)First Turn to new direction 2)Move forward

Table 4.1: Tank Motion States

heading angle feedback. We use a simple proportional derivative feedback control scheme to stabilize the tank in the desired direction. Note that the left and right driver are not identical, some electrical and mechanical discrepancies exist. A proportional close loop controller alters the left and right drive strength to maintain a straight line motion.

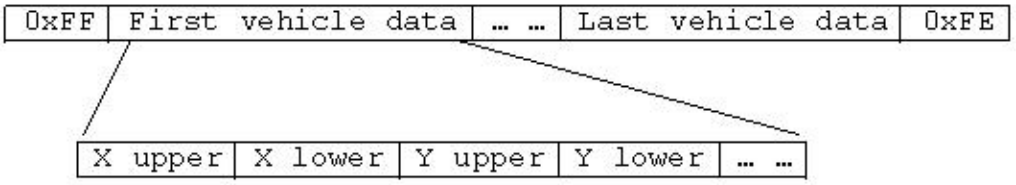

Figure 4.3: Formation of a positioning data frame.

4.3 Communication Driver

Vehicle tracking data is concatenated together and sent through the serial port as one package. To ensure successful transmission of the positioning data, we use the following scheme to package the data from multiple vehicles into a serial data frame. The frame starts with a one-byte delimiter and ends with a different one-byte delimiter to inform the decoder, where the useful data begins and ends. A specific pair of delimiters contain the purpose of the message being transmitted. For instance, we are currently using 0xFF and 0xFE delimiters to enclose a data message for broadcast of positioning data. To distinguish a data byte from a delimiter byte, we require all delimiters to have $MSB = 1$ and all data bytes to have a 0 as the MSB. Thus each data byte can only hold 7 bit of effective data. Data variables that exceeds 7 bit of data range (e.g. the x and y coordinates of the vehicle position) will be separated into two bytes (lowest 7 bits and the rest). Figure 4.3 shows the constitution of a frame of positioning data.

The receipt of the positioning data is interrupt driven. Upon receiving the whole packet, the vehicle can extract the tracking information for itself. We utilize the Windows API to interface with the serial port on the tracking computer.

The Wi.232DTS transceiver module has the flexibility to transmit and receive on separate channels, thus allow us to achieve a full duplex system. To make sure

the positioning data always gets delivery, we dedicate one particular channel to its transmission (on the PC side) and reception (on the vehicle side). A separate channel is used for the data-log function. When multiple vehicles are operating, they will share this channel to log their data. The transceiver module has built-in CSMA (Carrier Sense Multiple Access) to prevent transmission collision when more than one sources are transmitting. But our experiments reveal that data packets still get lost occasionally. To address this issue, we utilize a simple acknowledgment mechanism: the receiver is designed to immediately send back an acknowledgment upon receiving a data packet; if the transmitter receives the acknowledgment before a time-out, it continues to send the next packet, otherwise, it re-transmits the data packet. To make sure that the acknowledgment does not get lost, the same acknowledgment packet is transmitted five times in rapid succession.

4.4 Infrared Sensor Measurement Acquisition

The infrared sensor uses another ADC channel to provide an instantaneous digitized measurement. The execution rate is specified in the task scheduling, typically at 25Hz. Various filters can be coded to fit application needs. For the application described in this project, we implement a cumulative sum (CUSUM) algorithm for obstacle detection (see below). Another application, that of dynamic visibility, uses an ENO scheme method for processing spatial point cloud data obtained by the range sensors (see Section 5.2.2).

4.4.1 CUSUM Filter

The CUSUM filter is used for in-time reliable detection of obstacles while consuming minimal code space. It is implemented as follows. As a vehicle approaches an obstacle, sensor readings adjust from background noise to a level indicating the presence of the object. Figure 4.4 shows an example of raw sensor readings. To filter the signal, we use a particular version [36] of a standard cumulative sum algorithm [37]. Let X_n denote the raw sensor signal at time level n and μ denote the mean of the background noise when no obstacle is present. Define $Z_n = X_n - \mu - c$ where c is a fraction of the expected change in sensor reading due to the obstacle. Next define $W_n = \max(0, Z_n + W_{n-1})$. The calculated value W_n should remain around zero until the change of state occurs, at which point it ramps up. An example is shown in Figure 4.4. Once W_n passes a designated threshold (large enough to avoid false alarms with a high probability) the object is detected. Using the car chassis at 1/5 of the full throttle, we test the cumulative sum algorithm for different values of c ranging from 150 to 400. The results are well-reproduced in multiple trials. These values lie closely on a linear fit, therefore we use the $c = 200$ state in practice for the most advanced warning.

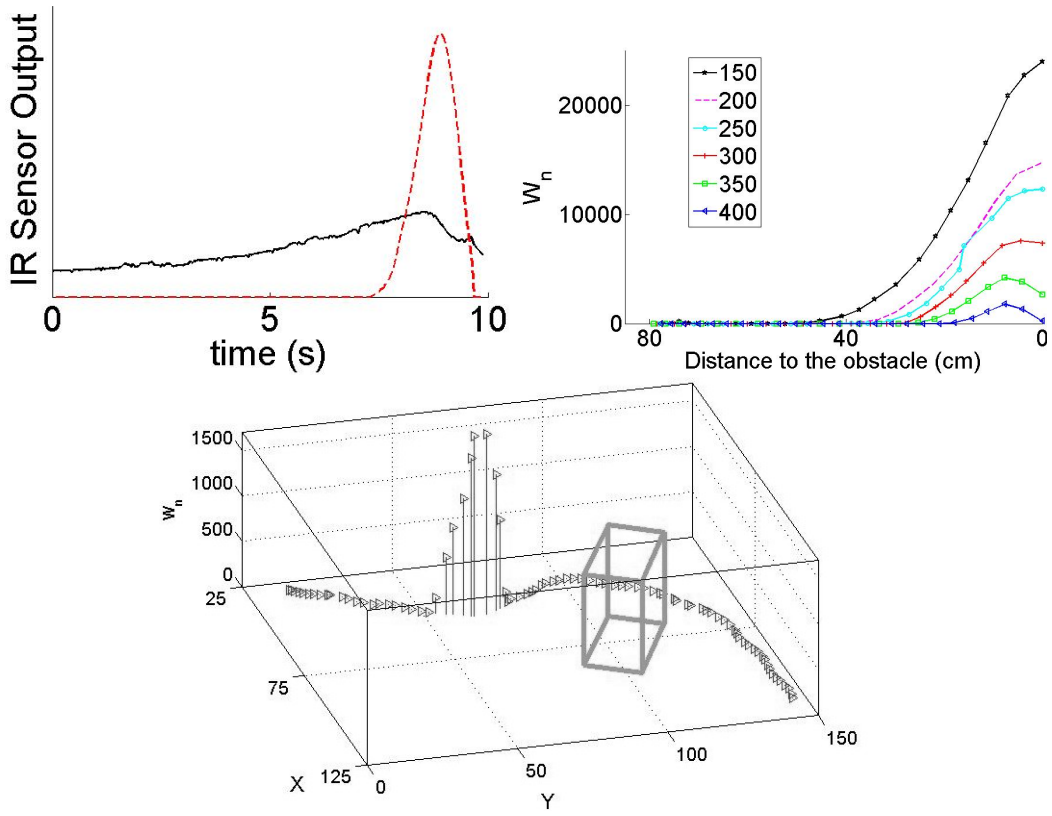

Figure 4.4: Cumulative sum algorithm applied to sensor data from a single car approaching an obstacle. Top left: raw data and cumulative sum; top right: cumulative sums for different choices of c; bottom: sample car path avoiding obstacle with cumulative sum sensor output.

4.5 Control Algorithms

The above functions are implemented in subroutines that can be called by a main routine that captures the particular control algorithms we want to emulate. Here is a list of control algorithms that we implemented:

1. A basic circle tracker
2. Split and remergence of a multi-vehicle swarm (homotopy)
3. Target seeking combined with dynamic barrier detection and avoidance
4. Pursuit-evasion game (motion camouflage)
5. Environment visibility exploration

Details about these implementation are presented in the next chapter.

CHAPTER 5

Implementation of Cooperative Control Algorithms

This chapter aims to provide several examples of testbed applications. For each application, we will first briefly introduce the algorithm under investigation. Next we will describe the adaption or modification made to the algorithm so that it is suitable for our testbed and vehicles. At last, we will present the results. Among these applications, some used the first generation testbed, while others utilized the second. The algorithms used in each example is also different. We sort them into categories so that comparisons between testbeds or algorithms can be made.

5.1 Cooperative Control Algorithms Based on Pairwise Interaction

Creating pairwise interactions among vehicles is a very important method in achieving multi-vehicle swarm and cohesive formations. We considered two kinds of pairwise interactions, with each one being suitable for each generation of testbeds. For the first generation testbed, we consider a pairwise interaction based on a generalized Morse potential. For the second generation testbed, we consider a Frenet-Serret frame based pairwise interaction that aim at achieving balance among three geometric optimizations. It is worth noting that one of the

three optimizations has a similar effect to the potential function in the first algorithm. Comparatively speaking, the first algorithm can work with motion-wise more constrained vehicles but demands considerably higher processing speed and more program memory (which the first generation testbed can satisfy since its control program resides on a 3GHz PC). The second algorithm, however, can be implemented with more space-efficient program but requires more subtlety (or say degree of freedom) in vehicle steering.

5.1.1 Morse Potential Based Pairwise Interaction

5.1.1.1 Basic Theory

We consider a general potential flow for a particle at position $\vec{r}_i$ [2], at distance $r_i = |\vec{r}_i|$ from the origin, subject to dissipation γ and to pairwise interactions U:

$$\dot{\vec{r}}_i = -\gamma \vec{\nabla}_i \sum_{j \neq i} U(r_{i,j}). \tag{5.1}$$

Here $r_{i,j} \equiv |\vec{r}_i - \vec{r}_j|$ denotes the distance between agents i, j. For simplicity in the remainder of this paper we will set $\gamma = 1$.

Our control algorithm adopts a generalized Morse potential that decays at infinite distances, as would be expected for systems of vehicles with a limited communication range:

$$U(r_{i,j}) = -C_a e^{-r_{i,j}/\ell_a} + C_r e^{-r_{i,j}/\ell_r}. \tag{5.2}$$

Here, C_a, C_r represent the strength of the attractive and repulsive potentials, and ℓ_a, ℓ_r their length scales, respectively.

5.1.1.2 Testbed Adaptation

The models described in Eq. 5.1 cannot be directly applied to a platform of autonomous vehicles due to mechanical constraints. The vehicles in our first generation testbed consist of Dubins micro-cars with fixed speed and fixed left and right turning radii. The first constraint implies our dynamical system must be described as first order. The only independent variable denoting agent i is its heading angle with respect to a fixed orientation we define as θ_i. The Dubins vehicles interact with each other by means of the Morse potential of Eq. 5.2 with variable parameters C_a, C_r, ℓ_a, ℓ_r. Due to the fixed turning radii, the interactions cannot directly control θ_i and an appropriate control algorithm must be devised. For each vehicle then, we measure the angle γ_i between vehicle heading and the total force $\vec{F}_i$ it experiences, as given by the right hand side of Eq. 5.1 and as shown in Figure 5.1. Vehicle i then changes direction only if $|\gamma_i| > \Gamma$, where Γ is an angular threshold $0 \leq \Gamma \leq \pi$. The equations of motion are as follows:

$$\dot{x}_i = \alpha \cos\theta_i, \quad \dot{y}_i = \alpha \sin\theta_i \tag{5.3}$$

$$\dot{\theta}_i = \begin{cases} \frac{\alpha}{R_L} & \text{if } \gamma_i > \Gamma \quad \text{(left turn)}, \\ -\frac{\alpha}{R_R} & \text{if } \gamma_i < -\Gamma \quad \text{(right turn)}, \\ \frac{\alpha}{R_S} & \text{otherwise.} \end{cases} \tag{5.4}$$

Here, α is the speed of the vehicle, and R_L, R_R are the left and right turning radii, respectively. R_S is the deviation radius. In the ideal case $R_L = R_R$ and $R_S = \infty$, so that vehicle direction is unaffected for $|\gamma_i| < \Gamma$. Because of alignment asymmetries in general $R_L \neq R_R$ and R_S is a large but finite number. Vehicular motion proceeds along the direction specified by the heading parameter θ_i until the turning commands $\dot{\theta}_i$ are given.

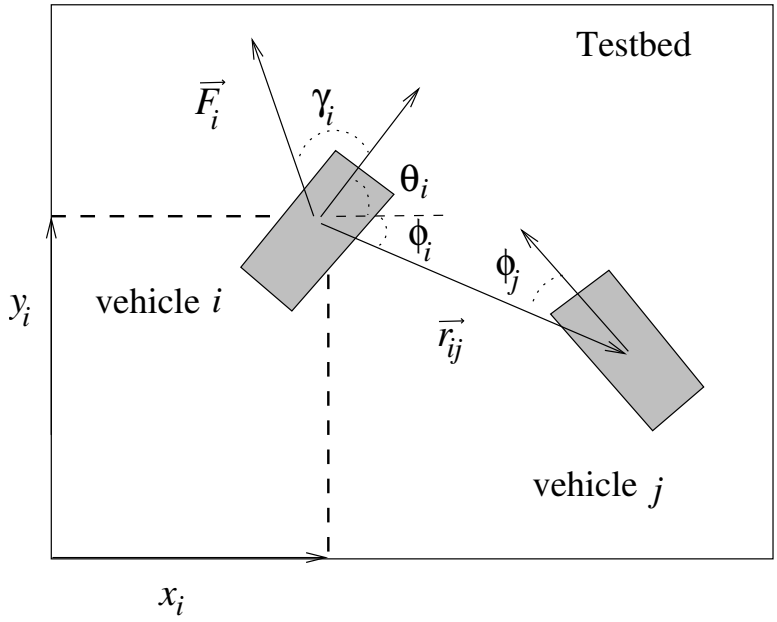

Figure 5.1: Definition of variables for vehicle i: The heading is denoted by θ_i, the angle between its direction of motion and the x axis of the testbed. $\vec{F_i}$ is the interaction force it experiences due to all other vehicles. This direction defines an angle γ_i with the heading direction. Vehicle i is at a distance $\vec{r}_{i,j}$ from vehicle j and the angles ϕ_i and ϕ_j here shown are used in the collision avoidance scheme described in the text. The origin of the reference coordinate system is fixed at the left-lower corner of the testbed. All vehicular angles, γ_i, θ_i, ϕ_i, are defined in $[\pi, -\pi)$.

A crucial point is that the interaction potential in Eq. 5.2 is soft-core and does not prevent vehicles from colliding. In fact, even hard-core potentials cannot avoid collisions due to communication delays, errors in position information, and the finite turning radius of the vehicles. The repulsive range may be increased to initiate turning at larger inter-vehicle distances. This however, would significantly affect pattern formation and the emergence of cooperative aggregates would be unlikely. Instead, we add an additional collision avoidance algorithm to address short range interactions. We use a 'wait and go' scheme for vehicles closer than a cutoff distance r_c. For vehicles i, j at distance $\vec{r}_{i,j}$ such that $r_{i,j} < r_c$, we define the angles ϕ_i, ϕ_j between their main axis and $\vec{r}_{i,j}$, as shown in Figure 5.1. If $\phi_i < \phi_j$ vehicle i will pause while vehicle j veers away, until $r_{i,j} > r_c$. The cutoff distance r_c in the control algorithm acts as an effective hard-core potential. If $\phi_i = \phi_j$ any one of the vehicles (in our simulations the one with a higher labeling index) will pause and let the other proceed. When $\phi_i, \phi_j \simeq 0$ the 'wait and go' scheme cannot avoid collision as shown in Figure 5.2, and an alternate algorithm is invoked. For vehicles i and j we define the angle $\Omega_{i,j}$ between $\vec{r}_{i,j}$ and the segment joining their opposite front edges measured from $\max\{\phi_i, \phi_j\}$ as shown in Figure 5.2. If $\max\{\phi_i, \phi_j\} < \Omega$, where Ω is an angular threshold $0 \leq \Omega \leq \pi/2$, then the vehicle closer to the center of the testbed is veered towards the center and the other in the opposite direction.

5.1.1.3 Experimental Results

In this section we study the behavior and performance scaling of a set of Dubins vehicles [18] controlled by the first order laws based on the model in the previous section [2]. We consider testbed implementation for a small number of vehicles. However, it is also possible to incorporate the presence of many virtual vehicles

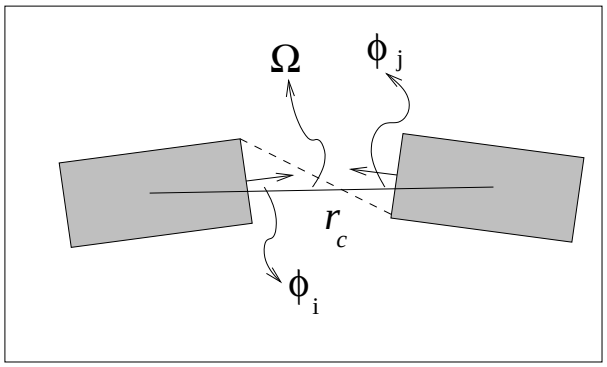

Figure 5.2: Collision avoidance failure: the angles ϕ_i and ϕ_j are too small and vehicles i and j collide even if one of them should pause. An additional algorithm is required to steer the vehicles away from each other and is described in the text. It relies on the angle $\Omega_{i,j}$ here depicted.

in practical testbed applications and study the effects of larger vehicle numbers on the actual ones.

Our first generation testbed has three working vehicles. A virtual leader moves around an ellipse with semi-major axis approximately 15 times the vehicle length. There is some variability in vehicle speed. To address this issue, the position of the leader is checked against the distance to the closest vehicle. If the distance becomes larger than a certain threshold d_t, the leader will pause; otherwise, it will move at its intrinsic speed, We select our parameters as follows: $\ell_r = 5.7\,\mathrm{cm}$, $\ell_a = 95.2\,\mathrm{cm}$, and $C = 1.667$. Note that these parameters correspond to a potential in the 'catastrophic regime' of [28]. For potential parameters in the H-stable regime we have not been able to realize stable configurations of vehicular aggregation due, in part, to the constant speed of the vehicles. The leader interacts with the vehicles according to the same Morse potential used for vehicle interaction. When leading more than one vehicle, the leader's contribution to the potential is increased 1.1 times and 2.1 times the vehicular potential for the two-vehicle and

the three-vehicle experiments, respectively.

5.1.1.4 One vehicle Follows a Leader

The parameters mentioned above provide short-range repulsion and long-range attraction resulting in an equilibrium separation. Figure 5.3 shows results for d_t near the equilibrium r_{eq}, calculated to be $r_{eq} = 20.2\,\text{cm}$. Running tests with $d_t = 20.5\,\text{cm}$, $d_t = 20.2\,\text{cm}$, and $d_t = 20.0\,\text{cm}$, we note that leader-following becomes ineffective for d_t below r_{eq}.

5.1.1.5 Two Vehicles Follow a Leader

The vehicles are found to alternate between a snake-like competing behavior as shown in Figure 5.4-top left and a stable gliding behavior as shown in Figure 5.4-top right. The stable behavior emerges when one vehicle trails the other and they form a rather flat triangle with the leader that glides around the ellipse as shown in Figure 5.4-bottom.

5.1.1.6 Three Vehicles Follow a Leader

The vehicles still alternate between competing and gliding behaviors as in the two-vehicle case as shown in Figure 5.5-top left. When stable motion emerges, the vehicles and the leader form a stretched quadrilateral that glides around the ellipse as shown in Figure 5.5-top right and bottom. We note that fragmentation can sometimes occur due to the stretched formation, as the attraction between the two slower vehicles overwhelms the long-range attraction from the leader.

To reduce such occurrences, we can enhance the leader attraction by increasing its weight. Also, both group cohesion and stabilization of the above examples can

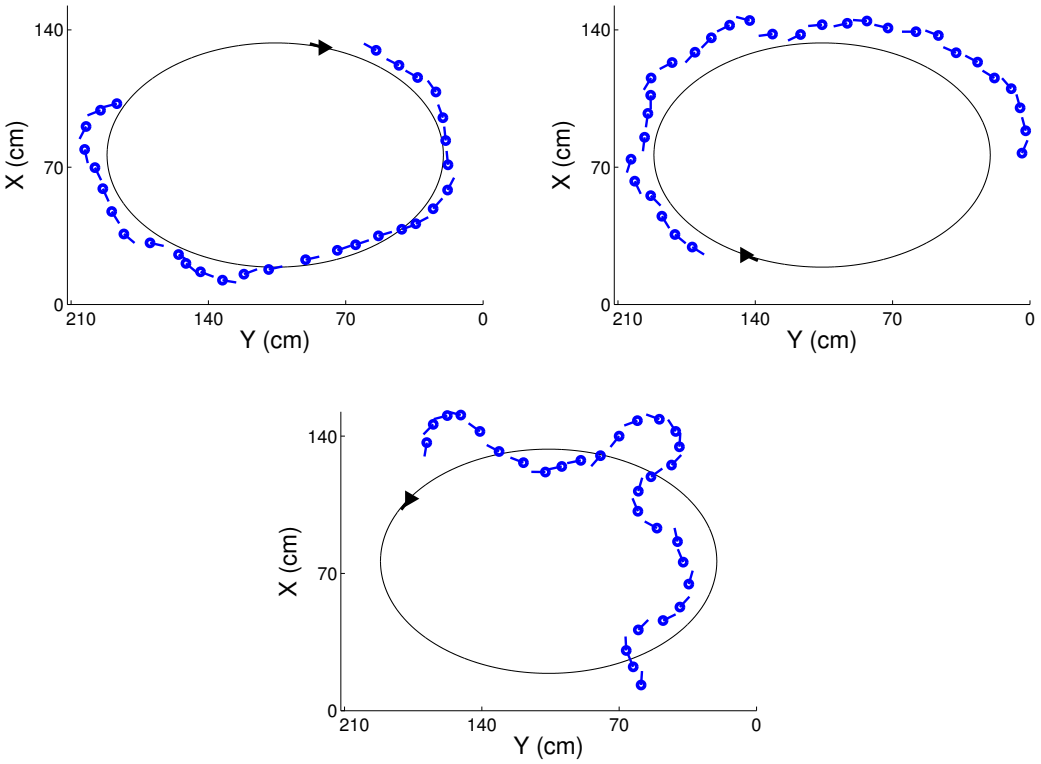

Figure 5.3: Vehicular motion: these panels show fragments of the vehicle's trajectory when it tries to follow a virtual leader along an elliptical path. The vehicle is unstable when d_t is decreased below $r_{eq} = 20.2\,\text{cm}$. Top left: $d_t = 20.5\,\text{cm}$; top right: $d_t = 20.2\,\text{cm}$; bottom: $d_t = 20.0\,\text{cm}$.

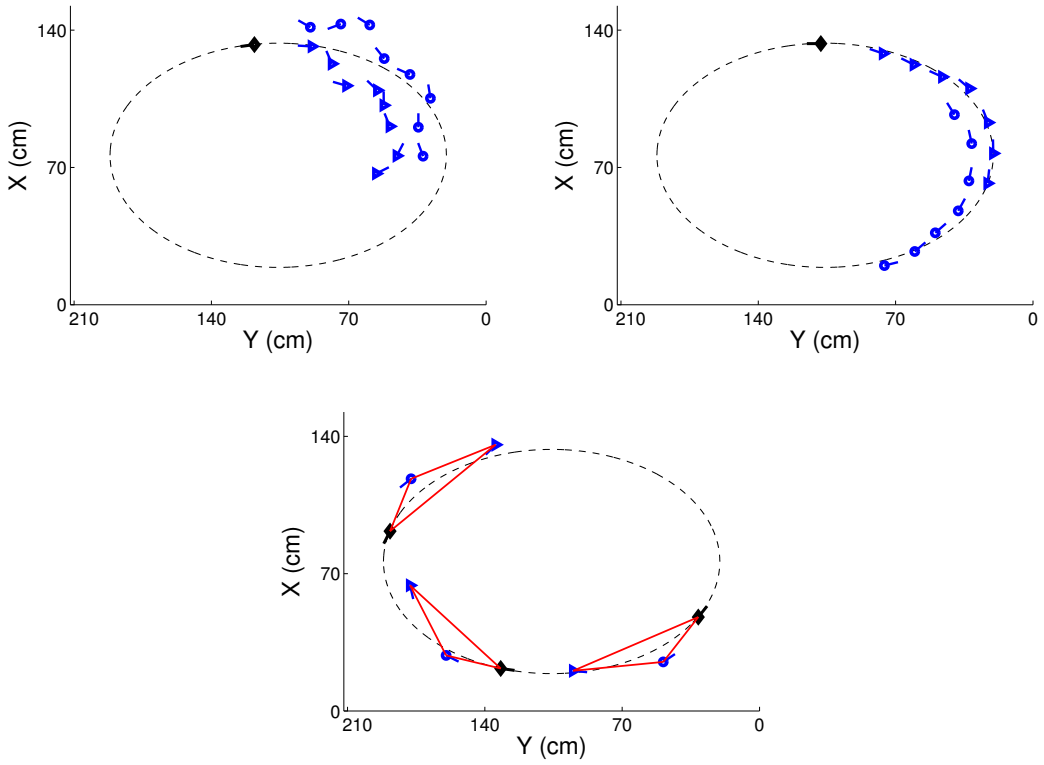

Figure 5.4: Two vehicles try to follow a virtual leader along an elliptical path. Top left: two vehicles exhibit snake-like motion as they compete for the optimal spot behind the virtual leader; top right and bottom: the vehicles' motion becomes stable when one trails the other, and they form a flat triangle with the leader, which glides along the path.

be realized by imposing rigid formations for the vehicle group as in [29]. Note, however, that in the absence of a rigid structure, even though the vehicles shift position with respect to each other, they are able to maintain a coherent group as they follow the leader around the track.

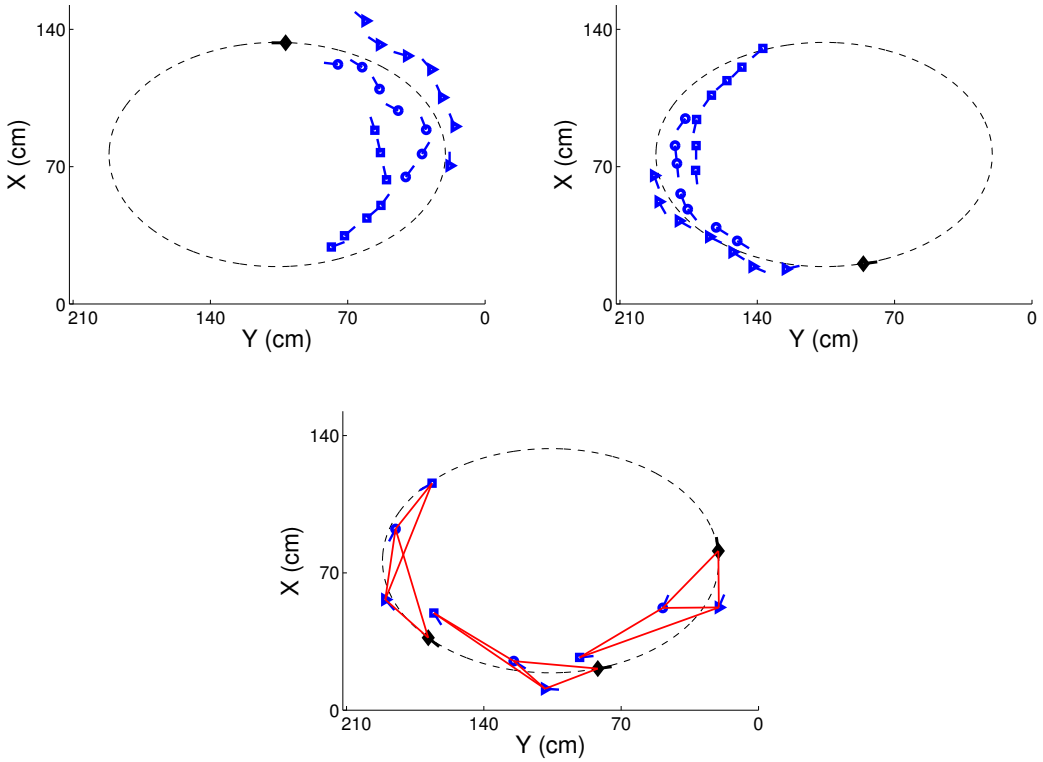

Figure 5.5: Three vehicles try to follow a virtual leader along an elliptical path. Top left: vehicles exhibit snake-like motion when they level with each other; top right: the formation becomes stable when one trails another. bottom: the vehicles and the leader form a stretched quadrilateral that glides along the path.

5.1.2 Frenet-Serret Frame Based Pairwise Interaction

We present four different applications in this section since they are all inspired by the original work of Justh and Krishnaprasad [30]. They are a basic circle tracker, homotopy (a swarm of agents split into two sub-swarm and later remerge to form back a swarm), target seeking combined with dynamic obstacle detection and avoidance using the newly added on-board IR range sensor, and motion camouflage (pursuit and evasion). The second and third applications, however, are more directly based on the algorithms by Morgan and Schwartz [32]. The motion camouflage application is directly based on another of Justh and Krishnaprasad publication [31]. But before showing the results, we will first provide some introduction of the theory that leads up to these applications.

5.1.2.1 Basic Theory

According to [30], the motion of a vehicle can be described by a differentiable curve $z(s) \in \mathbb{R}^2$ parametrized by arc length. Let x denote the unit vector from the position of the vehicle, in the direction of the tangent vector dz/ds and $y = x^{\perp}$ is positively oriented with respect to x. The motion of each vehicle is modeled by

$$\dot{z_k} = x_k, \quad \dot{x_k} = u_k y_k, \quad \dot{y_k} = -u_k x_k \tag{5.5}$$

where k is the vehicle index. The vehicles move at unit speed and the curvature of the k^{th} vehicle path is the scalar u_k. The control law is specified by dynamically changing u_k to create pairwise interactions between vehicles. Define $u_k = \sum_{j \neq k} u_{jk}$ where

$$\begin{aligned} u_{jk} = [&-\eta \left(\frac{r_{jk}}{|r_{jk}|} \cdot x_k \right) \left(\frac{r_{jk}}{|r_{jk}|} \cdot y_k \right) \\ &-f\left(|r_{jk}|\right) \left(\frac{r_{jk}}{|r_{jk}|} \cdot y_k \right) + \mu x_j \cdot y_k] \end{aligned} \tag{5.6}$$

where $r_{jk} \equiv z_k - z_j$, $f(|r_{jk}|) = \alpha[1 - \left(\frac{r_0}{|r_{jk}|}\right)^2]$, and $\eta = \eta(|r|)$, $\mu = \mu(|r|)$, $\alpha = \alpha(|r|)$ are specified functions.

The term $-\eta\left(\frac{r_{jk}}{|r_{jk}|} \cdot x_k\right)\left(\frac{r_{jk}}{|r_{jk}|} \cdot y_k\right)$ aligns the vehicles perpendicular to their common baseline. The potential function $f(|r_{jk}|)$ regulates the spacing between the vehicles and the term $\mu x_j \cdot y_k$ steers the vehicles to a common orientation. This control law requires position information of other agents within a neighborhood as described below.

5.1.2.2 A Circle Following Control Law

The above control law can be extended to maintain a car following the circumference of a circle of a given radius [30]. We use a two-car model in which one of the car is fixed at the center of the circle. We set $\mu = 0$, $\alpha = \eta = 1$, r_0 equal to the circle radius r and $|r|$ equal to the distance from the car to the center of the circle. The basic control law reduces to

$$u = \eta(|r|)(\frac{-r}{|r|}x)(\frac{-r}{|r|}y) - f(|r|)(\frac{-r}{|r|} \cdot y)$$

where $r = r_2 - r_1$ is the distance between the cars.

5.1.2.3 A Local Coupling and Leader Following Control Law

Local coupling is manifested by limiting the visible distance range of each vehicle to a neighborhood around that vehicle [32]. To ensure swarming, any two vehicles have to be within a distance with each other, either directly or via other vehicles. The local coupling control law is

$$u_k^{GL} = \sum_{j \neq k} c(|r_{jk}|, 0, w)\, u_{jk} \tag{5.7}$$

where u_{jk} is (5.6) and

$$c\left(|r_{jk}|, q, w\right) = \begin{cases} 1 & if |r_{jk}| < w, \\ q & otherwise. \end{cases} \tag{5.8}$$

Such local coupling is advantageous for large numbers of agents due to the scalability of the communication step.

A designated leader vehicle steers a swarm in a particular direction. The rest of the vehicles (follower vehicles) follow accordingly, by using the local coupling control law, with stronger coupling between follower and leader vehicles:

$$u_k^{follower} = c\left(|r_{l(k)k}|, 0, w\right) l_c u_{l(k)k} \tag{5.9}$$

$$+ \sum_{j \neq k, l(k)} c\left(|r_{jk}|, 0, w\right) u_{jk}, \tag{5.10}$$

where l_c is a leader coupling constant and $l(k)$ is the index of the leader vehicle closest to the k^{th} vehicle. The control law for the leader vehicles depends on the particular application.

5.1.2.4 A Homotopy Control Law

In order to transition smoothly from a global control law to a local coupling leader following control law, a homotopy parameter λ is introduced [32]. The homotopy control law $u_k(\lambda)$, $0 \leq \lambda \leq 1$ satisfies

$$u_k(\lambda = 0) = u_k^G, \quad u_k(\lambda = 1) = u_k^L \tag{5.11}$$

where u_k^G is the global control law and u_k^L is the local control law. If there are m leader vehicles, the homotopy control law for the $n-m$ follower agents is

$$\begin{aligned} u_k^{follower}(\lambda) = \\ c\left(|r_{l(k)k}|, 1-\lambda, w\right)\left[\left(l_c-1\right)\lambda+1\right]u_{l(k)k} \\ + \sum_{j\neq k, l(k)} c\left(|r_{jk}|, 1-\lambda, w\right)u_{jk}, \end{aligned} \tag{5.12}$$

where u_{jk} is given by Eq. (5.6) and $l_c \gg 1$ is a coupling constant which strongly attracts the followers strongly to their leaders. The leader agent homotopy control law is

$$u_k^{leader}(\lambda) = \sum_{j\neq k}[u_{jk}(1-\lambda) + \frac{s_k}{n-1}\lambda]. \tag{5.13}$$

When more than one leader is present, local coupling can be exploited to separate a swarm into two sub-swarms, as leader vehicles drive in different directions and follower vehicles follow their respective closest leaders. Such an example is demonstrated on the testbed in the next subsection.

5.1.2.5 A Target Seeking Control Law

To move towards a specified target, the kth vehicle uses

$$\begin{aligned} u_k = \sum_{j\neq k}\Bigg[c\left(|r_{jk}|, 0, w\right)\left(-\alpha\left(1-\left(\frac{r_0}{|r_{jk}|}\right)^2\right)\right) \\ \left(r_{jk}\cdot y_k\right)\Bigg] + \gamma\alpha\left(1-\left(\frac{r_0}{|\overline{r}_k|}\right)^2\right)\left(\overline{r}_k\cdot y_k\right), \end{aligned} \tag{5.14}$$

where $\overline{r}_k$ is the vector from the location of the kth agent to the target, and γ is a weighting constant [32]. Only the first term involves interaction between the vehicles in order to avoid collisions. The second term directs each vehicle to move toward the target. This control law does not guarantee swarming, but if

the agents start out in a swarm-like orientation, it is likely that they will stay together.

5.1.2.6 A Barrier Avoidance Control Law

Consider a fixed convex object in the plane, the exterior of which is defined by a set of m points $b_i \in \mathbb{R}^2$ [32]. The average barrier direction vector is computed as

$$v_k = \begin{cases} \sum_{i=1}^{m}[(b_i - y_k)c(|b_i - y_k|, 0, w)]/|\sum_{i=1}^{m}[(b_i - y_k)c(|b_i - y_k|, 0, w)]|, \\ \qquad if \quad |\sum_{i=1}^{m}[(b_i - y_k)c(|b_i - y_k|, 0, w)]| \neq 0, \\ \\ 0, \qquad otherwise. \end{cases} \tag{5.15}$$

where $c(\cdot)$ is a step cutoff function that is zero outside of a specified radius. The control law for barrier avoidance then consists of adding the term $\left(sign\left(v_k \cdot y_k^{\perp}\right)\right)(v_k \cdot y_k)$ to control law in (10). This term orients the vehicle perpendicular to v_k and the sign steers the vehicle away from the average barrier direction.

5.1.2.7 A Motion Camouflage Control Law

Motion camouflage is a sly technique that allows a pursuer to approach a prey while appearing to remain stationary from the viewpoint of the prey. To accomplish this, the pursuer follows a way such that it always lies on the line that connects the pursuer and fixed point. Here, we are trying to achieve three systems: a pursuer-pursuee system, a evader-evadee system, and a pursuer-evader system. In [31], the authors studied the pursuer's motion camouflage strategy for a pursuer-pursuee system, which is based on the same Frenet Serret framework

mentioned above. The strategy for a pursuer is provided as

$$u_p = -\mu \left(\frac{\mathbf{r}}{|\mathbf{r}|} \cdot \dot{\mathbf{r}}^{\perp} \right) \tag{5.16}$$

where $\mathbf{r}$ is the baseline vector between the pursuer and the evader and $\mathbf{r} = \mathbf{r}_p - \mathbf{r}_e$. Let the strategy for the evader be

$$u_e = \beta \left(\frac{\mathbf{r}}{|\mathbf{r}|} \cdot \dot{\mathbf{r}}^{\perp} \right) \tag{5.17}$$

Here $\beta, \mu \geq 0$.

It is also assumed that the pursuer and evadee move at unit speed, while the pursuee and evader moves at a constant speed $\nu < 1$. Using the fact that $\dot{\mathbf{r}}^{\perp} = \mathbf{y}_p - \nu \mathbf{y}_e$, we can obtain scalar forms for the above strategies:

$$u_p = -\mu \left(\cos\theta_p - \nu \cos\theta_e \right)$$

and

$$u_e = -\beta \left(\cos\theta_p - \nu \cos\theta_e \right)$$

where θ_p is the angle between $\mathbf{r}$ and $\mathbf{y}_p$, θ_e is the angle between $\mathbf{r}$ and $\mathbf{y}_e$. Both can be easily calculated from the vehicles' headings.

5.1.2.8 Algorithm Adaptation: Steering Angle

To map the curvature control u_k to a car's desired steering angle ϕ_k, we use the formula $\rho = L_{car} / \tan\phi_k$ [35], where ρ is the car turning radius. Thus, the steering angle of a car can be calculated as:

$$\phi_k = \tan^{-1}\left(L_{car}/\rho \right) = \tan^{-1}\left(L_{car} u_k \right). \tag{5.18}$$

L_{car} is measured to be 4 cm. The steering law assumes the cars to have unit speed thus we scale L_{car} according to the actual vehicular speed.

However, the vehicle's wheels have limited turning range from $-25°$ to $25°$. Hence, we constrain ϕ_k as:

$$\phi_{k,\text{ constrained}} = \begin{cases} -25°, & \text{if } \phi_k \leq -25°, \\ \phi_k, & \text{if } -25° < \phi_k < 25°, \\ 25°, & \text{if } \phi_k \geq 25°. \end{cases} \tag{5.19}$$

5.1.2.9 Implementation of a Circle Tracker

In this section, we present circle tracking implementation using two generation of cars separately. Figure 5.6 shows a circle following implementation on a first generation car using a Lyapunov based algorithm [1]. However this is already the best circle tracker we manage, which has a 12% average distance error from the reference circle. Figure 5.7 shows an implementation of the Frenet-Serret frame based circle following control law a second generation car [3]. A single car follows three different circular paths in sequence with a change in r upon reaching a particular heading. As one can see, the average distance error has been significantly reduced comparing to the implementation on the first generation car and the car trajectory is much smoother. We attribute this improvement mainly to the new found ability of variable steering.

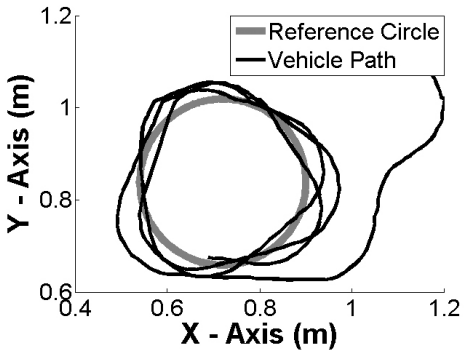

Figure 5.6: Measured vehicle path about a reference circle using the first generation testbed.

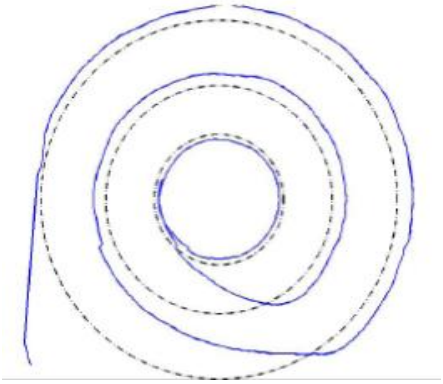

Figure 5.7: A single car path (solid line) tracing, in sequence, concentric circles (dashed lines) of radii 65.5cm, 41.7cm, and 28.3cm using the second generation testbed.

5.1.2.10 Implementation of Homotopy Control Law

Due to the testbed physical size constraints, the homotopy control law discussed above is modified to a sequence of three stages [3]. In stage one, two leaders are designated, with steering program $u_k^{leader}(\lambda = 0)$ given by Eq. (5.13) which is reduced to global control law Eq. (5.6). The regular global control law, which in this case includes only the leaders. Four followers are given control laws $u_k^{follower}(\lambda = 0)$ from Eq. 5.12. In stage two, the leaders are switched to $u_k^{leader}(\lambda = 1)$ where s_k is an explicit non-interactive control law causing the leaders to drive away from each other. The control law of the followers remains the same, causing a propagation of two swarms led by separate leaders. In stage 3, the global control law Eq. 5.6 between the leaders is reinstated, causing the two swarms to merge back together. Note that the control law Eq. 5.6 does not define the direction of the swarm, the heading direction is arbitrary and the outcome of the re-merging of the group can result in a different overall heading after merging, as seen in the two experiments shown in Figure 5.8.

5.1.2.11 Implementation of Target Seeking, Dynamic Barrier Detection and Avoidance

We combine the target seeking (Section 5.1.2.5), barrier avoidance (Section 5.1.2.6) and cumulative sum algorithm (Section 4.4.1) for obstacle detection to generate a path dynamically [3]. A box ($W23 \times L6 \times H13$) cm is placed along the path of a swarm of four cars moving toward a common target. All cars have priori knowledge of the box dimensions and orientation, but not its location. The box's widest surface is perpendicular to the cars' initial heading direction. We designate two front cars as observers. They use the on board long range infrared sensor to estimate the box location. Once the obstacle is located, data is sent to

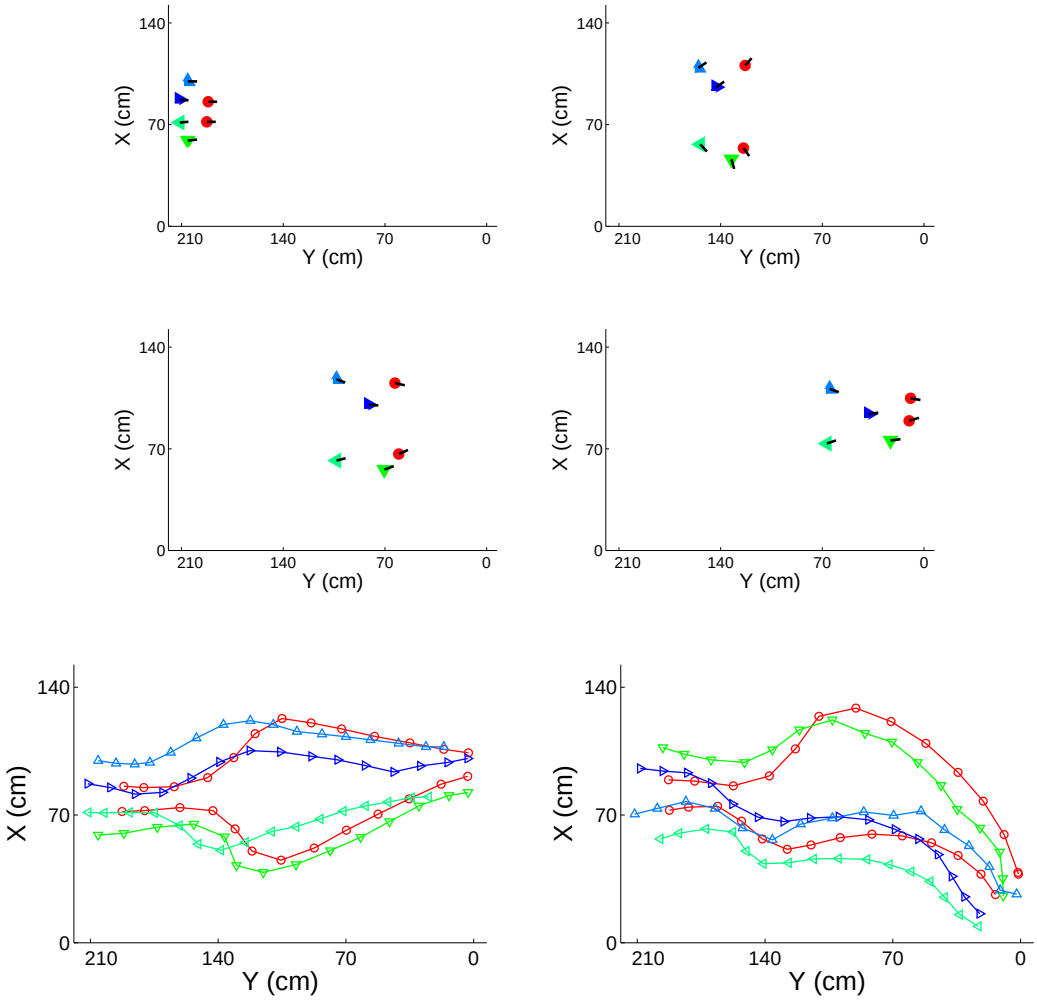

Figure 5.8: The top four figures show testbed data of a time sequence of six cars performing the maneuver described in Section 5.1.2.10. The trajectories are shown in the middle figure. The circular dots show the position of the leaders; the triangular dots show the position of the followers. The bottom figure shows trajectories for second experiment in which one follower goes with the top leader and three go with the bottom leader. In this second run, the overall heading of the group is different after the merger. In the implementation, the tracking information is interpreted in pixel domain and the corresponding parameters are $\mu = \alpha = \eta = 1$ and $w = 600$ for every agents, $r_0 = 50$ for leaders, and $r_0 = 40$ and $l_c = 20$ for followers.

the computing station, which generates a virtual barrier location and dimension. This information is distributed to all four cars. The virtual barrier is constructed as follows: let $\vec{p_i}$ be the point on the obstacle detected by the i^{th} observer. A rectangle of length $(2L_{obstacle} - |\vec{p_1} - \vec{p_2}|)$ and width $2W_{obstacle}$ is constructed with the center of the barrier side, closest to the swarm, located at $(\vec{p_1} + \vec{p_2})/2$. Since we only have two measures of distance to the obstacle, we extend the barrier to ensure collision avoidance. To avoid crossing paths and collision between cars after passing the obstacle, we reduce the barrier term weight when the car passes a certain distance from the barrier. Figure 5.9 shows the trajectories and snapshots of the implementation.

5.1.2.12 Implementation of Motion Camouflage

We performed experiments with 2 vehicles for 3 cases: pursuer-pursuee, evader-evadee, and pursuer-evader, where the pursuee and evadee are neutral agents that just follow some arbitrary controls [5]. In all three cases, the parameters assume values as: $L = 0.067, \mu = 15$, and $\nu = 0.75$. For the cases involving pursuers, the fact that $\nu < 1$ always results in the captures of the evader or pursuee. From the vehicular trajectory plots (Figure 5.10, 5.11, and 5.12), we can clearly see that the baseline vector $\mathbf{r}$ is able to maintain its initial orientation with only minor deviations. We attribute these deviations to the sundial mechanical conditions of the vehicles and noise in the vehicular heading estimate, which has a maximum error of approximately $\pm 2.5°$.

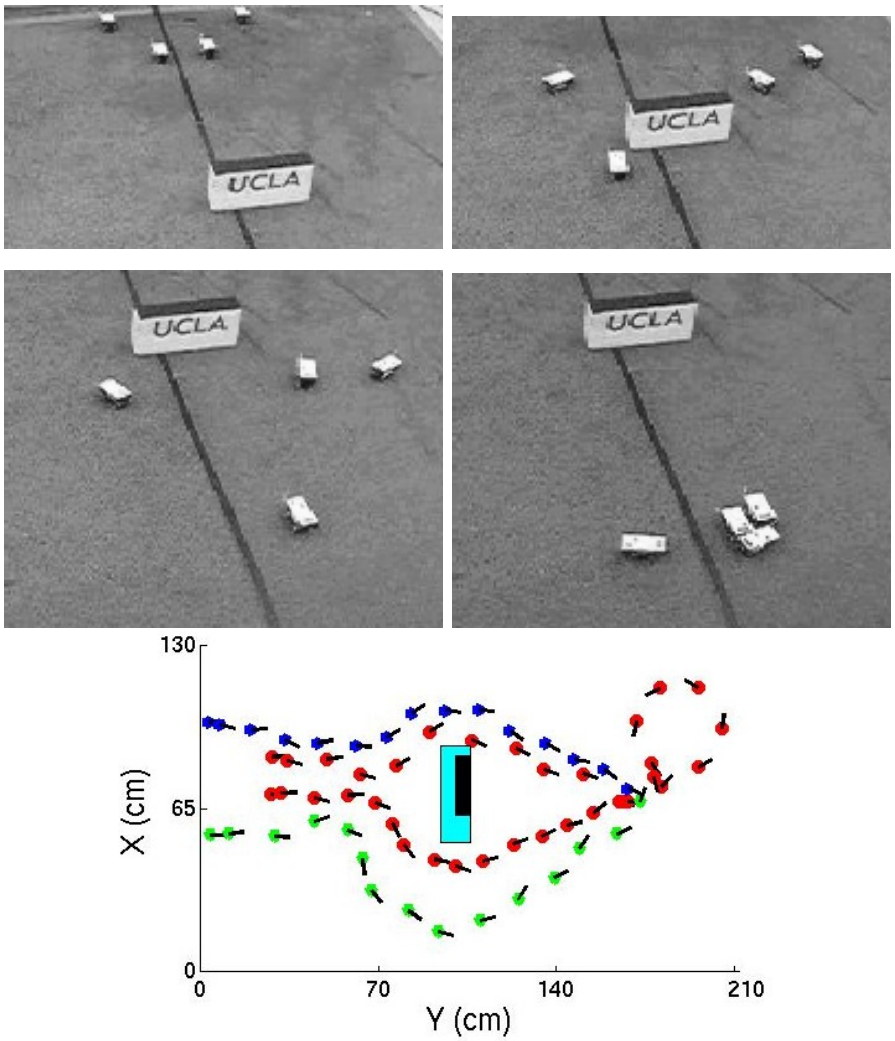

Figure 5.9: Target seeking with barrier avoidance. The top four panels show snapshots, at different times, of a single demonstration of the maneuver. The time progresses from top left to bottom right. The bottom figures shows trajectories of the cars compared to both the actual barrier (dark) and the larger virtual barrier (light) as computed from the range sensors of the observers. In the implementation, the tracking information is interpreted in pixel domain and the corresponding parameters are $\alpha = 1$, $\gamma = 25$, $r_0 = 60$, and $w = 100$ for the target seeking term. The weighting constant for the barrier avoidance term is 85 and is reduced to 1 after passing the barrier with $w = 180$.

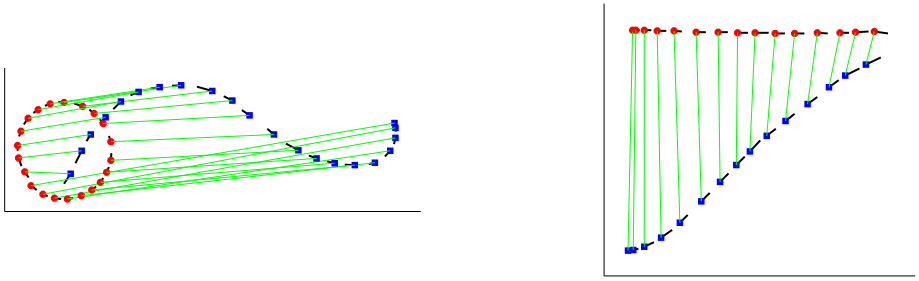

Figure 5.10: The trajectories of a pursuer-pursuee pair. The round dots represent the pursuee, as the square dots represent the pursuer. Left: the pursuee traverses a circle; right: the pursuee traverses a straight line.

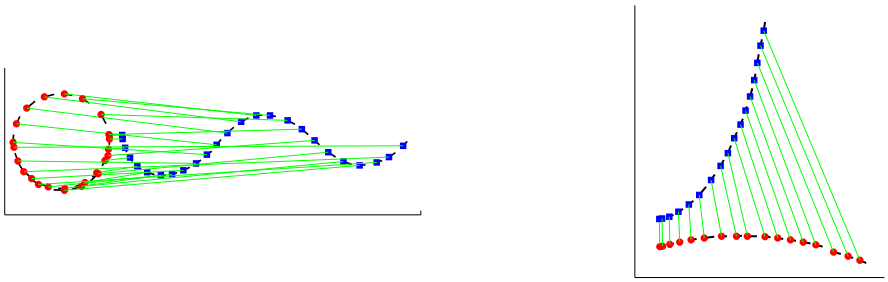

Figure 5.11: The trajectories of a evader-evadee pair. The round dots represent the evadee, as the square dots represent the evader. Left: the evadee traverses a circle; right: the evadee traverses a line.

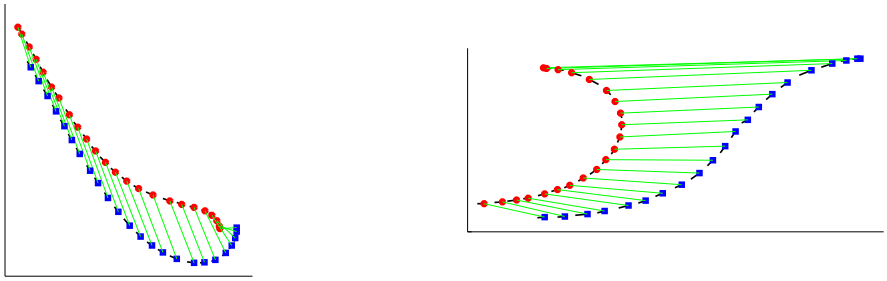

Figure 5.12: The above figures show the trajectories of a pursuer-evader pair. The round dots represent the evader, as the square dots represent the pursuer. Left: the pursuer and the evader are positioned close-by while facing opposite directions initially; right: the pursuer and the evader are positioned relatively far away while facing each other initially.

5.2 Other Applications

In this section, we present two applications that are not based on pairwise interactions between vehicles: UAV routing emulation [1] and environment visibility exploration [4].

5.2.1 UAV Routing

As an example of the capabilities of the first generation testbed, we report experimental results from the implementation of a recent algorithm for UAV routing [38] and compare the observed performance to analytically-derived bounds. Consider a number of vehicles with constant speed and bounded curvature that must visit stochastically-generated targets in a convex, compact two dimensional plane. Targets appear according to a spatio-temporal Poisson process, uniformly in space. We want to minimize the expected waiting time between the appear-

ance of a target and the time it is visited. We limit our analysis to the case in which targets appear infrequently. In this paper, we associate to each vehicle a

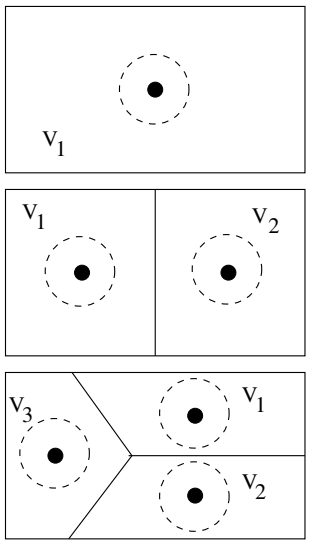

Figure 5.13: Loitering regions (Voronoi partition) for the testbed with 1 (top), 2 (middle), and 3 (bottom) vehicles. The dashed lines show the loitering path for each vehicle.

unique and distinct Voronoi cell. Figure 5.13 shows the Voronoi cells used for our testbed with one, two, and three vehicles.

Since our vehicles have only three available forward or backward input states (left, right, straight) a hybrid control [39] method makes sense for implementing both the loitering patterns and straight line motion to targets. In our implementation of a point-to-point controller, we deviate from [39] in that when the heading error is within 10° of the reference angle we use a straight line motion. This less aggressive control law yields a smoother path to the target on our testbed vehicle. We also implement a circular path tracking using the distance error to the desire radius. We select the input that would minimizes the distance error when applied to the vehicle model. We use heuristic and some switching logic to track the circle. By being less aggressive in moving the wheel from side to side, we

gain a smoother path along a polygon inside the desired circle. Figure 5.14 (top) shows a complete flowchart for the algorithm.

We now present data from the testbed running experiments of the algorithm discussed above. Initially, vehicles are placed at any point and they automatically drive toward their respective loitering state. Due to the size constraint of the testbed, the loitering state for each vehicles are now defined as a circular trajectory of $\alpha\rho$ with $\alpha \approx 1.5$. The corresponding γ is 5.54. Once a target appears each vehicles checks whether the target lies within its patrol domain and the vehicle responsible for the target performs a point-to-point maneuver to approach the target. Once the vehicle reaches the target, it returns to its designated loitering state. We generate targets randomly within the testbed arena (using a uniform distribution). In order to guarantee the light load case, targets generated are put into queue only after the vehicle returns to its loitering state. In practice our target generation rate is less than $\frac{1}{6}Hz$ and the average service rate is smaller than 6 seconds (see Figure 5.14). To demonstrate that the performance is within the theoretical upper and lower bounds proposed in [38], we perform the experiment using 1, 2, and 3 vehicles. In each continuous run, we sequentially generate approximately 200 targets. Less than 10% of targets generate involve a vehicle approach that either requires more than two passes to reach the target or involves out of bounds motion of the vehicle. We exclude these multi-pass (greater than 2) values from the data collected and compute the arithmetic mean of the service time for the remaining targets. As is expected, the average time decreases as the number of vehicles increases and lies within the bounds of the theory described in the previous section. The data is shown in Figure 5.14. Figure 5.13 shows the loitering regions and loitering circles used in the experiment.

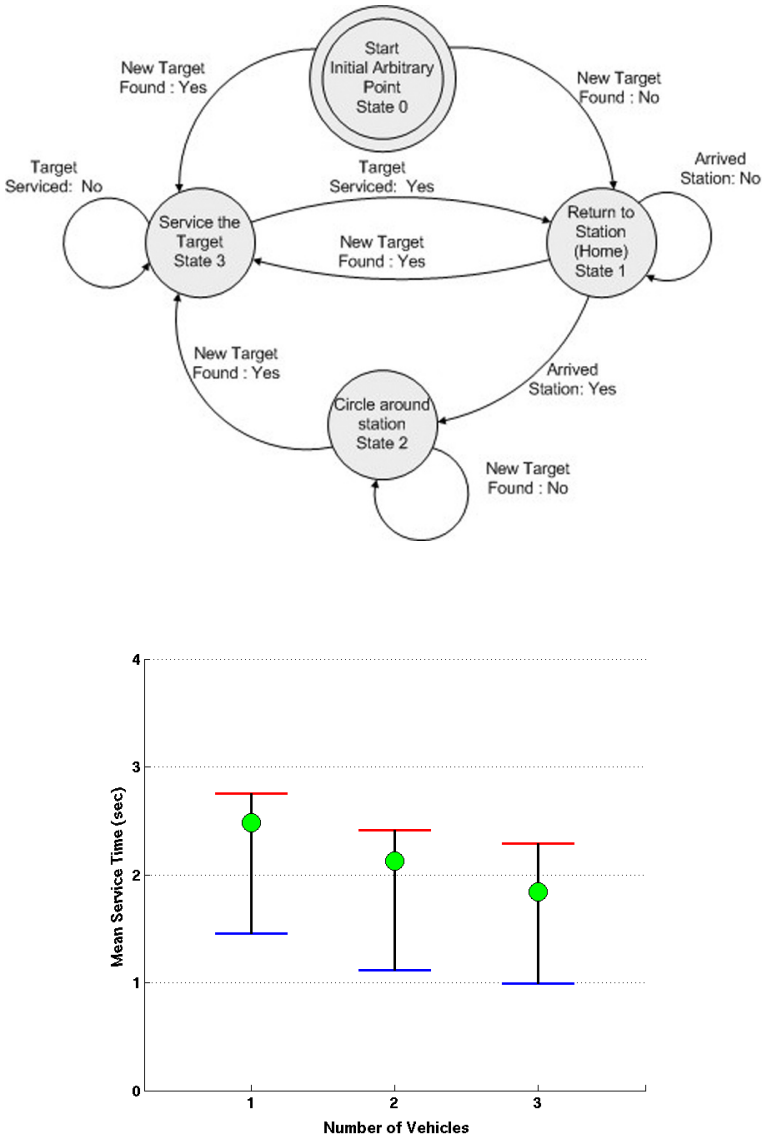

Figure 5.14: Top: vehicle control state diagram. Bottom: average service time of randomly generated targets in the light load case. Data from three experiments with respectively one vehicle, two vehicles, and three vehicles. The lower and upper bars denote theoretical bounds as proposed in [38].

5.2.2 Environment Visibility Exploration

In this application, we consider the problem of exploration of an unknown bounded two-dimensional region which may contain obstacles. To achieve such goal, we utilized second generation tank vehicles equipped with long-range IR sensors and apply a recent environment mapping algorithm [40] based on *Essentially Non-Oscillatory* (ENO) interpolation [41]. Details about the algorithm can be found in [4].

5.2.2.1 Range Sensor Calibration

Here we describe the process of sensor data calibration. The sensor takes readings at a rate 25 Hz. Sensor readings are produced by Analog Digital Converter (ADC), which outputs values proportional to voltage output (V×204.8). The raw data obtained from the sensor over a period of several seconds is depicted in Figure 5.15. We use the most frequent reading as the value at current position. In Figure 5.16 we plot values at given distances from the object measured along the normal to the surface of an object.

In Figure 5.17 we show several range curves constructed from different angles to the surface of the object. As one can see from Figure 5.17, the range calibration curves are shifted with respect to one another for different viewing angles (upward, when the object is viewed from the right, downward, when object is viewed from the left). This results in the same sensor output value for two different sensor positions. For example, sensor output at a distance of 90 cm from the object at an angle -85° to the normal to the surface is the same as the sensor output at a distance of 45 cm at an angle +75° and yet the same as the output at a distance of 60 cm along the normal to the surface. If we take as a reference the range curve measured along the normal to the surface of reflective object,

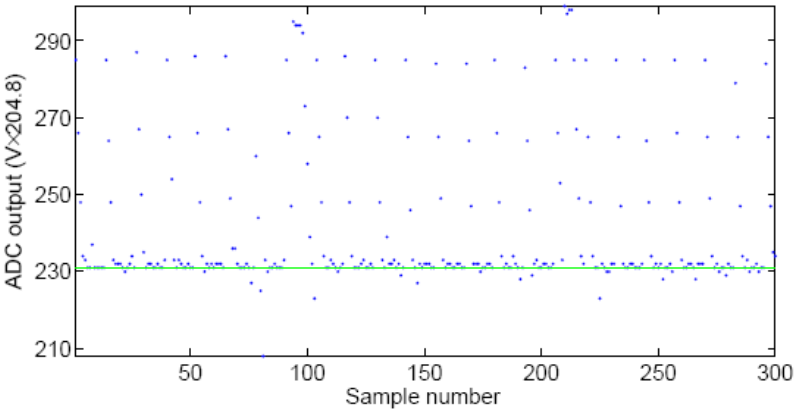

Figure 5.15: Sensor ADC output 60 cm away from the object; the green line corresponds to the most frequent value.

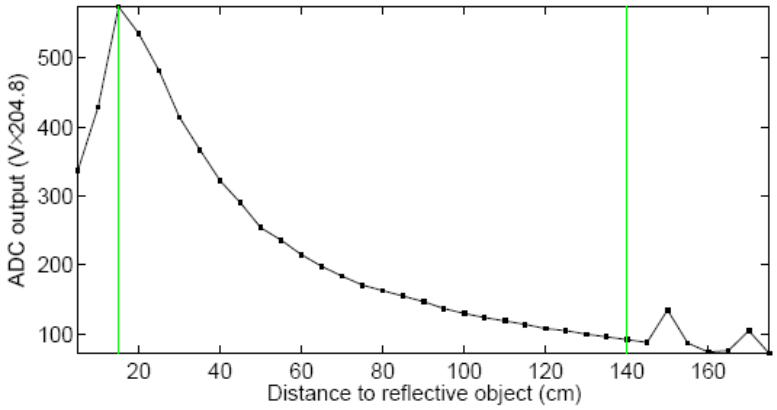

Figure 5.16: Sensor ADC output corresponding to distance to reflective object measured along the normal to the surface; the green vertical lines mark working sensor range.

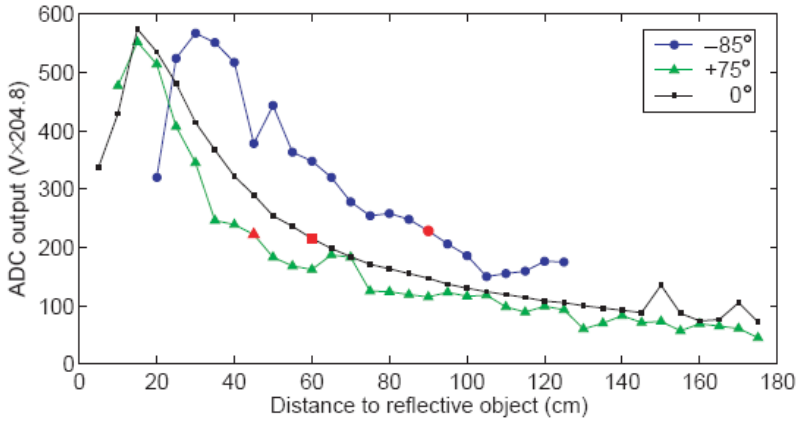

Figure 5.17: Sensor ADC output corresponding to distance to reflective object measured along different angles to the normal to the surface; the red marks correspond to points on the range curves with similar sensor output.

we obtain inaccuracies when looking at an object from a different angle. For example, one can see from Figure 5.18 the tilt in the measured surface position with respect to the actual one. The results may be improved by taking several measurements along a given direction. This way we can find a matching range curve from which we can deduce the distance to the object and the incident angle. However, this solution is too expensive and thus we did not implement it. In addition, we note that the shift is only significant when looking at an object from the right. Thus, a path-planning algorithm is modified with a bias towards moving in a counter-clockwise manner. See Figure 5.18 for an example.

We implement a multi-vehicle environment mapping algorithm based on a Visibility Interpolation formulation introduced in [40]. The algorithm does not require any shape priors for the occluding objects. We use two boxes as our sample obstacles for easy representation. The positions, shapes, and quantities of obstacles are unknowns. Two tank-based vehicles equipped with the range sensors are initially positioned on the testbed floor outside the obstacles. Each

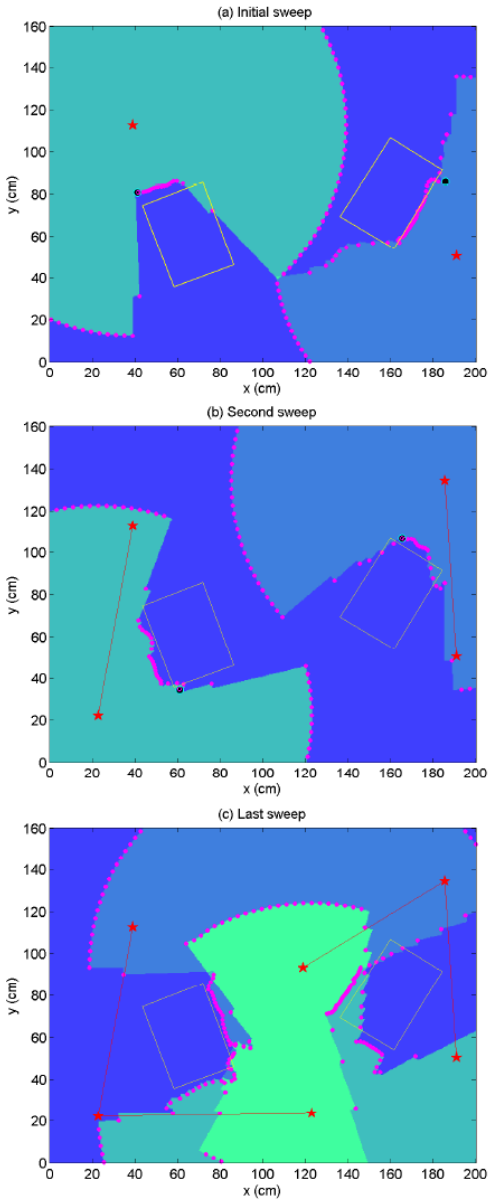

Figure 5.18: Exploration of environment with 2 observers. The red stars are observers positions; the magenta circles are the sensor output converted to range data; the big dark circles are the next edges to be approached; yellow boxes are the actual obstacle outlines; the dark regions are currently invisible; the light regions are currently visible.

tank makes a 360° sweep to gather range data from its surrounding environment. About 80 samples are taken in one sweep. Each sweep takes less then a minute to complete. Then, a visibility map and next position of each vehicle is computed off-board based on sensor output. The next *observer's* position is transmitted to the robots and they proceed to collect data from new vantage point. This process is repeated until the whole region has been explored as in Algorithm 2 above. In the example, exploration took two steps by each *observer*. The obtained range data is fit to the range calibration curve in Figure 5.16 via cubic interpolation. Then the data is processed in the following way. Whenever we get a hit which is outside of the range of the sensor or its x, y position is outside the testbed floor, we assign the value of "infinity", which is set to be at 120 cm. Joint visibility maps after each step are depicted in Figure 5.18. Actual obstacle boundaries are represented by yellow lines on each figure. Red stars represent positions of the robots after each step. The red lines mark the path of each vehicle up until its current location. Dark regions are invisible at current step and lighter regions are visible. Magenta circles represent shadow boundary obtained via ENO high order interpolation of the obtained range data. Black circles represent *horizon* points which will be approached in the next step. The complete visibility map is depicted in Figure 5.19. It is constructed by taking the union of visibility maps of all observers at all steps. From this map, one can estimate the quantity, size, and locations of the obstacles. However, the boundaries are not accurately represented due to low sensor accuracy and small number of samples. As was mentioned above, the results may be improved by correcting for the angle of incidence of the IR beam. Overall, the quality of the results is satisfactory taking into account hardware limitations.

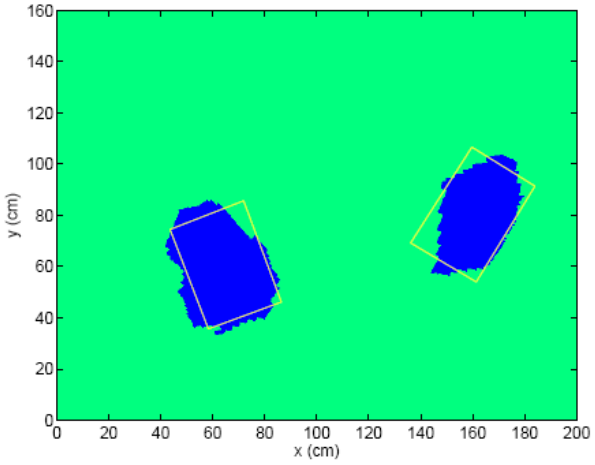

Figure 5.19: Map resulting from the environment exploration. The dark regions are invisible and light regions are visible; the yellow boxes are the actual outlines of obstacles.

CHAPTER 6

Conclusion and Future Work

The aim of the proposed testbed is to address the space and cost constraints typical to many research groups. By using many off-the-shelf devices, open-source software and the C programming language, we were able to construct a fully operational testbed that only occupies a small $2.0 \times 1.5m$ space and costs $160 per vehicle. While the position information is achieved through overhead cameras and off-board image processing, we are able to provide sub-palm size vehicles proximity sensing capability by integrating a compact infrared range detector. With the testbed, we are able to implement and validate algorithms for multi-vehicle flocking, dynamic obstacle detection and avoidance, multi-agent routing, and environment visibility exploration.

Figure 6.1 shows the system evolution from our first to second generation testbed. The second generation testbed is still confined to an indoor environment due to its overhead camera based positioning system. To develop an autonomous vehicle system that can be deployed in geographically complex environment, a viable tracking solution is mostly needed.

Satellite Navigation System (SNS) can provide the ideal solution for outdoor wide-area tracking as a vehicle equipped with a GPS receiver can estimate its position with an error less than several meters. Galileo, the GPS equivalent that EU is developing, is proposed to have accuracy as good as 10 cm by including a ground station into the system. However, the vehicles would have to be designed

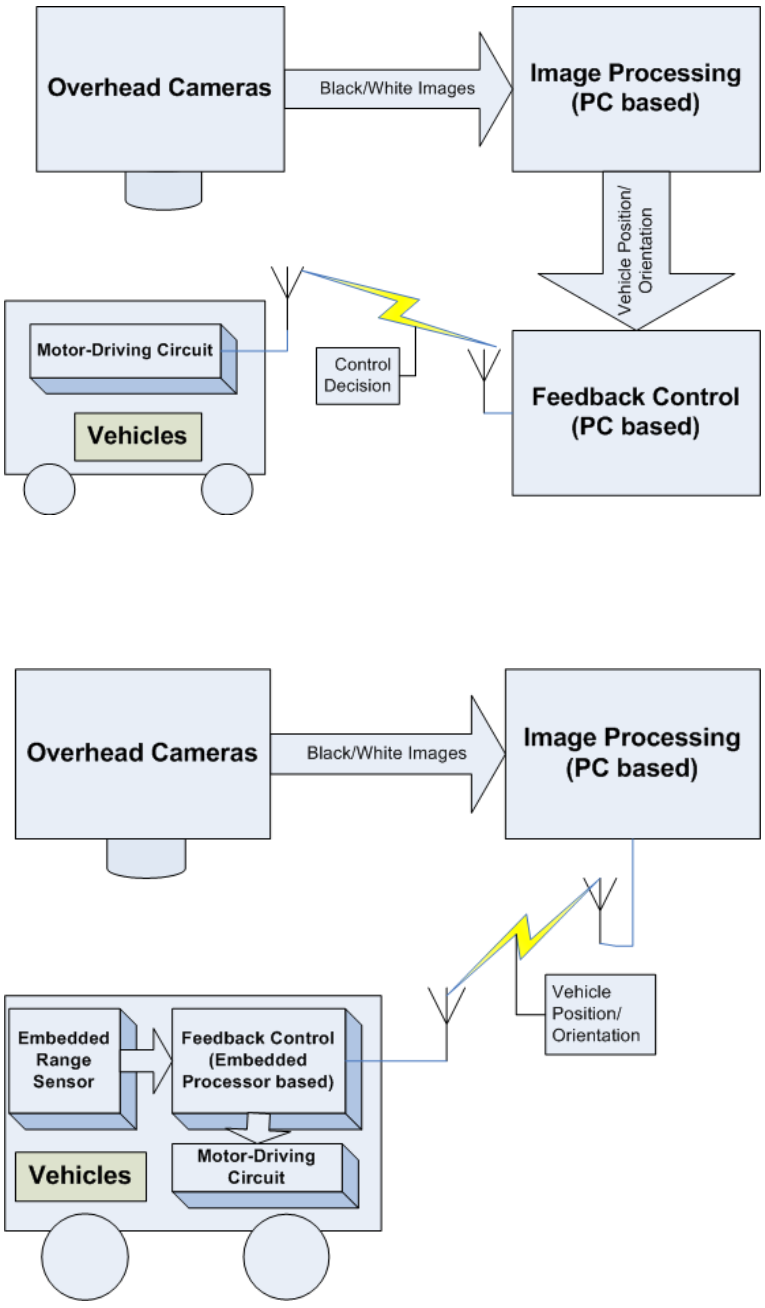

Figure 6.1: A system overview of the first (top) and second (bottom) generation testbed.

to be a few times larger to accommodate the integration of a GPS or Galileo receiver. In addition to the tracking system, vehicle size increase is necessary because larger wheels can be designed to withstand harsher ground conditions and last longer; geographical information is necessary for successful navigation as the ground is not meant to be flat; higher performance proximity sensors are needed as natural obstacles are inevitable and longer range communication is also crucial to establish effective inter-agent coordination. Figure 6.2 shows a system diagram for what we aim at achieving in the future.

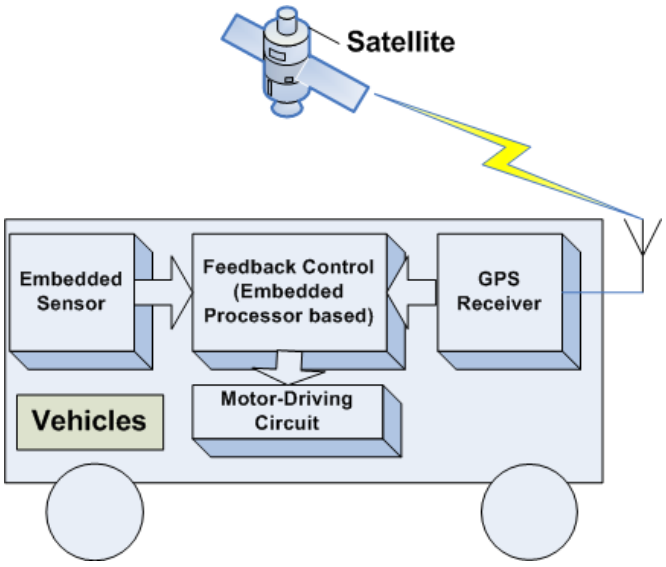

Figure 6.2: A system overview of a proposed autonomous vehicle system.

References

[1] C. H. Hsieh, Y. Chuang, Y. Huang, K. K. Leung, A. L. Bertozzi, and E. Frazzoli, *An Economical Micro-Car Testbed for Validation of Cooperative Control Strategies*, Proc. of the 2006 American Control Conference, Minneapolis, MN, June 14-16, pages 1446-1451.

[2] Y.-L. Chuang, Y. R. Huang, M. R. D'Orsogna, and A. L. Bertozzi, *Multi-Vehicle Flocking: Scalability of Cooperative Control Algorithms Using Pairwise Potentials*, submitted to 2007 IEEE International Conference on Robotics and Automation.

[3] K. K. Leung, C. H. Hsieh, Y. R. Huang, A. Joshi, V. Voroninski, and A. L. Bertozzi, *A Second Generation Micro-Vehicle Testbed for Cooperative Control and Sensing Strategies*, submitted to 2007 American Control Conference.

[4] Y. Landa, D. Galkowski, Y. R. Huang, A. Joshi, C. Lee, K. K. Leung, G. Malla, J. Treanor, V. Voroninski, A. L. Bertozzi and R. Tsai, *Robotic Path Planning and Visibility with Limited Sensor Data*, submitted to the 2007 American Control Conference.

[5] Carmeliza Navasca, Ani Asatryan, Vatche Attarian, Yuan R. Huang, Kevin K. Leung, Abhijeet Joshi, Vlad Voroninski, Meghdi Aboulian, and Krsytle McBride, *Implementations of control laws for motion camouflage*, submitted to the 2007 American Control Conference.

[6] T. Chung, L. Cremean, W.B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. van Gogh, and S. Waydo, *A platform for cooperative and coordinated control of multiple vehicles: The Caltech Multi-Vehicle Wireless Testbed*, Proc. of the 3rd Conference on Cooperative Control and Optimization, Dec. 2002.

[7] Z. Jin and S. Waydo and E. B. Wildanger and M. Lammers, H. Scholze and P. Foley and D. Held and R. M. Murray, *MVWT-II: The Second Generation Caltech Multi-vehicle Wireless Testbed*, Proc. of the 2004 American Control Conference, pp. 5321-5326, 2004.

[8] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, *Coordination and control experiments on a multi-vehicle testbed*, Proc. of the 2004 American Control Conference, pp. 5315- 5320 , 2004.

[9] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, J.P. Ostrowski, *Cooperative localization and control for multi-robot manipulation*, Proc. of the 2001 Intelligent Robots and Systems, pp. 631-636, 2001.

[10] A. Stubbs and G. E. Dullerud, *Networked control of distributed systems: a testbed*, Proc. 2001 ASME International Mechanical Engineering Congress & Exposition, New York, New York, 2001.

[11] V. Vladimerouy , A. Stubbs, J. Rubel, A. Fulford, J. Strick, and G. Dullerud *A Hovercraft Testbed for Decentralized and Cooperative Control*, Proc. of the 2004 American Control Conference.

[12] R. DAndrea, M. Babish, *The RoboFlag Testbed*, Proc. of the 2003 American Control Conference, pp. 656-660, 2003.

[13] R. DAndrea, *Robot soccer: A platform for systems engineering*, Computers in Education Journal, 10(1):5761, 2000.

[14] T. W. Mclain and R. W. Beard, *Unmanned Air Vehicle Testbed for Cooperative Control Experiments*, Proc. of the 2004 American Control Conference, pp. 5327-5331, 2003.

[15] The Minnow Project at Carnegie Mellon University, http://www.cs.cmu.edu/∼coral/minnow/.

[16] Robotic Embedded Systems Laboratory, Univ. of Southern California http://www-robotics.usc.edu/resl/

[17] MIT Computer Science and Artificial Itelligence Laboratory, http://people.csail.mit.edu/jamesm/

[18] L.E. Dubins. *On Curves of Minimal Length with a constraint on average curvature and with prescribed initial and terminal positions and tangents.*American Journal of Mathematics 79, 497-516, 1957.

[19] http://www.intel.com/technology/computing/opencv/index.htm

[20] http://msdn.microsoft.com/directx

[21] J. Borenstein, et al., *Where am I? Sensors and Methods for Mobile Robot Positioning*, http://www-personal.umich.edu/ johannb/shared/pos96rep.pdf

[22] http://www.gumstix.com/

[23] http://www.rabbitsemiconductor.com/products/rcm3400/

[24] http://www.junun.org

[25] http://www.radiotronix.com

[26] http://www.maxstream.net

[27] http://www.microrccars.com/

[28] M. R. D'Orsogna, Y. L. Chuang, A. L. Bertozzi and L. Chayes, "Self-propelled particles with soft-core interactions: patterns, stability, and collapse", *Phys. Rev. Lett.*, vol. 96, 2006.

[29] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials, and coordinated control of groups", in *Proc. Conf. Decision Contr.*, Orlando, FL, pp. 2968-2973, 2001.

[30] E. W. Justh and P. S. Krishnaprasad, "Equilibria and steering laws for planar formations", *Systems and Control Letters*, vol. 52, pp. 25-48, 2004.

[31] E. W. Justh and P. S. Krishnaprasad, "Steering laws for motion camouflage", Preprint.

[32] D. S. Morgan and I. B. Schwartz, "Dynamic coordinated control laws in multiple agent models," *Physics Letters A*, vol. 340, pp. 121-131, 2005.

[33] F. Zhang, A. O'Connor, D. Leubke, and P. S. Krishnaprasad, "Experimental Study of Curvature-based Control Laws for Obstacle Avoidance", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[34] Carmeliza Navasca, Ani Asatryan, Vatche Attarian, Yuan R. Huang, Kevin K. Leung, Abhijeet Joshi, Vlad Voroninski, Meghdi Aboulian, and Krsytle McBride, "Implementations of control laws for motion camouflage," submitted to the 2007 American Control Conference.

[35] S. M. LaValle, "Planning Algorithms", Cambridge University Press, 2006.

[36] A.G. Tartakovsky, B.L. Rozovskii, R. Blazek, H. Kim, "Detection of intrusions in information systems by sequential change-point methods," *Statistical Methodology*, vol. 3, Issue 3, pp. 252-340, 2006.

[37] M. Basseville and I.V. Nikiforov, Detection of Abrupt Changes: Theory and Applications. Prentice Hall, Englewood Cliffs, 1993.

[38] J.J. Enright and E. Frazzoli and K. Savla and F. Bullo. *On Multiple UAV Routing with Stochastic Targets: Performance Bounds and Algorithms.* Proc. of the AIAA Conf. on Guidance, Navigation, and Control,August 2005.

[39] A. Balluchi, P. Soues and A. Bicchi, *Hybrid Feedback Control for Path Tracking by a Bounded-Curvature Vehicle*, Proc. 4th International Workshop on Hybrid Systems: Computation and Control. pp. 133-146, 2001.

[40] Y. Landa, R. Tsai, and L.-T. Cheng, "Visibility of point clouds and mapping of unknown environments", *Advanced Concepts for Intelligent Vision Systems, ACIVS 2006*, Sept 18-21, 2006, University of Antwerep, Belgium (preprint available as UCLA CAM report 06-16).

[41] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy, "Uniformly high order accurate essentially nonoscillatory schemes, III", *Journal of Computational Physics*, 71, (1987), 231-303.