Chapter 1

A PLATFORM FOR COOPERATIVE AND COORDINATED CONTROL OF MULTIPLE VEHICLES

The Caltech Multi-Vehicle Wireless Testbed

Timothy Chung, Lars Cremean[♮], William B. Dunbar, Zhipu Jin, Eric Klavins, David Moore, Abhishek Tiwari, Dave van Gogh, Stephen Waydo
California Institute of Technology
Engineering and Applied Sciences
[♮]Corresponding Author
lars@caltech.edu

Abstract The Caltech Multi-Vehicle Wireless Testbed (MVWT) is an experimental platform for investigating the increasingly important intersecting frontiers of reliable distributed computation, communication and control. The testbed consists of eight autonomous vehicles equipped with onboard sensing, communication and computation. The vehicles are underactuated and exhibit nonlinear second-order dynamics, key properties that capture the essence of similar real-world applications at the forefront of cooperative control.

The relative simplicity of the testbed facilitates the investigation and application of novel ideas in reliable computing, real-time optimal control, stability of interconnected systems, control of and over networks, and formation flight. In this paper, we describe in detail the MVWT and its components so that readers may envision how it can be used to provide proof-of-concept for new techniques in multi-vehicle control.

Keywords: Multiple vehicles, coordinated, distributed, cooperative, testbed.

1. INTRODUCTION

We have developed a versatile experimental testbed (called the Multi-Vehicle Wireless Testbed, or MVWT) for the implementation of control algorithms for single- and multi-vehicle control problems. It consists

of eight fan-driven vehicles that glide on low-friction omni-directional casters. Each vehicle is equipped with an onboard computer and local sensors, and can communicate over a local wireless network.

The goal of the MVWT is to embody the difficult problem of multi-vehicle control (as found, for example, in multiple UAV systems) in an easy to use, low-cost and low-risk system. Our testbed captures many aspects of real-world application environments and can be used to validate theoretical advances in a variety of disciplines. Multi-vehicle control problems appear in a wide variety of applications in which theoretical advances can significantly improve our ability to perform complex tasks. Military applications include operation of unmanned aerial vehicles (UAVs) and unmanned combat aerial vehicles (UCAVs) in a variety of situations. Civilian examples include automation of air traffic control systems, automated highway systems, planetary exploration, and search and rescue operations.

The ultimate success of these applications depends heavily on a greater understanding of the research areas related to multi-vehicle (in particular, decentralized) control. This is especially true in situations where agents in a decentralized control scheme are part of a communications network and the effects of delay, communication constraints and dynamic constraints must be considered.

The main purpose of our testbed is to provide a platform where new theoretical ideas can be implemented and their practicability can be evaluated in light of the reality of a physical experiment. We briefly mention several areas that users of the MVWT are either exploring or plan to explore.

A natural task for a multi-vehicle system is formation flight wherein vehicles must maintain their positions relative to some global "shape". In [9, 22] formations and formation changes are specified by abstract *graphs* that define the neighbors of each vehicle and the desired distances and angles between them. In particular, the relationship between the stability of a formation and the structure of the associated graph is explored. Applying these results in a system with non-trivial dynamics such as the MVWT would represent a significant advance.

Optimal control techniques such as trajectory generation [20] and receding horizon control [12, 7, 14] are obvious choices for the control of nonlinear vehicles such as those in the MVWT. However, it is not at all obvious how to "distribute" optimal control problems [15, 21] over multiple vehicles so that each is solving a sub-problem in a truly decentralized fashion.

Low bandwidth communication introduces many difficulties into controlling the MVWT and systems like it. For instance, to achieve a

consensus (e.g regarding what formation to assume) in a distributed asynchronous network is theoretically impossible [11]. To address such difficulties, we are examining novel communication strategies [10, 23] and investigating their interactions with multi-vehicle control algorithms.

Finally, writing reliable control programs (to implement the above algorithms) and communication protocols is a fundamentally difficult problem even without the complicated dynamical scenarios that are commonplace in multi-vehicle control. We are exploring formal software methods and ideas including novel languages [17] and compiler techniques [13], and we plan to use these tools to implement multi-vehicle algorithms on the MVWT.

Many of the research areas described above carry novel results applicable to flight control experiments, but which have been verified only in simulation. In many cases a combination of high cost, prohibitive risk, and labor-intensive implementation inhibits the process of transferring novel control ideas to aircraft flight experiments.

To address this, our vehicles are designed to possess dynamics that capture the essence of real-world applications and thus *require* real-time and perhaps reconfigurable control. Many academic experiments involving coordinated or cooperative planning use vehicles for which real-time control is not an inherent requirement (e.g., wheeled kinematic vehicles). Our vehicles are described by equations that are second order dynamic, nonlinear, underactuated, input constrained, and input sign constrained. As in the case of real-world high performance systems such as military aircraft, these features make quick response time to new or existing constraints critical for mission success.

While the ground based flight control testbed we describe here captures many of the dynamical features of aircraft, it also avoids the danger of expensive failures or the overhead of launching, landing and maintaining temperamental aircraft. Our testbed is low cost, low risk and is designed specifically for ease of use and versatility in controller design and implementation.

We proceed here to give an overview of the architecture of the MVWT to show how we decided to address the goals defined above. We also describe the educational uses of the MVWT, and provide a brief discussion of related experimental testbeds. The later sections of the paper are devoted to the description of the components of the MVWT.

Overview. An overhead view of MVWT arena is indicated by Figure 1.1. The MVWT vehicles operate on a smooth floor of approximate dimension 6.7 m $\times$ 7.3 m. Each vehicle in the MVWT is equipped with an onboard computer, local sensors and an 802.11b wireless Ethernet

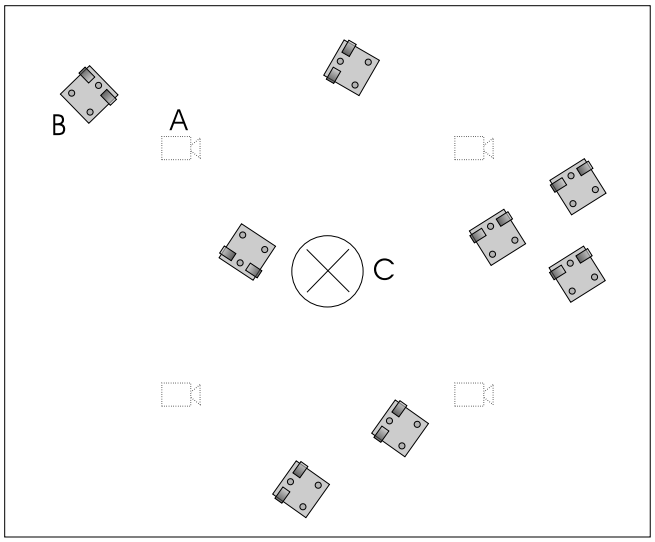


Figure 1.1. A diagram of the Caltech MVWT. An off-field vision computer processes images from overhead cameras (A) and provides state information to the vehicles (B). An obstacle (C), due to an unrelated experiment in the same space must be avoided.

card for communication over our laboratory's local area network. Video cameras mounted on the ceiling take pictures of the field at 60Hz and are connected to an off-field vision processing system. The vision system identifies each vehicle, its position, and its orientation by the unique symbol that appears on the vehicle's "hat" (Figure 1.9). It broadcasts this information over the local area network and the vehicles receive it via their wireless cards. Each vehicle can communicate through the same wireless network to the other vehicles and to auxiliary computers off-field that may serve as command systems, real-time monitors, data loggers or human user interfaces.

The block diagram of Figure 1.2 depicts the MVWT architecture. A picture of an MVWT vehicle appears in the lower right. Each vehicle's input is composed of communicated information (e.g. from the vision system, other vehicles or a command center), and sensed information from onboard sensors (e.g. gyroscopes and ultrasonic range finders). The vehicle's output includes messages to other vehicles (or auxiliary computers) and actuation commands to its two ducted fans, which provide thrust and the vehicle's means of mobility. A configurable user interface is defined in a program called "Master Control" which handles high level control commands, including those for data logging. Information flows both ways between Master Control and the vehicles, which

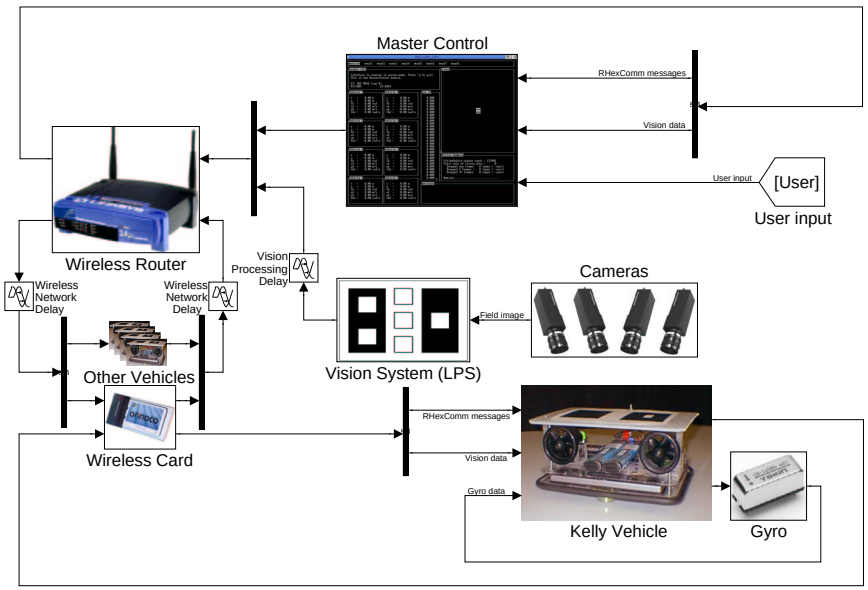


Figure 1.2. The structure of the MVWT: Vehicles receive input from sensors and from the wireless network and output thrust commands to their fans and messages to the network. A vision system senses the positions and orientations of the vehicles.

allows for easier testing of control designs and real-time monitoring of results.

The computing platform onboard each vehicle is a laptop which runs the QNX real-time operating system and a C++-based software suite called RHexLib (described in Section 6). This software architecture allows us a rapid prototyping environment that places few restrictions on the type and design of controllers that we are able to implement. Essentially, we can implement any controller that can be written in C/C++ and is amenable to a statically-scheduled and fixed-rate environment. The additional details of our software environment (see Section 6) and our communication capabilities (see Section 7) make for a highly configurable platform for experiments in many cutting edge areas of control theory. Our testbed can be used to investigate decentralized control algorithms, high-level coordination algorithms, real-time communication protocols, novel wireless network architectures, distributed sensing and actuation strategies, and much more.

Educational Uses. In the Spring of 2002 the MVWT was used as the main experimental testbed for the Caltech course "Applications of Control Technology" (CDS 111). The purpose of the course is to provide

students with the opportunity to put into practice the theoretical techniques learned in previous courses. In CDS 111, the students become familiar with the components of the testbed and how to operate the vehicles. They then run simple programs to become familiar with controller design and deployment, and finally are required to design their own controllers to accomplish specific tasks on the testbed. The course drives continuous improvement in software infrastructure and user-friendliness of the MVWT for future users. We expect the testbed to be used by CDS 111 for at least the next several years.

Related Work. There are several other testbeds related to the MVWT. For example, the recent popularity of the RoboCup [6] robotic soccer tournament has seen numerous wheeled robot systems built. The main difference between such systems and the MVWT is that they can often be treated as if they are first order, thereby significantly simplifying their control. Several flight based testbeds are also in development that use fixed-wing UAVs [8], unmanned helicopters [16], or hovercraft [25]. The MVWT differs from the first two in that vehicle failures are not as dramatic and costly. The latter testbed is quite similar to the MVWT, although the vehicles float on a forced air table rather than resting on casters, limiting the range of the vehicles to the size of the (necessarily small) table.

Paper Organization. In the rest of this paper we describe in detail the architecture of the MVWT including the physical and computational specifications of the vehicles and the laboratory infrastructure that supports them. In Section 2 we describe the details of the system hardware. We discuss the vision system in Section 3 and onboard sensing capabilities in Section 4. In Section 5 we describe the various electronic components. Our software environment is described in Section 6 and our communication systems in Section 7. We discuss how the testbed has been used to explore new control algorithms in Section 8. In Section 9 we attempt to predict the future of the system in terms of continued infrastructure development and planned interesting uses of the MVWT.

2. VEHICLE HARDWARE

One of the MVWT vehicles is shown in Figure 1.3. It is supported by three low-friction omni-directional casters and is driven by two ducted fans (fixed to the vehicle), each producing up to 5.0 N of thrust. The vehicle is essentially a "laptop sandwich", with a laptop computer secured between two pieces of machined acrylic. Attached to the top acrylic piece are the fan assemblies, battery packs, identifying "hats", interface

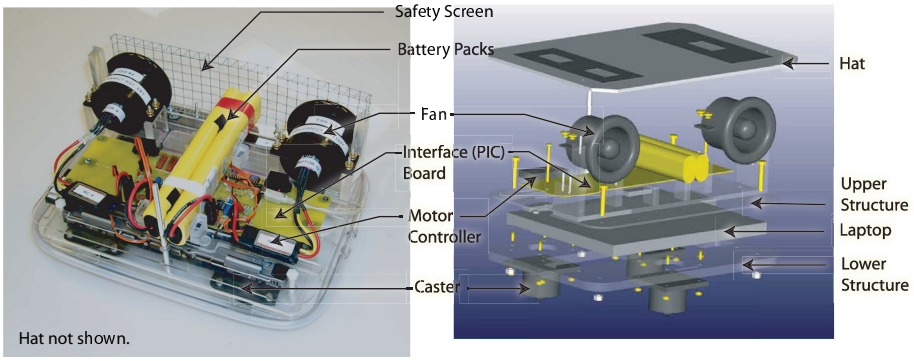


Figure 1.3. An MVWT vehicle (left) and exploded view (right).

board, and local sensors. The casters are attached (ball-down) to the bottom piece of acrylic. All of these components are described in detail below.

See Figure 1.4 for critical dimensions and the MVWT website [2] for vehicle specifications, including a bill of materials, detail drawings of the acrylic structure, and laptop specifications.

2.1. COMPONENTS

Chassis. The chassis is machined from acrylic, chosen for its low cost and machinability. Acrylic also allows us to easily bond additional components to the main chassis (e.g., to keep the bumpers from sliding off or to hold the batteries in place). The structure was designed to allow access to most of the laptop ports, including USB, power, video, and PS/2 ports.

A loop of clear vinyl tubing is stretched around the bottom plate of the chassis to protect the vehicle and its contents during bumps with other vehicles or with the surrounding walls of the testbed floor. Finally, a corrugated plastic "hat", placed on the top of the vehicle, is used by the vision system to determine vehicle identity, position and orientation. See Section 3 for more details.

Casters. The casters (also known as *ball transfer units*) are furnished by Alwayse Engineering Limited. They are nearly ideal for our purposes since they have relatively low friction and are self-cleaning. According to the Alwayse website [1], the load ball (50.8mm diameter) "rotates on a bed of small balls supported on a hardened steel, precision

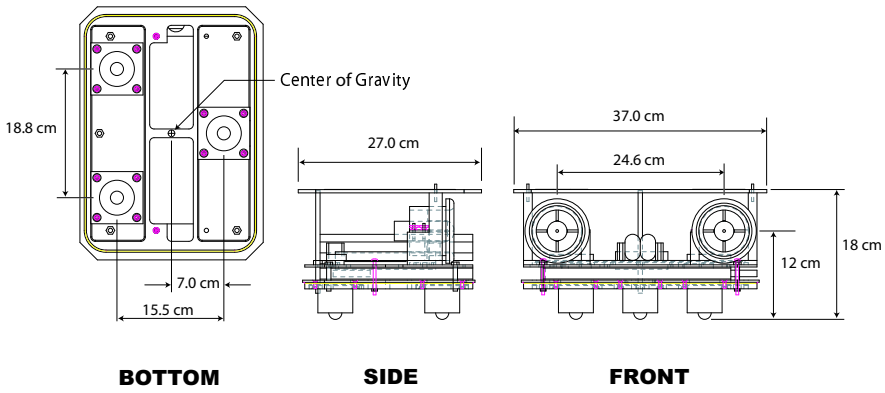


Figure 1.4. Critical dimensions of an MVWT vehicle.

machined table." There are two seals that keep dirt and dust away from the bearing surfaces - one to remove large particles and another to scrape the ball clean. This self cleaning feature has proved to be indispensable, as other omni-directional casters we tried quickly "gummed up" with dust and debris, drastically changing their friction characteristics.

Fan Assembly. The fan assembly consists of the ducted fan housing and rotor (WeMoTec Minifan 480), motor (Hacker B40-14S), and motor controller (Jeti MASTER 40-3P Opto). Assembling these components is fairly straightforward; the most critical step is balancing the fan rotor. An unbalanced rotor not only degrades thrust performance, but also causes the rotor to vibrate and hit the side of the duct during operation, causing excessive noise and damage to the rotor and duct. We found the Du-Bro Prop Balancer to be an effective tool to balance the rotor.

Fan Characterization. For a control system to be effective, the actual control input (in our case, the actual forces produced by each fan) should be equal to the control input commanded by the control system. In the case of the MVWT vehicle, the control system software onboard the laptop commands forces to each fan. These forces are converted to integer values and sent via USB to the microcontroller. Therefore it is necessary to determine what fan force is produced for a given integer input to the microcontroller. See Section 8 for details of controllers used on the vehicle, Section 5 for details of the vehicle's electronics

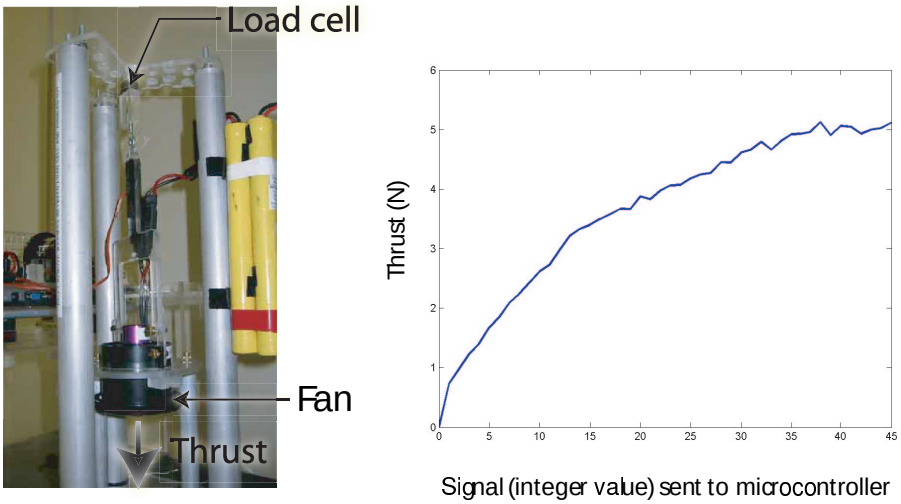


Figure 1.5. Fan assembly hanging in the thrust map test fixture (left) and a typical thrust map (right). Note that the entire weight of the fan assembly is supported by the load cell.

(microcontroller and interface board), and Section 6 for details of the software used on the vehicle.

For each fan assembly, thrust was measured over the full range of integer input signals sent to the microcontroller, producing a map between the microcontroller's input signal and actual fan thrust. The fixture used to create these maps is shown in Figure 1.5, along with a typical thrust map produced by the fixture. To measure thrust, the fan is hung in the fixture as shown, with the entire weight of the fan assembly supported by a load cell (which outputs a voltage proportional to axial load).

Battery Pack. Each fan assembly is powered by ten sub-C size nickel metal hydride (NiMH) 3000 mAh cells, manufactured by both Sanyo and Panasonic and packaged by Batteries America of Middleton, WI. The battery packs are charged with a Robbe Power Peak Infinity II charger, which automatically detects when the batteries are fully charged, preventing damage to the battery cells. During normal use, a battery pack lasts about 20 minutes and takes approximately 1.5 hours to charge. Current draw is approximately 20 A at full thrust.

Computing Platform. Each vehicle is equipped with either a 700 MHz Dell Inspiron 2100 laptop or Dell Latitude L400 laptop (both

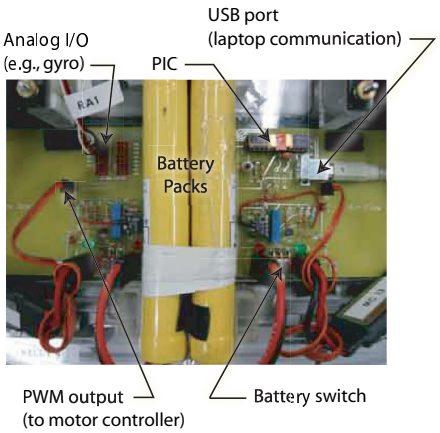


Figure 1.6. Detail view of the interface (PIC) board that relays information between the onboard computer and the fans and sensors.

of which have the same form factor). The screens are removed to save space and reduce weight and power.

Interface Board. An electronic interface board relays fan force output from the onboard computer via the USB bus, converts it into a pulse-width modulated (PWM) signal, which is the format expected by the motor controllers. The interface board is also used to relay sensor information to the onboard computer. See Figure 1.6 and Section 5 for more details.

2.2. PARAMETER IDENTIFICATION

The simplified equations of motion (assuming perfect sensing and actuation, no delays, no disturbances, and linear friction) for the vehicle are listed in Equations 1 - 3. These are derived by observation from the simple schematic of the vehicle shown in Figure 1.7 (A more detailed discussion can be found in Section 8).

$$m\ddot{x} = -\mu\dot{x} + (F_R + F_L)\cos\theta \quad (1)$$

$$m\ddot{y} = -\mu\dot{y} + (F_R + F_L)\sin\theta \quad (2)$$

$$J\ddot{\theta} = -\psi\dot{\theta} + (F_R - F_L)r_f \quad (3)$$

These equations include four physical parameters: the mass m, the mass moment of inertia J, and the linear and rotational viscous friction

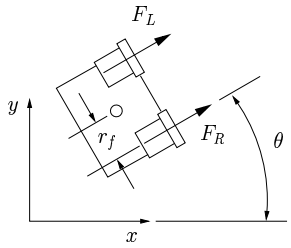


Figure 1.7. Schematic of MVWT vehicle. The coordinate frame is inertial and the forces F_L and F_R are applied at the fan axes.

coefficients μ and ψ. The geometric parameter $r_f (= 0.123$ m) is the distance between the center of mass of the vehicle and each fan axis.

Mass. Mass was measured with a digital scale. Each vehicle has a mass of 5.05 ± 0.05 kg.

Mass Moment of Inertia. To measure J, the vehicle was hung from a rod as shown in Figure 1.8. It was then displaced from its equilibrium position and allowed to swing freely. J was calculated from the small-angle approximated equation $J = mL^2(\frac{gT^2}{4\pi^2 L} - 1)$, where L is the vertical distance from the axis of rotation to the vehicle's center of gravity and T is the period of oscillation. The value of J was found to be $0.050 \pm 0.005\ kg\ m^2$.

Linear Friction. A test fixture for determining μ is shown in Figure 1.8. The vehicle was pulled back from its equilibrium position and allowed to oscillate. The vision system recorded the position of the vehicle as a function of time, and the logarithmic decrement technique was used to determine the damping coefficient [19]. μ was found to vary between 3.3 and 5.5 $\frac{kg}{s}$ (different for each vehicle).

Rotational Friction. The rotational viscous friction coefficient ψ was calculated by measuring the steady state angular velocity of a vehicle with one fan operating at constant thrust and the other turned off. The equation of motion in this case is $J\ddot{\theta} = Fr_f - \psi\dot{\theta}$ (assuming the vehicle is rotating about its center of mass – this was observed in practice). At steady state ($\ddot{\theta} = 0$), $\dot{\theta}_{ss} = \frac{Fr_f}{\psi}$, or $\psi = \frac{Fr_f}{\dot{\theta}_{ss}}$. All parameters in this equation are known (F from the thrust map, r_f by direct measurement, and $\dot{\theta}_{ss}$ via vision system), making the calculation of ψ trivial. This

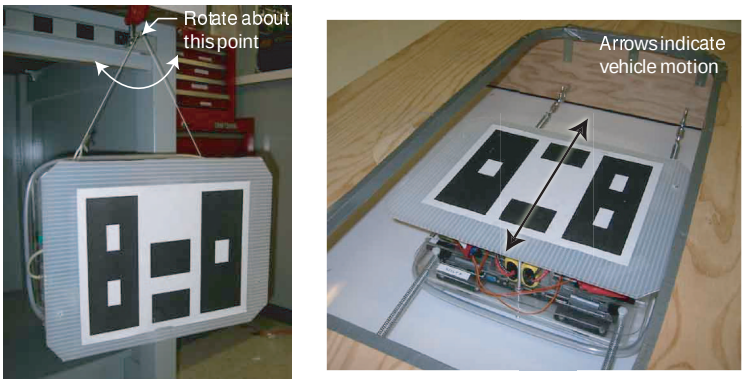


Figure 1.8. Test fixtures used to measure J, the mass moment of inertia (left), and μ, the linear viscous friction coefficient (right).

parameter was found to vary between 0.049 and 0.064$\frac{kg\ m^2}{s}$ (different for each vehicle) using this method.

3. LAB POSITIONING SYSTEM

Vehicle localization is accomplished using an overhead camera system known as the Lab Positioning System (LPS). This system consists of four Pulnix TM-6710 monochrome CCD cameras and three Matrox vision processing boards (one Genesis and two Genesis Plus). Each camera covers an area of approximately 4.88 x 4.26 m on the floor and produces 640 x 480 pixel images at 60 Hz. The Genesis vision board combines these four images into one and each Genesis Plus board processes alternate frames of data.

Vehicle localization and identification is accomplished using a pattern of black "blobs" on white "hats" atop each vehicle. To facilitate processing, the floor of the MVWT lab is entirely white so blob identification is unambiguous. The vision processing boards perform a standard blob analysis, identify all blobs in the image that are in an appropriate size range, and send that information to the vision processing algorithm. The algorithm takes this information and identifies the patterns of blobs corresponding to the hats with their associated vehicle identities, positions and orientations.

Figure 1.9 depicts an example hat pattern. It has two large blobs, one with one white patch, or "hole," and the other with two, that are used for localizing and determining the orientation of the corresponding vehicle.

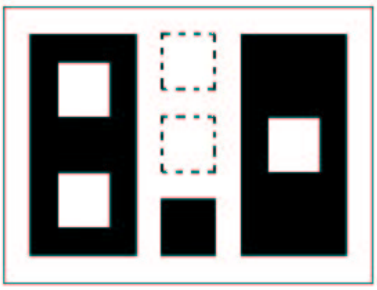


Figure 1.9. An example vehicle "hat" (number 1 shown). The large "blobs" are used to determine vehicle location and orientation and the smaller blobs determine the vehicle's identity.

Between these blobs are up to three smaller blobs (with no holes) that represent, in binary, the vehicle's integer identification number.

Localization is accurate to approximately 1 cm (when the vehicles are stationary). Noise when the vehicles are stationary is insignificant, with a standard deviation of 0.2-0.25 mm in position and 0.0035 rad in orientation. The vision system also provides rate information using an (unfiltered) "dirty derivative" which is significantly noisier than the position data, with a standard deviation of 1.0-1.5 cm/s in linear velocity and 0.2 rad/s in angular velocity for a stationary vehicle. All filtering and smoothing of this data is currently done on board the vehicles. Processing of one frame of vision data takes approximately 33 ms. This information is then broadcast to the vehicles over the local wireless network.

4. ONBOARD SENSING

Onboard sensing not only serves to augment the vision system in observing the vehicle motion and behavior, it also enables the vehicles, individually or collectively, to gather information about their surrounding environment. The onboard sensing capability of the MVWT facilitates research of interesting topics such as distributed sensing, sensor fusion, and decision making.

4.1. ULTRASONIC RANGEFINDER

In many multi-vehicle applications, whether they be formation control, mapping and tracking tasks, or obstacle avoidance, knowing the relative distance from one vehicle to another vehicle or object is vital. The most common types of proximity sensors include infrared, acous-

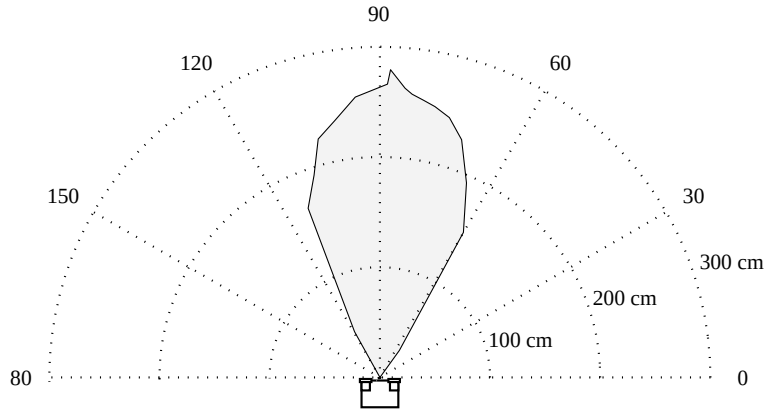


Figure 1.10. The beam pattern for the Devantech SRF04 ultrasonic rangefinder. The SRF04 has a range of approximately 3 cm to 3 m and a resolution of approximately 8 cm.

tic, and optical (laser) rangefinders. After weighing the requirements of range and accuracy against the constraints of cost and dimensions we chose to explore the use of an ultrasonic (i.e. acoustic) rangefinder.

Sensor description. The ultrasonic rangefinder we selected for the MVWT is the Devantech SRF04. It has a minimum range of approximately 3 cm and a maximum range of approximately 3 m. The sensing cone of the SRF04, or the angular range, sweeps through an approximately 50°angle, as illustrated in Figure 1.10. The resolution allows for detection of a 8 cm target at approximately 2.8 m.

Software Integration. The sonar measurements are made available to the vehicle's onboard computer via the electronics interface (Section 5). Each sonar sensor uses two bytes of the data packet sent via the USB to the *SonarModule* in RHexLib (described in Section 6). The *SonarModule* allows the control software to collect range measurements at a user-specified rate and output them using the logging capabilities built into the MVWT RHexLib software.

4.2. RATE GYROSCOPE

The vision system introduces significant delays (greater than 0.05 sec) into the vehicle's control system, which degrade the vehicle's performance. Use of an onboard piezoelectric rate gyroscope (gyro) can significantly reduce the angular rate delay, since minimal signal processing

and no communication is required to get the gyro output to the control system. In addition, the gyro measures $\dot{\theta}$ directly, which has advantages over estimating $\dot{\theta}$ based on our discrete vision measurements.

Sensor Description. The gyro we are using is the Tokin CG-16D, whose output is proportional to the angular rate of the sensor, a product of the Coriolis effect on an internal vibrating ceramic column printed with electrodes.

Gyro Calibration. The gyro was placed on a turntable and its output measured for a given angular rate. We found that the output of the gyro was linear in the range $\pm$ 840 degrees/s, which in fact exceeds Tokin's specified maximum detectable angular rate of $\pm$ 90 degrees/s.

Gyro Circuit. Before using the gyro signal in the vehicle's onboard control system, it must be converted from an analog signal to a digital one. The PIC, which does the A/D conversion, accepts input in the 0 - 5 Volt range and converts it to an 8-bit digital signal. While on the vehicle, however, the gyro outputs a signal only in the range $2.0 \pm 0.3\ V$, even during aggressive maneuvers. If this signal were sent directly to the PIC, the resulting digital output would have poor resolution. To improve this resolution, we built a circuit to amplify the gyro's output, using more 0 - 5 Volt range of the A/D input. In addition, the circuit includes a low-pass filter to reduce the effect of high-frequency noise.

Preliminary Results. See Figure 1.11 for a plot of the vehicle's angular rate as measured by both the gyro and vision system (to produce this data, the vehicle was rotated back and forth by hand). As discussed above, the gyro output is sent through an analog filter prior to use by the vehicle's onboard control system. As currently implemented, the angular rate from the vision system is calculated by performing a "dirty derivative" on the vision system's vehicle orientation (angle) output. To accurately compare the gyro and vision system angular rate data, the vision system's angular rate output was processed through a digital filter with characteristics similar to the gyro's analog filter. Figure 1.11 shows that with the same filtering, the gyro signal produces a cleaner signal and one with less delay than the vision system.

5. ELECTRONICS

In this section we describe the electronics onboard the vehicles. In Section 5.1 we describe the latest version of electronics, the *MVWT interface board*, which communicates with the onboard laptop though the

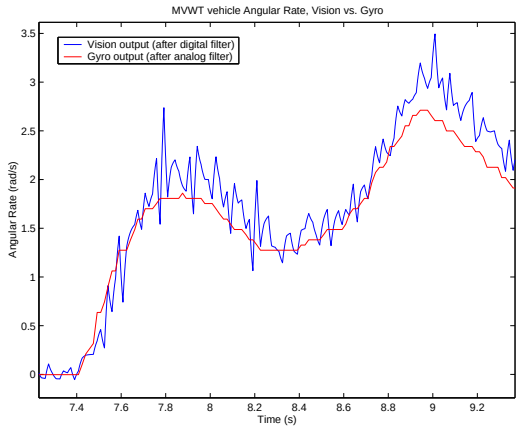


Figure 1.11. A plot showing the angular rate of an MVWT vehicle, from both the gyro and the vision system. The vision data has been post-processed by a digital filter with the same characteristics as the gyro's analog filter. Note the smoother signal from the gyro and the slight delay introduced by the vision system.

laptop's Universal Serial Bus (USB). A description of the Pulse Width Modulation (PWM) scheme and the command communication protocol is given. In Section 5.2 we describe the recently developed sensor integration scheme at the hardware level and the local feedback communication protocol. Section 5.3 summarizes the electronics with the help of a functional flow chart.

The motor speed control assembly consists of a microcontroller which receives integer values (corresponding to a commanded force) which are generated by the onboard laptop. The microcontroller converts these values to PWM signals with appropriate duty cycles, which it outputs to the two motor controllers, one each for the left and right fans. The motor controllers control the fan speeds proportional to the duty cycle of the input PWM signals.

5.1. THE MVWT INTERFACE BOARD

The MVWT interface board is the latest version of electronics installed on the vehicles. An 8-bit microcontroller Microchip PIC16C745 with USB 1.1 support is the core of the interface board. In addition to translating force commands from the onboard computer to PWM signals required by the two motor controllers, the board is also capable of collecting data from onboard sensors and sending it back to the computer.

The force command instruction format that the interface board expects from the onboard computer is composed of three bytes. The first byte is used as a "preamble" recognized by the microcontroller. The second and the third bytes are desired force values for the right and the left fans respectively.

The interface board provides I/O ports and USB socket for the microcontroller, and also protects the keyboard of the onboard computer. In addition, the interface board integrates a monitoring circuit for battery voltage and switching circuits for the batteries.

5.2. SENSOR INTEGRATION

The interface board is capable of collecting data from the onboard sensors and reporting them back to the laptop. In general, for an analog sensory signal, the signal first needs to be filtered and amplified. We can use any of the 5 available A/D channels available on the microcontroller to measure this signal. As examples, we discuss this interface mechanism as employed for two different sensors.

Gyroscope. Being an analog device, the gyroscope (gyro) interfaces to the microcontroller through an analog to digital (A/D) converter channel available on the chip. Whenever there is a new value available at the A/D conversion register, the microcontroller sends it to the laptop through the USB. For more details on the gyro, please refer to Section 4.

Ultrasonic Sensors. In every alternate timer 1 interrupt service routine, the microcontroller sends a firing pulse to the sonic sensor through the bi-directional I/O PORT A. In response the sonic sensor sends an ON_time pulse back to the external interrupt (INT) pin of PORT B. A high to low transition on the INT pin triggers the external interrupt and the width of the ON_time pulse is caught by Timer 1. For the details of the ultrasonic sensor, please refer to Section 4.

Local Feedback Communication Protocol. We refer to the link from the interface board to the laptop as the local feedback communication protocol. The microcontroller sends data packages of length 8 bytes back to the laptop through USB. The first byte is reserved for the system. The second byte is used for the gyro data. The third and forth bytes are used for the sonic sensor. The others are free for future use.

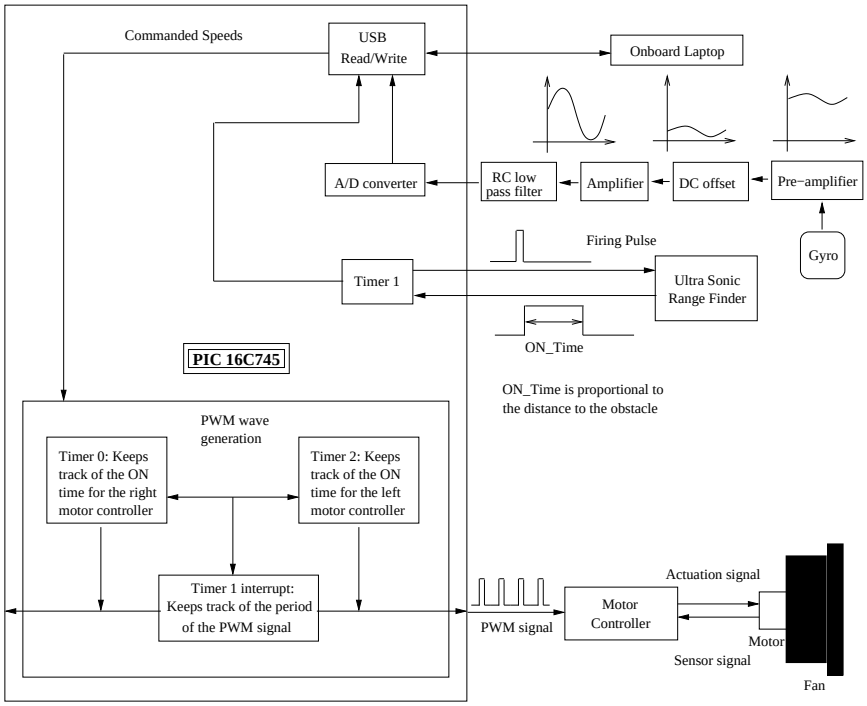


Figure 1.12. MVWT Electronics Design. The current version of the interface electronics use the Microchip PIC microcontroller (enclosed in rectangle) to interface to the laptop, two onboard sensors, and the motor controllers.

5.3. ELECTRONICS DESIGN

The primary purpose of the electronics design is to provide an interface between the laptop and the other electrical components, such as electrical fans, the gyroscope, and ultrasonic sensors. In Figure 1.12, the big block on the left side is the microcontroller. From top to bottom, there are four channels connecting to the right side. The first one is the data flow between the microcontroller and the laptop. It is a two-way USB channel. The second one is the gyro data flow. It is a one-way A/D converter channel from the gyro to the microcontroller. The third one is the ultrasonic sensing channel, and the microcontroller sends out firing pulses by PORT A and gets ON_time pulses back by PORT B. The fourth one is the one-way PWM channel which sends the PWM control signals to motor controllers.

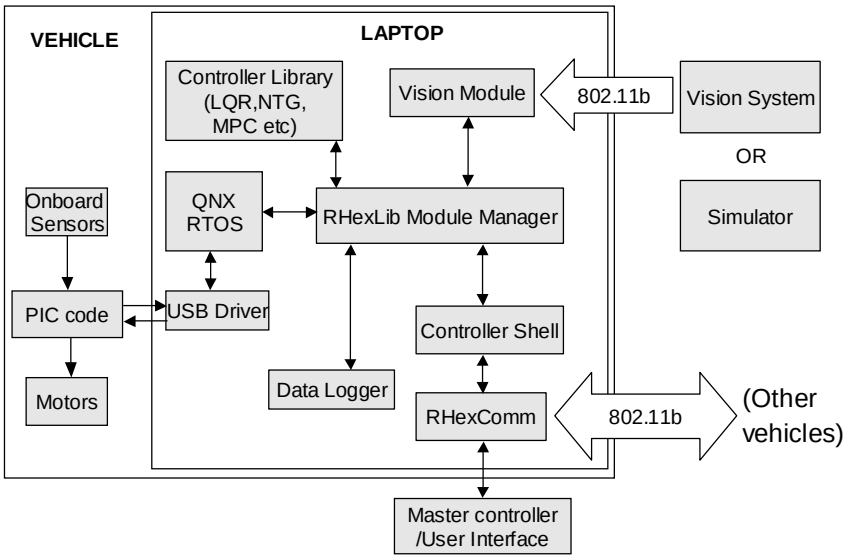


Figure 1.13. The structure of the MVWT software: Software on the PIC communicates between the laptop and the sensors and motors via the USB bus. A QNX USB driver is managed by the operating system to created a `/dev/usbpic` interface to the PIC. The RHexLib Module Manager manages various RHexLib modules for control, reveiving communication from the vision system, logging data, and communicating with other vehicles and the master controller.

6. SOFTWARE ENVIRONMENT

The software environment for the MVWT consists of the operating system (OS), the low level device drivers, inter-vehicle communications and a library and API that tie everything together. We discuss each of these components in turn. Figure 1.13 illustrates the structure of the software on an MVWT vehicle.

Operating System. The operating system running on the vehicles and on the command and control workstation is QNX, a freely available real time OS [26]. QNX is a POSIX compliant UNIX-like OS that provides for real-time scheduling and interprocess communication. Most of the software we have written for the MVWT (except the device drivers) is independent of the details of QNX and will run equally well on other versions of UNIX. We chose QNX because it is highly configurable for embedded systems (the kernel can, for example, be configured to be quite lean) and because several related software enabled control projects [28, 8, 16] also use QNX, allowing us to easily share software.

Hardware Interface. As we described in Section 5, communication between the microcontroller on the interface board and the processor is via a USB bus. We have written a QNX device driver that manages this interface. The details of the driver are beyond the scope of this paper so we only describe its operation here. Starting the `usb-pic` driver places a UNIX-style character device in the filesystem at `/dev/usbpic`. Integer triples of the form $\langle \texttt{0x05}, \texttt{L}, \texttt{R} \rangle$ can be written to `/dev/usbpic` and are sent to the microcontroller where they are converted into PWM signals and sent to the motors. There is an empirically measured map $f : \{0, 1, ..., 255\} \mapsto \mathbb{R}^{\geq 0}$ that maps commands c into the steady state force $f(c)$ generated by the fans (see Figure 1.5). Thus to generate 2.5 N on the left fan and 0.1 N on the right fan, the user writes $\langle 05, f^{-1}(2.5), f^{-1}(0.1) \rangle$ to `/dev/usbpic`. Software may also read integers from `/dev/usbpic` that are generated by sensors (such as the gyroscope described in Section 4).

RHexLib. Control software on the MVWT vehicles and command and control station are written in C++ using the *RHexLib* library and API [3]. RHexLib is freely available and was originally written for the RHex robot [24, 18], although it is quite general and can be easily adapted to new applications. RHexLib programs consist of a collection of *Modules* and a static schedule set up by the user for executing the modules in an interleaved fashion. The schedule is executed by the RHexLib *ModuleManager*. RHexLib provides a template for modules in the form of a C++ abstract base class with the following abstract methods that must be defined by user modules that inherit from it: `init()`, `activate()`, `update()`, `deactivate()`, and `uninit()`. The `init()` and `uninit()` methods are called when the modules are added to the ModuleManager and serve as the constructor and destructor for the module (for memory and resource (de)allocation). The `activate()` and `deactivate()` methods are called when the Module Manager activates or deactivates the module. When a module is active, the ModuleManager calls the module's update method at some frequency that is an attribute of the module. For example, the most basic single vehicle MVWT program consists of three modules: `VisionModule`, `ControlModule` and `DeviceWriter`. The update methods of these modules read vehicle position data from the vision system, compute a state-based control law and write the computed forces to `/dev/usbpic` respectively.

Simulator. We also have a virtual device that can replace the actual USB device driver. It sends force commands written to `/dev/usbpic` to a numerical simulator. The simulator integrates the forces with respect to

the model and broadcasts vehicle position data just as the vision system does. Thus, the same RHexLib code that runs on the actual vehicles may also be tested in simulation without any code being rewritten.

Trajectory Generation. An auxiliary component of our software environment is the Nonlinear Trajectory Generation (NTG) package [20]. NTG computes optimal trajectories given initial, trajectory, input and final constraints and can be used to generate feasible trajectories for the MVWT vehicles to follow. Because the computation of such trajectories takes time (0.5 to 1.5 ms), NTG is run as a separate thread so that the main inner loop control thread can continue to control the vehicle.

7. COMMUNICATIONS

Communication between the vehicles and with command and control computers is made possible by wireless Ethernet using an 802.11b access point. The computer onboard each vehicle is equipped with an 802.11b PC card, the standard QNX TCP/IP stack, and its own unique IP address. The wireless network serves three purposes. First, the setup provides for ease of development by allowing us to remotely modify, compile, and start the control algorithms on the vehicles. Second, cooperative control algorithms use the wireless network to communicate with other vehicles and the master controller. Lastly, the real-time position and orientation information for each vehicle is broadcast to the vehicles and master controller using the UDP protocol.

The implementation of software algorithms for cooperative control is facilitated by a communications software suite of our own design called *Libcomm*. Libcomm is a library that provides communication primitives for real-time control applications and abstracts away the underlying network protocol. It consists of several parts: A standard C library providing the communications API; a C++ module for RHexLib that wraps the API and makes it useful when designing control applications with RHexLib; and a MATLAB implementation of the API for rapid prototyping of control algorithms at a high-level. The Libcomm API is not tied to any particular network implementation and allows additional network backends to be added to the library with ease. The backend currently in use by the MVWT project uses the UDP protocol for all communications traffic. UDP is an adequate choice for unreliable wireless networks because of its lightweight implementation, which utilizes datagrams rather than more expensive streams.

With Libcomm, message passing between hosts consists of sending a list of parameters with their corresponding values to a given host or broadcast to all. In addition, each message has a type which identifies

it to the appropriate RHexLib module on the receiving end. Naming of hosts is achieved in a distributed fashion by the automatic broadcast of a string identifier when Libcomm is initialized. This allows each vehicle to keep track of all other vehicles within communications range without requiring knowledge of the naming method on the underlying network protocol.

Investigation of coordinated control in ad-hoc networks is a goal of the MVWT project. Ethernet over 802.11b provides a *managed* rather than *ad-hoc* wireless network. In order to study the performance of ad-hoc networks further, a Bluetooth backend for Libcomm is currently in development. See Section 9 for more details.

8. MODELING AND CONTROL

This section details our initial approaches to controller design. We begin with a discussion of a general system model and its features. We then discuss the effects and implications of closed-loop delay on the system, and describe several controller designs that have been demonstrated on the testbed and/or in simulation.

8.1. SYSTEM MODELING

A significantly simplified model of a single MVWT vehicle is given in Figure 1.7 of Section 2. Although useful for simple, moderately aggressive control, the model neglects factors such as closed-loop delay, vision noise and disturbances, vehicle disturbances and actuator dynamics. A more general version of the system equations that includes these considerations is given in equations (4) through (7).

$$\dot{\xi} = \mathbf{g_a}(\xi, \mathbf{q}(\mathbf{u_{des}})) \tag{4}$$

$$\mathbf{u} = \mathbf{h_a}(\xi) \tag{5}$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathbf{d}(\mathbf{x}) \tag{6}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}(t - \tau(t))) + \mathbf{n}(\mathbf{x}) \tag{7}$$

Equations (4) and (5) represent the actuator dynamics. The forces computed by the controller are denoted $\mathbf{u_{des}} \in \mathbb{U} \subset \mathbb{R}^2$, $\mathbf{q}$ represents the fact that these forces are quantized for input into our digital electronics, and $\mathbf{u} \in \mathbb{U}$ represents the actual forces that the fans apply to the vehicle. The input space $\mathbb{U}$ is defined by the bounds on our actuation variables. The vector $\xi \in \mathbb{R}^m$ represents a state-vector of the actuator system (electronics, motor controller, motor and fans), $\mathbf{g_a}$ and $\mathbf{h_a}$ represent the dynamics of the actuators. For example, we could model our actuator system as a first-order lag where $\mathbf{u} = \xi$, in which case $\xi \in \mathbb{R}^2$.

More sophisticated models could augment the state with battery voltage and $\mathbf{g_a}$ and $\mathbf{h_a}$ could include the evolution and effect of battery voltage on the actuators.

Equation 6 represents the vehicle dynamics. The variable $\mathbf{x}$ is the vehicle state vector $[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]^T \in \mathbb{R}^6$. The vehicle dynamics are described by $\mathbf{f}$, and $\mathbf{d}$ represents disturbances to the vehicle, such as bumps in the floor or wind from other vehicles' fans.

Equation 7 represents the sensor dynamics. The vector $\mathbf{y} \in \mathbb{R}^6$ is the full state estimate formed by discrete-time (60 Hz) measurements of x, y, and θ. The function $\tau : \mathbb{R} \to \mathbb{R}$ represents the closed-loop system latency, not including delay due to actuator dynamics. Note that this latency may, in general, be time dependent. The function $\mathbf{n}$ represents measurement noise and includes errors in calibration of the map from pixel coordinates to floor coordinates.

The function $\mathbf{f}$ in (6) consists of a nonlinear input term and a (generally nonlinear) friction term. If we assume linear friction, perfect actuation ($\mathbf{u} = \mathbf{u_{des}}$), and perfect sensing ($\mathbf{y} = \mathbf{x}$), then we recover the equations presented in Section 2.2. These are repeated here for completeness:

$$\begin{aligned} m\ddot{x} &= -\mu\dot{x} + F\cos\theta \\ m\ddot{y} &= -\mu\dot{y} + F\sin\theta \\ J\ddot{\theta} &= -\psi\dot{\theta} + Tr_f \end{aligned} \tag{8}$$

where $F \triangleq F_L + F_R$ and $T \triangleq F_R - F_L$. The force ranges for F_L and F_R are $[0, F_{max}]$, and so the input space can be described by $(F_L, F_R)^T \in \mathbb{U} \triangleq [0, F_{max}] \times [0, F_{max}]$. Note that the model above has hovercraft-like dynamics, particularly because we have assumed linear viscous friction.

8.2. TIME DELAY

The closed-loop time delay, not including actuator delay, was determined experimentally to be 65 ms. This is the average time from when an event occurs on the floor to when a signal is sent to the actuators based on this event. The value was determined by replacing one of the identification blobs on the vehicle hat with a bank of LED's. When the LED's were lit, the identification blob would be obscured and the vision system would lose the vehicle. A controller was written that would turn the LED's off when the vision data indicated that the vehicle was missing and back on when the vision data indicated that the vehicle was visible. The vision data then consisted of a train of square waves indicating whether or not the vehicle was visible, and the period of these waves was twice the closed-loop time delay. Table 1.1 is a break-down of approximately where the time-delays occur.

delay (ms)	Event	elapsed time (ms)
0	LED on/off	0
8	Camera takes picture	8
33	Vision broadcasts data	41
6	Laptop receives data	47
1	Controller receives data	48
6	PIC receives control signal	54
11	LED on/off	65

Table 1.1 . The closed loop time delay of the system not including actuator delay. The table shows various factors that contribute to the delay.

8.3. LINEAR CONTROL TECHNIQUES

In this section, we discuss linearization of the model in equation 8. Although the model given is uncontrollable around any equilibrium, we describe a class of simple trajectories for which the error dynamics associated with tracking can be linearized to regain controllability. Results on stabilization of the error dynamics for a circular trajectory (described below) are then given, using both LQR and classical control techniques.

8.3.1 Linearization. The equilibria for the dynamics in equation (8) are any constant position and orientation with zero velocity. However, the linearized dynamics are not controllable around any such equilibria. To achieve controllability, we can consider, for example, the error dynamics around a constant velocity $\dot{x}_{nom}$ and heading θ_{nom}, yielding a reference state

$$[x_r(t_0) + t\dot{x}_{nom},\ y_r(t_0) + t\dot{y}_{nom},\ \theta_{nom},\ \dot{x}_{nom},\ \dot{y}_{nom},\ 0],$$

where $\dot{y}_{nom} = \dot{x}_{nom}\tan(\theta_{nom})$. The nominal inputs are $F_L = F_R = F_{nom} \triangleq (\eta\dot{x}_{nom})/(2\cos\theta_{nom})$. The error dynamics, denoting the error states with a subscript e, become

$$\begin{aligned} m\ddot{x}_e &= -\eta(\dot{x}_e + \dot{x}_{nom}) + F\cos(\theta_e + \theta_{nom}) \\ m\ddot{y}_e &= -\eta(\dot{y}_e + \dot{y}_{nom}) + F\sin(\theta_e + \theta_{nom}) \\ J\ddot{\theta}_e &= -\psi\dot{\theta}_e + Tr_f. \end{aligned} \tag{9}$$

The controllable equilibria of equation 9 are any constant $[x_e, y_e, \theta_e, \dot{x}_e, \dot{y}_e, \dot{\theta}_e]$ such that $\theta_e = \dot{x}_e = \dot{y}_e = \dot{\theta}_e = 0$. To track the straight line path exactly, we can specify that $(x_e, y_e) = (0, 0)$ and use linear control techniques to stabilize the error dynamics to this equilibrium (the origin).

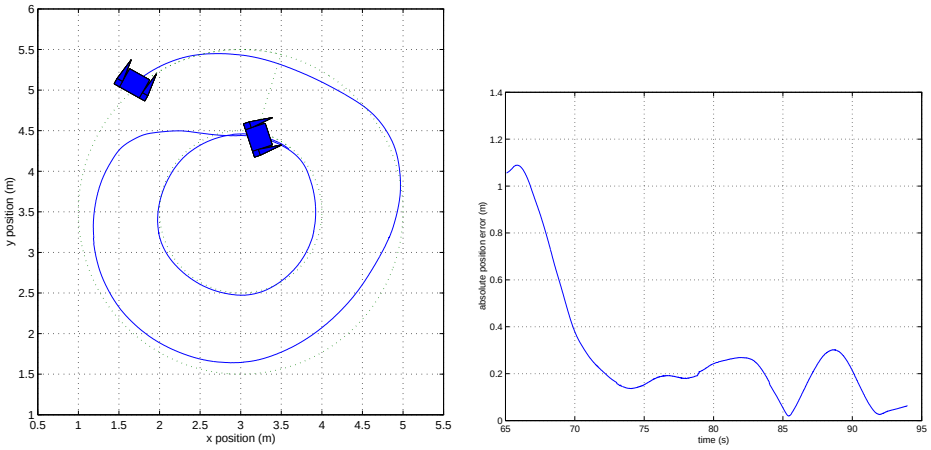
Rather than tracking a straight line, a vehicle motion more suited to the spatially constrained testbed floor is to trace a circular path with a constant radius and constant angular velocity. Such motion is most easily analyzed when the model in equation (8) is written in polar coordinates. As with tracking the straight line path above, tracking the circular path becomes a stabilization problem in the (polar) error dynamics, for which we design LQR controllers. Experimental results for this are given in the next section.

8.3.2 LQR Control. The error dynamics in equation 9 have been stabilized using an linear quadratic regulator (LQR) controller in simulation and on the testbed. An LQR controller has also been experimentally validated for stabilization along the circular path described above, namely a trajectory that moves with constant radius and angular velocity around the post obstacle in the middle of the testbed floor. Figure 1.14 depicts the response of the controller to a step change in radius in both an xy plot and a time history of the absolute position error. On the xy plot, the step command was received when the vehicle was at the position indicated by the inner vehicle on the figure. The solid line depicts the actual path the vehicle took, and the dashed line is the reference. On the error time history, the step command was received at the beginning of the time span of the plot. The gyro sensor was used for angular rate measurement during this experiment.

The optimal LQR gain matrix K for the vehicle before the step response is different from the optimal gain matrix after the step. This is due to the fact that the linearized A and B matrices depend on the radius (ρ) and angular velocity ($\dot{\xi}$) of the circular trajectory to be tracked. The above results, therefore, reflect LQR control that is gain scheduled between the two trajectories. The gain schedule we have designed is scheduled on thirty values of ρ between 0.1 m and 3.0 m, and ten values of $\dot{\xi}$ between 0.1 rad/s and 1.0 rad/s. Structural symmetry in the K matrix is exploited to allow negative angular velocities in the same magnitude range. Bilinear interpolation for each of the elements of K is used to allow reasonable tracking of any (ρ, $\dot{\xi}$) in the given ranges.

This general gain scheduled design allows us to track more general trajectories than indicated above, and can be used, for instance, to change directions in mid-flight operations. This maneuver has been experimentally validated on a triangular formation of three vehicles, where each vehicle uses it's own local copy of the LQR controller described above.

8.3.3 Classical Control. Classical lead compensators have been designed to stabilize the vehicle about a circular reference tra-



(a) xy position. Step command received when vehicle was at location indicated by inner vehicle on figure.

(b) Absolute position error vs. time. Step command received at t = 85s, the left edge of the plot.

Figure 1.14. LQR step response. Vehicle was commanded to increase radius from 1 m to 2 m while traveling at 0.5 m/s.

jectory. The primary difference between this approach and the LQR controller described above is that it is a dynamic compensator rather than a static set of gains, which helps to smooth out the effects of noisy vision data. Robustness to time delay can also be treated explicitly in controller design. Another significant difference in this controller implementation as compared to the LQR was that trajectory center, radius, and velocity were specified, but position along the trajectory was not, reducing the state dimension to 5. This type of controller would be useful in a situation where an obstacle avoidance algorithm specifies at each instant a desired velocity rather than position; see for example [27].

A nested-loop approach was used to design the controllers. First, a first order lead compensator of the form $C_\theta = K\frac{s+a}{s+b}$ was designed as an inner loop to control the orientation of the vehicle with respect to the dynamics linearized about driving at a constant velocity in a straight line. A proportional controller was then implemented as an outer loop to command the reference orientation to drive the vehicle to the correct radius. Velocity along the trajectory was also controlled using a static proportional gain. Figure 1.15 is the block diagram of this controller.

The primary advantage of the classical design methodology was the ease with which time delay could be explicitly taken into account to help

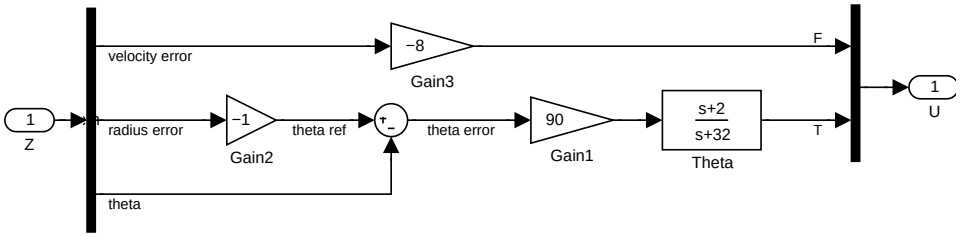


Figure 1.15. Schematic of classical controller. The error in radius is used to generate a reference angle, then lead compensation is used to control to this angle. Speed along the trajectory is regulated using proportional control.

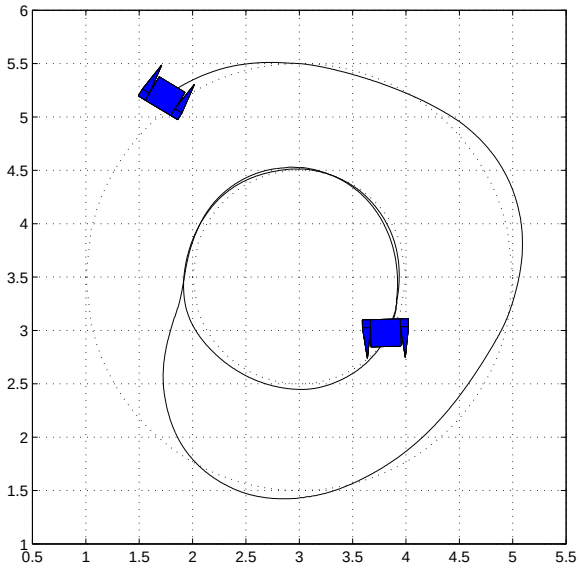


Figure 1.16. Step response xy plot. Vehicle was commanded to increase radius from 1 m to 2 m while traveling at 0.5 m/s.

ensure robustness. The controllers were designed to have at least 60 degrees of phase margin after taking into account 100 ms of time delay. The maximum bandwidth achieved by the inner loop (orientation) controller was 6 rad/s with 65 degrees of phase margin (after time delay taken into account) using the parameters $a = 2, b = 30, K = 90$ in the lead compensator given above. Bandwidth in the outer loop (radius) controller was 0.45 rad/s with 60 degrees of phase margin (again with time delay accounted for) using unity gain. A proportional gain of 8 was used to control the speed along the trajectory. Note that these controllers were designed for early vehicle prototypes and so these parameters would be

different for maximum performance on the newer vehicles. Figure 1.16 is an xy plot of the step response in radius from 1 m to 2 m at 0.5 m/s velocity. For the step out, 10% overshoot was observed in both simulation and testing. For the opposite case, the step in from 2 m to 1 m, 20% overshoot was observed in both simulation and testing.

8.4. NONLINEAR CONTROL TECHNIQUES

The linear techniques above do not necessarily respect the nonlinearities of the model or the input constraints, i.e. that fact that the two fan forces F_L and F_R are uni-directional and bounded by F_{max}. An explicit nonlinear feedback control law that does respect the nonlinearities of the model in equation 9 is currently under construction.

Optimization based control is a powerful tool in that such nonlinearities and constraints can explicitly be accounted for in the optimization problem. By optimization based control, we refer to the generation of optimal trajectories for path planning. The trajectories may be tracked by an inner-loop control or may implemented in a receding horizon fashion. Both numerical and experimental results for single vehicle optimization based control have been carried out and are detailed in the work by Chauvin, Sinègre and Murray [5]. The optimization itself is carried out using the Nonlinear Trajectory Generation software package [20]. Specifically, open-loop time optimal trajectories were generated on-line by NTG, for point to point motions of a testbed vehicle. Tracking of the trajectories was achieved using an LQR-based inner-loop control and the experimental results are given in [5]. Simulations for receding horizon control of single and multiple testbed vehicle are also detailed in [5] and [7], respectively.

Although the optimal control framework allows you to specify arbitrary (well-posed) problems with constraints, the trade-off of course is that we are subject to the complexities and reliability issues related to on-line optimization. Namely, while and LQR controller may go unstable when the inputs are saturating, an optimization based controller can go unstable if the computations take too long or fail to give an answer all together. These issues are discussed in [5].

9. FUTURE DIRECTIONS

The MVWT is in a constant state of flux due to the variety of needs various researchers at Caltech and elsewhere place upon it. We conclude our report by describing some of the advances we have planned for the near future that we hope will increase the usability of the testbed for new research.

First, we plan to add to and enhance the sensor suite available to the MVWT control designer by adding new or upgrading old sensors and by providing more complete sensor fusion and filtering in the software environment. In particular, integrating the gyro data and fusing it with the orientation data from the vision system will make available a better state estimate for the vehicle. Alternatively, accelerometers can be added to the vehicle so that it is not dependent on the vision system at all and therefore usable outside of our laboratory.

Second, we plan to explore other communications systems in addition to wireless Ethernet, such as Bluetooth, a standard for short range wireless communication that has a variable communications range dependent on the transmission power. Such communications systems are useful for battery-powered vehicles wherein the power draw of all systems needs to be minimized. In this direction, a backend to our communications software that utilizes the L2CAP Bluetooth protocol on the QNX operating system is currently under development. We are also exploring the possibility of simulating bandwidth limited and congested networks in our communications software so that the robustness of our control algorithms to such disturbances can be investigated.

Third, we plan to greatly increase the flexibility of our software systems on the vehicles and on the command computers. This will involve designing monitor software and user interfaces for easily issuing high level commands to the vehicles. We also have begun work toward providing standard interfaces between the low-level RHexLib software and other software architectures such as the Open Control Platform [28] and Ptolemy [4]. Finally, we have begun implementation of a new formal programming language called CCL [17] for writing verifiable control code.

10. CONCLUSION

Our goal with the Multi-Vehicle Wireless Testbed is to explore and validate new research advances in cooperative control by providing a challenging and realistic environment for their implementation. Our expectation is that this document, detailing the MVWT hardware, laboratory infrastructure, electronics design, software systems and basic modeling and control issues, will serve as a practical foundation for future applied work on the MVWT and in multi-vehicle systems in general.

Acknowledgments

The authors wish to thank Richard M. Murray, Jason Hickey, John Doyle, Steven Low and Michelle Effros for their support and mentorship during the development of the MVWT. We also wish to acknowledge

the contributions to the construction of the MVWT by many students over the past two years. Particular recognition in this regard goes to Jason Melzer, Kelly Klima, Melvin Flores and Cristian Tapus. This work was funded in part by AFOSR grants F49620-01-1-0460, F49620-01-1-0361 and F49620-01-1-0227 and DARPA grants F33615-98-C-3613 and F30602-01-0577. Steve Waydo was funded in part by the Fannie and John Hertz Foundation.

References

[1] Alwayse website. URL: http://www.alwayse.co.uk.

[2] MVWT website. URL: http://www.cds.caltech.edu/~mvwt.

[3] Rhexlib. URL: http://sourceforge.net/projects/rhex/.

[4] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *International Journal of Computer Simulation (Special Issue on Simulation in Software Development*, 4:155–182, April 1994.

[5] J. Chauvin, L. Sinegre, and R. M. Murray. Nonlinear trajectory generation for the caltech multi-vehicle wireless testbed. In *Submitted to 2003 European Control Conference*, 2003.

[6] R. D'Andrea. Robot soccer: A platform for systems engineering. *Computers in Education Journal*, 10(1):57–61, 2000.

[7] W. B. Dunbar and R. M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[8] J. Evans, G. Inalhan, J. Jang, R. Teo, and C. Tomlin. Dragonfly: A versatile UAV platform for the advancement of aircraft navigation and control. In *Proceedings of the 20th IEEE Digital Avionics Systmes Conference*, October 2001.

[9] J. A. Fax and R. M. Murray. Graph laplacians and stabilization of vehicle formations. In *15th IFAC World Congress on Automatic Control*, 2002.

[10] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. In *15th IFAC World Congress on Automatic Control*, 2002.

[11] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM, April 1985*, 32(2):374–382, April 1985.

[12] Ryan Franz, M. B. Milam, and J. Hauser. Applied receding horizon control of the caltech ducted fan. In *Proceedings of the 2002 American Control Conference*, 2002.

[13] A. Granicz and J. Hickey. Phobos: A front-end approach to extensible compilers. In *International Conference on System Sciences (HICSS-36)*, Hawaii, 2002.

[14] A. Jadbabaie and J. E. Hauser. Control of a thurst vectored flying wing: A receding horizon-lpv approach. *International Journal of Robust and Nonlinlear Control*, Submitted.

[15] D. Jia, B. H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, February 2002.

[16] E. Johnson, S. Fontaine, and A. Kahn. Minimum complexity uninhabited air vehicle guidance and flight control system. In *AIAA Digital Avionics Conference*, 2001.

[17] E. Klavins. A formal language approach to embedded systems design and verification. In *Conference on Decision and Control*, 2003. Submitted for review.

[18] E. Klavins and U. Saranli. Object orient state machines. *Embedded Systems Programming Magazine*, 2002. In Press.

[19] L. Meirovich. *Elements of Vibration Analysis*. McGraw-Hill, 2^{nd} edition, 1986.

[20] M. B. Milam, K. Mushambi, and R. M. Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proceedings of the IEEE Conference on Decision and Control*, 2000.

[21] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray. Cooperative control of multi-vehicle systems using cost graphs and optimization. In *Submitted to 2003 American Control Conference*, 2003.

[22] R. Olfati-Saber and R. M. Murray. Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *Proceedings of the 41st Conference on Decision and Control*, 2002.

[23] R. Olfati-Saber and R. M. Murray. Consensus protocols for networks of dynamic agents. In *Submitted to 2003 American Control Conference*, 2003.

[24] U. Saranli, M. Buehler, and D. E. Koditschek. RHex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, July 2001.

[25] A. Stubbs, V. Vladimerou, A. Vaugn, and G. E. Dullerud. Development of a vehicle network control testbed. In *Proceedings of the American Control Conference*, Alaska, 2002.

[26] QNX Realtime Systems. The QNX real time operating system. URL: http://www.qnx.com/.

[27] S. Waydo and R. M. Murray. Vehicle motion planning using stream functions. In *Accepted: IEEE International Conference on Robotics and Automation*, 2003.

[28] L. Wills, S. Kannan, S. Sander, M. Guler, B. Heck, J. V. R Prasad, D. P. Schrage, and G. Vachtsevanos. An open platform for reconfigurable control. *IEEE Controls Systems Magazine*, 21(3), June 2001.