

# Autonomous Robotics Laboratory: Hardware Demonstration of Cooperative Formation Control Laws

Kristen E. Holmstrom\*, Roy Palacios†, Brock Spratlen‡,  
Cynthia Ochoa§, Lisa Warren¶, and Lesley A. Weitz||

*Space Engineering Institute at Texas A&M University, College Station, TX, 77843, U.S.A*

**Autonomous robots are expected to be a key part of future space exploration, and formation control of robotic vehicles may be one autonomous task used in lunar or Martian exploration. This paper presents a cooperative control law to drive a group of autonomous robots to a desired formation. The cooperative control laws were implemented using the Autonomous Robotics Laboratory at Texas A&M University, and several challenges in the software and hardware implementation were encountered. These challenges and hardware results are presented in the paper. The results reveal hardware requirements for a successful implementation of this autonomous-robotics application.**

## I. Introduction

NASA's vision for space exploration includes returning to the moon by 2020 with the goal of living on the moon for extended periods of time in order to prepare for longer-duration manned-missions to Mars.<sup>1</sup> Robots are expected to play a key role in future space exploration of the moon and Mars including tasks such as manned and unmanned surface exploration, transport of materials between landing and exploration sites, and the assembly or construction of a lunar habitat. NASA and others have already developed a number of robots for planetary exploration and scientific experiments, and these robots have had varying levels of autonomy. The Mars Rovers, Spirit and Opportunity, for example, were teleoperated from the surface of the Earth,<sup>2</sup> and Chariot has been developed for either manned operation or teleoperation from the surface of the moon, Mars, or Earth.<sup>3</sup> In contrast, fully autonomous robots are able to execute complex tasks with little or no human intervention. Autonomous robotic systems can aid in lunar or Martian exploration and habitation by reducing workload for the astronauts on the planetary surface or mission control on Earth.

Challenges in the development of autonomous systems are numerous. Operation in unknown environments is especially challenging where autonomous decision-making must be robust to unanticipated events. The Autonomous Robotics Team, which is a part of the Space Engineering Institute at Texas A&M University, seeks to investigate and develop the necessary technologies for autonomous robotic systems. The team has developed the Autonomous Robotics Laboratory to demonstrate a number of autonomous tasks using simple, differentially-driven robotic platforms. The Autonomous Robotics Laboratory is comprised of several subsystems including a global-positioning system that measures robot states using an overhead camera system, a wireless-communications network, and a central PC that manages autonomous tasks including trajectory planning and tracking.<sup>4</sup>

This paper describes the implementation and demonstration of cooperative formation control laws in the Autonomous Robotics Laboratory. Autonomous formation control laws are used to drive a group of robots to a desired formation. Applications on the lunar or Martian surface may include terrain mapping, search

---

\*M.S. Student, Aerospace Engineering, 3141 TAMU, and AIAA Student Member.

†Undergraduate Student, Electrical Engineering, 3118 TAMU.

‡Undergraduate Student, Computer and Electrical Engineering, 3118 TAMU.

§Undergraduate Student, Aerospace Engineering, 3118 TAMU.

¶Undergraduate Student, Aerospace Engineering, 3118 TAMU.

||Ph.D. Candidate (Graduate Mentor), Aerospace Engineering, 3141 TAMU, and AIAA Student Member.

and rescue, and material transportation. The cooperative control laws were developed by researchers at Texas A&M University,<sup>5</sup> and the hardware demonstration is used to support the theoretical developments. Additionally, the hardware implementation and testing reveals hardware and software limitations and the effects of time delays on system stability.

The paper is organized as follows. In Section II, the subsystems that comprise the Autonomous Robotics Laboratory are described. The cooperative formation control laws are presented in Section III. Implementation challenges are discussed in Section IV, and hardware-testing results are presented in Section V. Lastly, conclusions and future work are in Section VI.

## II. Autonomous Robotics Laboratory

The Autonomous Robotics Team developed the Autonomous Robotics Laboratory over three academic semesters. The lab includes several subsystems that enable testing of autonomous robotics tasks. Subsystems include three iRobot Create robotic platforms, a ZigBee wireless-communication system, a global-positioning system that measures robot positions and orientations using an overhead camera, and a central PC that manages the autonomous tasks being tested. The system components were individually developed and tested, and have been integrated to allow rapid development and implementation of various autonomous tasks.

### A. Robotic Platform

The iRobot Create platform was selected for rapid hardware implementation and development. The iRobot Create offers several advantages in implementation with its embedded sensors (wheel odometers, bump sensors, and infrared detectors) and command module with an 8-bit microcontroller.<sup>6</sup> The command module enables the user to embed algorithms onboard the robot using C/C++. The platform is differentially driven; therefore, each wheel velocity is individually controlled, which allows 360-degree rotational motion with no translation.

### B. Wireless Communication

The Autonomous Robotics Laboratory is equipped with a ZigBee wireless-communication network. The ZigBee communication standard allows wireless communication between all nodes in the system compared with the Bluetooth standard, which is point-to-point communication only. The ZigBee module is designed to transmit at a rate of 250 Kbits/second, which is relatively slow compared to Bluetooth capabilities of 1 Mbits/second; however, network communication is more ideal for this application. Accurate data transmission is at a frequency of about 5 Hz, and the speed of the robot's onboard microcontroller to parse the data packets is the primary limiting factor in the speed of the input/output data frequency.

### C. Global-Positioning System

The team has implemented an overhead camera and image-processing system to measure inertial positions and orientations of the robotic platforms. Whereas a real lunar or Martian environment does not support an overhead camera, the overhead camera can represent a geosynchronous satellite system. A MDCS2 monochrome IEEE1394 camera was mounted above the laboratory area in order to determine the robot states within a  $2 \times 2.75$ -meter area. Robot identification is accomplished by using distinct patterns assigned to each robotic platform. The image-processing software, developed by Texas A&M University graduate students, James Doebbler and Kevin Daugherty, determines inertial positions and orientations of the robots by first locating the unique patterns in the field of view, and then by finding the centroid and orientation of each pattern. The software outputs the measured  $x$ ,  $y$ , and  $\theta$  of each robot in its field of view. This state data from the camera is constantly updated and sent to the central PC. Figure 1 shows the view of three unique patterns from the overhead camera. The image-processing software locates and outlines each pattern, and the orientation of the pattern is also shown by the pink line.

### D. Central PC

Due to the lack of computational power available on the iRobot-Create microcontroller, a centralized system has been implemented to do the majority of the processing for the robots. The central PC processes

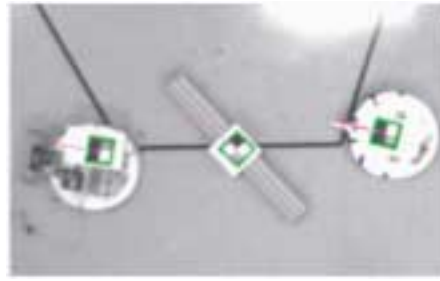


Figure 1. View from the overhead camera with unique patterns located by the image-processing software.

raw measurements from the overhead camera using a Continuous-Discrete Extended Kalman Filter, which combines the raw position and orientation measurements with a robot model to determine the best estimates of each robot’s position and orientation.<sup>7</sup> The central PC also computes control inputs to each robot using the desired control algorithms and communicates velocity commands to the robots using the wireless-communication system. Common functionality has been abstracted into classes to build a common system, and the control algorithms can be easily changed to permit testing of various autonomous tasks.

### E. Subsystem Integration

Figure 2 shows the integration of the subsystems that comprise the Autonomous Robotics Laboratory. The process flow within the system software can be described as follows: 1.) camera data is processed by the camera PC and sent to the central PC at a rate of approximately 10 Hz using UDP communication; 2.) the central PC then passes these camera measurements to the Kalman Filter, which gives a best state estimate for each of the robots; 3.) the control laws use this best state estimate to calculate commanded velocities for each robot; and, 4.) the commanded velocities are then sent to each robot. The central PC is able to process camera packets at a rate of approximately 8 Hz, causing some camera packets, which are being received at a faster rate, to be ignored. The time required to send velocity commands to each of the three robots is the most significant contributor to the overall process time. Velocity commands are sent sequentially to the three robots due to the design of the wireless-communication network. This system constraint is further discussed in Section IV.

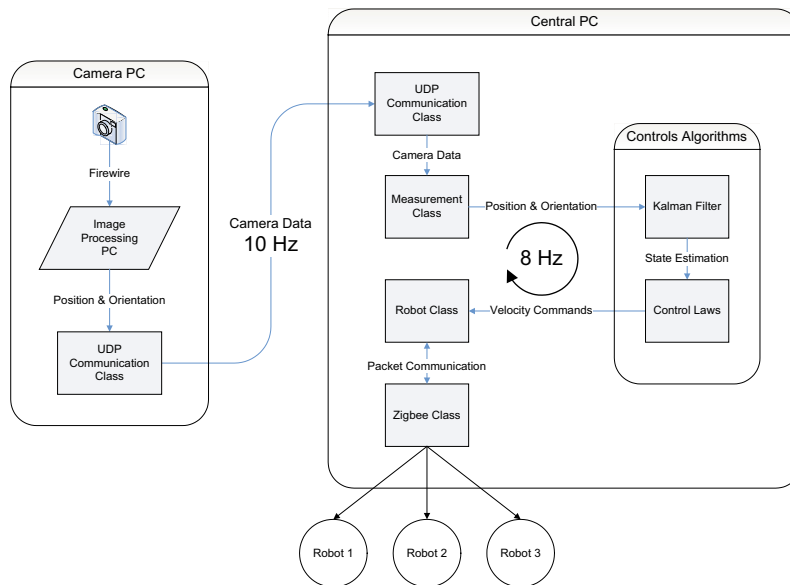


Figure 2. Subsystem integration.

### III. Cooperative Control Laws

Decentralized, cooperative control laws can be used to drive a group of robots to a desired formation. Cooperative control laws couple separate robotic vehicles through shared state information as defined by a communication structure. For example, the cooperative control laws presented here assume a leader-follower communication structure. Therefore, each robot has state information from an assigned lead robot, and control inputs are designed to maintain a desired spacing relative to the assigned leader. In a decentralized control scheme, each robot has the instrumentation necessary to determine its own control inputs. This implementation can be contrasted with a centralized control implementation, where one central control authority determines the control inputs to each individual robot.

In this section, the robot model is presented. The control law design is motivated by a linear representation of the robot model. Simulation results are shown for one reference trajectory designed for the hardware demonstration in the Autonomous Robotics Laboratory.

#### A. Robot Equations of Motion

Figure 3 shows an overhead view of the robot. The inertial position,  $x$  and  $y$ , and orientation,  $\theta$ , of the robot are defined in the figure. A body-fixed reference frame is shown aligned with the heading angle of the robot. The robot's motion is constrained to lie along the heading angle in the  $\hat{b}_1$  direction; therefore, the robot's velocity is zero in the  $\hat{b}_2$  direction.

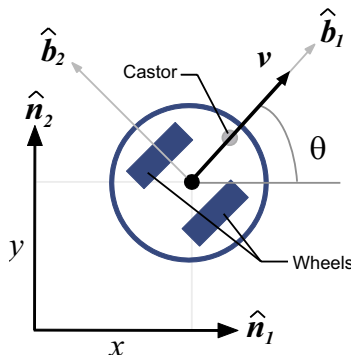


Figure 3. Overhead view of the differentially-driven robot. Robot states and reference frames are defined.

The equations of motion for the robot are shown in the equations below where the robot states are  $x$ ,  $y$ , and  $\theta$ . The velocity,  $v$ , and angular velocity,  $\omega$ , are the control inputs.

$$\dot{x} = v \cos \theta; \quad \dot{y} = v \sin \theta; \quad \dot{\theta} = \omega \quad (1)$$

Control inputs to the iRobot Create are the left and right wheel velocities, which are easily found from  $v$  and  $\omega$ .

$$v_L = v - \frac{d}{2}\omega; \quad v_R = v + \frac{d}{2}\omega \quad (2)$$

Here,  $d$  is the diameter of the wheelbase.

The properties of this nonlinear vehicle model allow the model to be represented using a double-integrator form, i.e.,  $\dot{x} = u$  and  $\dot{y} = w$ . This double-integrator form is the control design space, and a linear transformation can be written to determine the original robot controls.<sup>5</sup> The transformation equation to the robot control inputs is shown below, where  $u$  and  $w$  are the design-space control inputs.

$$\begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \frac{1}{v} \begin{bmatrix} \dot{x} & \dot{y} \\ -\frac{\dot{y}}{v} & \frac{\dot{x}}{v} \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} = T(x, y) \begin{bmatrix} u \\ w \end{bmatrix}; \quad v = \sqrt{\dot{x}^2 + \dot{y}^2} \quad (3)$$

Note that the robot control inputs are now the robot acceleration along the heading angle,  $\dot{v}$ , and the angular velocity,  $\omega$ . This transformation allows the cooperative control laws to be developed using the double-integrator form, which decouples the control design in the  $x$  and  $y$  directions.

## B. Cooperative-Control Law Design

The control laws are designed by defining spacing-error terms between vehicles pairs in the formation. Each vehicle is assigned a lead vehicle that it follows. Whereas the error equations are shown in the  $x$  direction only, the control development is identical in the  $y$  direction.

$$\begin{aligned} e_1 &= x_r - x_1 - d_1; & \dot{e}_1 &= \dot{x}_r - \dot{x}_1; & \ddot{e}_1 &= \ddot{x}_r - \ddot{x}_1 = \ddot{x}_r - u_1 \\ e_2 &= x_1 - x_2 - d_2; & \dot{e}_2 &= \dot{x}_1 - \dot{x}_2; & \ddot{e}_2 &= \ddot{x}_1 - \ddot{x}_2 = u_1 - u_2 \\ e_3 &= x_2 - x_3 - d_3; & \dot{e}_3 &= \dot{x}_2 - \dot{x}_3; & \ddot{e}_3 &= \ddot{x}_2 - \ddot{x}_3 = u_2 - u_3 \end{aligned} \quad (4)$$

The  $e_1$  equation is the spacing error of the first vehicle in the formation with respect to a reference trajectory denoted by  $x_r$ . The desired distances between vehicle pairs are denoted by the constants  $d_1$ ,  $d_2$ , and  $d_3$ .

The general form of the control law can be written with spacing-error terms or using the definitions of the spacing errors in Equation (4).<sup>5</sup> Equation (5) reveals that there are five terms in the general control form: 1.) position error with respect to preceding vehicle, 2.) velocity error with respect to preceding vehicle, 3.) position error with respect to reference position, 4.) velocity error with respect to reference velocity, and 5.) acceleration of the reference trajectory.

$$\begin{aligned} u_i &= k_{p_i} e_i + k_{v_i} \dot{e}_i + c_{p_i} (e_1 + e_2 + \dots + e_i) + c_{v_i} (\dot{e}_1 + \dot{e}_2 + \dots + \dot{e}_i) + \ddot{x}_r \\ &= k_{p_i} (x_{i-1} - x_i - d_i) + k_{v_i} (\dot{x}_{i-1} - \dot{x}_i) + c_{p_i} (x_r - x_i - \sum_{j=1}^i d_j) + c_{v_i} (\dot{x}_r - \dot{x}_i) + \ddot{x}_r \end{aligned} \quad (5)$$

Different tracking schemes are included in this general control form. For a lead-robot tracking scheme, the gains on the errors with respect to the reference trajectory are set to zero ( $c_p, c_v = 0$ ). This scheme may be used if the reference trajectory is not known by all robots in the formation; therefore, each robot tracks its assigned lead robot only. For a reference-trajectory tracking scheme, the gains on the errors with respect to the lead robot are set to zero ( $k_p, k_v = 0$ ). This scheme may be used if state information about the assigned lead robot is unknown.

## C. Control-Law Simulations

Two reference trajectories were designed to fit within the  $2 \times 2.75$ -meter field of view of the overhead camera: a piece-wise, constant-velocity trajectory and a circular trajectory. Figure 4(a) shows a piece-wise trajectory that has constant velocity segments as shown in Figure 4(b). The camera field of view is denoted by the black, dashed lines.

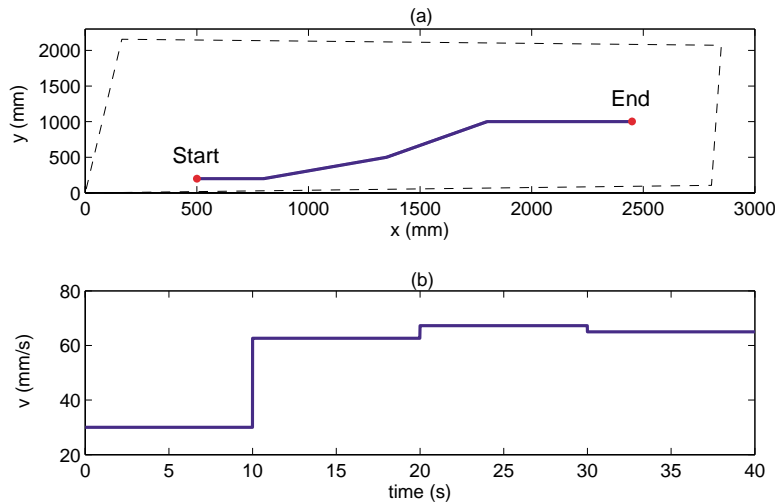


Figure 4. Piece-wise reference trajectory (a) and velocity along the trajectory (b).

The formation control laws were simulated using MATLAB to determine control gains that bound control inputs to the robotic platforms. Figure 5 shows the simulated robot formation as a function of time using a lead-robot, reference-trajectory tracking control law with  $k_p = k_v = c_p = c_v = 0.4$ . Robot 1 tracks the reference trajectory, Robot 2 tracks Robot 1 and the reference, and Robot 3 tracks Robot 2 and the reference. The simulation results show that the robots converge to their desired separation distances with respect to their assigned lead robots. Because the reference trajectory has discontinuous reference velocities, there are initial-condition errors at each new segment of the trajectory.

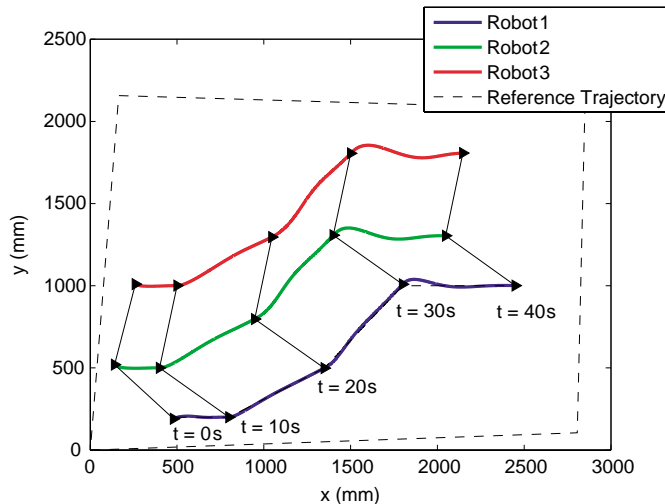


Figure 5. Simulated formation using a lead-robot, reference-trajectory tracking control scheme.

## IV. Hardware-Implementation Challenges

Several challenges were encountered in the hardware and software implementation of the cooperative formation control laws. The cooperative control laws were designed for a continuous, decentralized implementation; however, the actual implementation in the Autonomous Robotics Laboratory was discrete and centralized. In addition, the control inputs to the iRobot Create are the left and right wheel velocities, which are easily found from  $v$  and  $\omega$ , whereas the double-integrator design space leads to control inputs  $\dot{v}$  and  $\dot{\omega}$ . These challenges are discussed below.

### A. Description of Discrete Implementation

Software processing yields a system that is intrinsically discrete. The camera PC processes images from the overhead camera at discrete points in time (about 10 Hz), and the maximum rate at which the best state estimate of the system can be updated on the central PC is 10 Hz. Additionally, robot velocity commands cannot be updated in a continuous manner. Each robot's onboard microcontroller must parse discrete velocity commands received through wireless communication from the central PC, and commanded wheel velocities are updated through the robot's control interface.

### B. Decentralized vs. Centralized Implementation

The controls laws were designed to be implemented in a decentralized system; however, due to the lack of computational power available onboard each robot, a centralized system has been implemented. Additionally, the wireless-communication protocol was designed such that a command sent by the central PC, intended for one robot, is received and processed by all of the robots. If a robot receives a command intended for another robot, it is ignored. This means that updating the velocity commands for  $n$  robots in the system requires the central PC to send  $n$  sequential velocity commands. A planned delay of 20 ms was implemented on the central PC to allow time for each robot in the system to parse a command. An inherent characteristic of this

process is that each robot receives commands at different times than the other robots. This implementation reduces system stability for the robots that receive velocity commands with larger delays.

### C. Robot Control Inputs

The cooperative control laws yield robot commands  $\dot{v}$  and  $\omega$  (acceleration and angular velocity); however, the iRobot Create platform requires commands in the form of  $v$  and  $\omega$ . Two different approaches to implement the cooperative controller on the iRobot Create were considered.

The first approach considered was to send the  $\dot{v}$  and  $\omega$  commands determined by the control algorithm directly to the robot. After the command is received, each robot's onboard microcontroller uses a first-order approximation to determine incremental changes in the wheel velocities until the next command is received by the robot.

$$v(t + \Delta t) = v(t) + \dot{v}\Delta t \tag{6}$$

The second method considered was to use the central PC to send  $v$  and  $\omega$  commands each time a camera packet was received. This method assumes that the velocity remains constant until the next camera update has been received and processed by the central PC and sent to the robot. Figure 6 illustrates the differences between these two approaches. The dark blue line in Figure 6 represents the ideal application of the control-law output to the robotic system: a constant acceleration between each command received from the central PC (shown above by the vertical, dashed gray lines). The red line represents actual robot velocity versus time using the first method described above, which would allow for multiple incremental changes in robot velocity between each command received from the central PC. The light blue line represents actual robot velocity versus time for the second method described above, which maintains a constant velocity between each command received from the central PC.

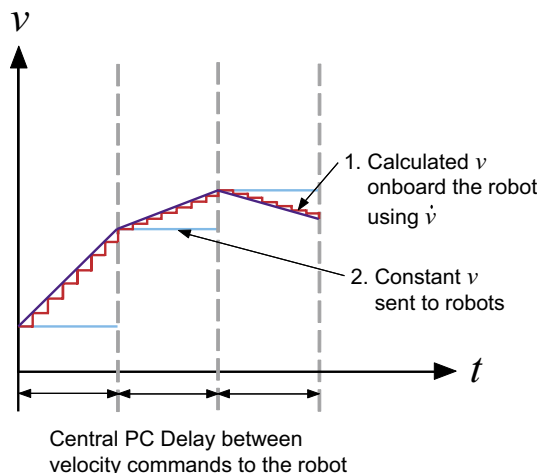


Figure 6. Different methods of commanding robot velocities.

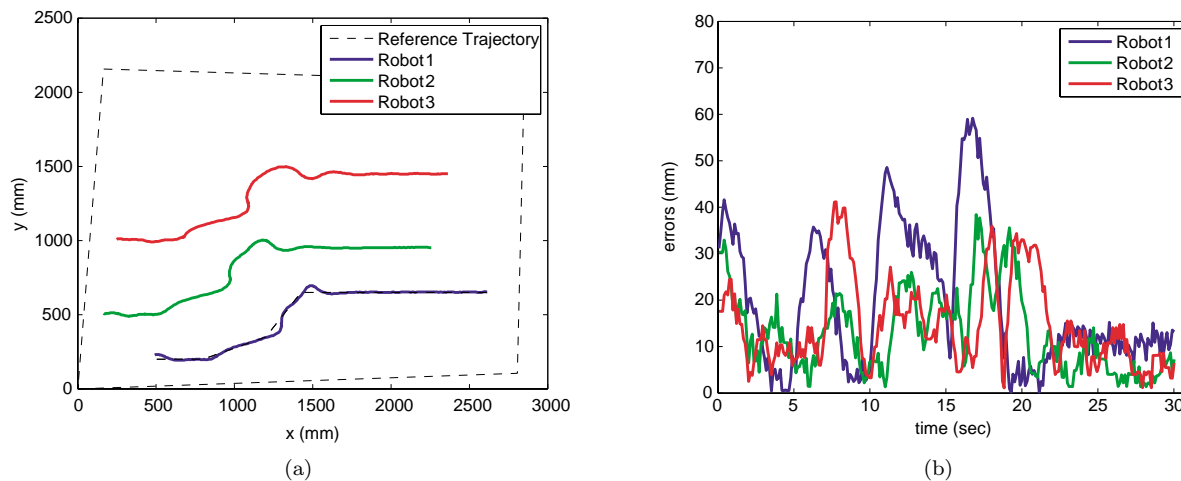
Due to the discrete nature of our robotic system, it is not possible to implement the ideal application of the control law output where the robot velocity is continuously updated. The first approach described above has the potential to be very close to the ideal implementation, but it was found through experimentation that the onboard microcontrollers do not have enough computational power to provide any benefit over the simpler implementation (the second approach). The second approach does not require much computational power on each robot's microcontroller, but the actual motion of the system does not exactly represent the motion expected by the control laws. This implementation creates a lag in the commanded velocities, which reduces system stability in some cases.

## V. Hardware Results

Hardware testing of the cooperative control laws investigated the stability and performance of three different tracking schemes:

1. Lead-robot tracking only (LR);
2. Reference-trajectory tracking only (REF);
3. Lead-robot and reference-trajectory tracking (LR + REF).

Figure 7(a) shows the inertial positions of the three robots for the lead-robot and reference-trajectory tracking scheme with control gains  $k_p = k_v = c_p = c_v = 0.5$ . Figure 7(b) shows the Euclidean spacing errors as a function of time. Note that the errors are measured with respect to each assigned lead robot. For example, the error for Robot 1 is measured with respect to the reference trajectory, and the error for Robot 2 is measured with respect to Robot 1.



**Figure 7.** (a) Inertial positions of the robots for the piece-wise reference trajectory and the lead-robot, reference-trajectory tracking control law. (b) Euclidean spacing errors between robots.

Three different performance metrics are presented to compare the convergence of the different tracking schemes and different control gains. The error “energy” represents the area under the spacing error function for each robot relative to its assigned lead. The maximum error is the greatest spacing error over the duration of the test, and the error at the end is the robot’s error with respect to its lead robot at the end of the test.

The results in Table 1 are for the piece-wise trajectory shown in Figure 4. The control gains were changed to determine how the errors in the formation are affected by the choice of gains. For the lead-robot, reference-trajectory tracking scheme (LR+REF), increasing the control gains from 0.25 to 0.50 decreased the spacing errors in the formation. However, increasing the gains to 1.00 caused the formation to become unstable by saturating the robot controls. For this control scheme, the errors in the formation accumulate along the vehicles; therefore, errors in the third robot’s position and velocity include the errors from the first and second robots. The reference-trajectory tracking scheme had improved convergence for all gains. In this case, the effects of the gains are easily observable. The higher gains result in lower total error energy for the three robots. Here, the errors for each robot are with respect to the reference trajectory; therefore, the cumulative error effects on the third robot do not occur. The lead-robot tracking scheme was tested for a number of control gains, and was not stable. The instability may be attributed to the aggressive piece-wise trajectory. The order that velocity commands were sent was reversed, so that Robot 3 received its velocity command before Robot 1 and Robot 2 to try to mitigate the delay effects; however, the formation was unstable in this case as well.

A circular trajectory was also designed and tested. Figure 8 shows the robot positions for the lead-robot, reference-trajectory tracking scheme with control gains  $k_p = k_v = c_p = c_v = 0.5$ . The formation was stable and converged to the desired formation despite poor initial conditions. Note that the trajectories of the three

Table 1. Piece-Wise Trajectory-Tracking Results.

	LR+REF ( $k, c = 0.25$ )	LR+REF ( $k, c = 0.50$ )	REF ( $k, c = 0.25$ )	REF ( $k, c = 0.50$ )	REF ( $k, c = 1.00$ )
<b>Error “Energy” (mm-s)</b>					
Robot 1	942.59	588.69	1722.70	596.31	646.89
Robot 2	670.56	388.75	242.88	390.29	253.44
Robot 3	704.67	445.58	274.96	565.68	233.35
<b>Maximum Error (mm)</b>					
Robot 1	81.45	59.17	150.16	60.63	54.04
Robot 2	46.55	38.40	15.78	56.25	34.97
Robot 3	55.44	41.18	35.58	62.25	34.33
<b>Error at End (mm)</b>					
Robot 1	13.18	13.17	59.48	9.40	9.04
Robot 2	1.38	7.25	4.87	7.45	3.94
Robot 3	11.71	6.45	2.48	7.92	4.60

robots overlap because the control laws are designed for constant-distance spacing in the inertial reference frame. The accumulation of errors and the time-delay effects are evident in the motion of the third robot, which shows greater oscillation when converging to its desired spacing with respect to the second robot.

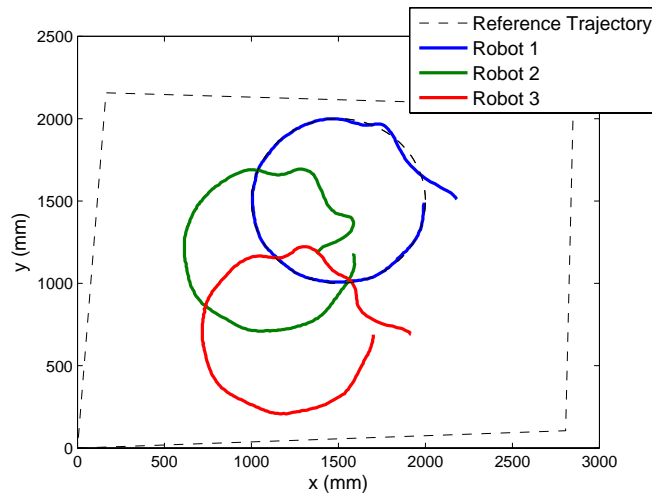


Figure 8. Inertial positions of the robots for the circular reference trajectory and the lead-robot, reference-trajectory tracking control law.

Hardware results for the circular reference trajectory are presented in Table 2. The lead-robot tracking scheme was tested for this trajectory. When the velocity commands were sent in the sequence 123 (Robot 1 received velocity commands followed by Robot 2 and then Robot 3), the formation was unstable. However, changing the order of velocity commands did yield a stable formation as shown in Table 2. Results for the circular reference trajectory still show that including reference-trajectory tracking reduces spacing errors and improves convergence.

Table 2. Circular Trajectory-Tracking Results.

	REF ( $k, c = 1.00$ ) Good ICs	LR+REF ( $k, c = 0.50$ ) Good ICs	LR+REF ( $k, c = 0.50$ ) Bad ICs	LR ( $k, c = 0.50$ ) Order 312	LR ( $k, c = 0.50$ ) Order 321
<b>Error “Energy” (mm-s)</b>					
Robot 1	402.78	296.00	616.57	447.71	550.36
Robot 2	258.91	409.17	1005.50	655.99	940.13
Robot 3	143.25	392.34	946.83	797.78	1654.60
<b>Maximum Error (mm)</b>					
Robot 1	53.78	34.59	181.45	45.41	58.23
Robot 2	43.54	54.95	407.91	57.55	66.35
Robot 3	18.10	37.74	442.49	62.80	103.65
<b>Error at End (mm)</b>					
Robot 1	16.01	11.51	12.24	13.20	11.40
Robot 2	8.43	10.08	9.39	25.03	5.14
Robot 3	4.51	12.73	12.84	35.36	66.30

## VI. Conclusions

The Autonomous Robotics Team has demonstrated the use of cooperative control laws to control a three-robot formation. The cooperative control laws are functions of spacing errors between robots in order to drive formation errors to zero. A linear transformation relates control inputs designed for a linear representation of the robot model to robot velocities. Several challenges were encountered in the implementation of the control laws including the discrete nature of the software architecture, the calculation of velocities sent to the robots, and the mitigation of time delays in the system. The cooperative control laws were tested for a variety of control gains and for two different trajectories. Hardware results showed that the reference trajectory can affect the stability of the formation. The effects of processing time delays were evident for the aggressive piece-wise trajectory, and these delays resulted in formation instabilities when reference-trajectory tracking was not included in the control law. However, the circular trajectory yielded better results, and the formation was stable for the lead-robot tracking scheme when the order in which velocities were sent to the robots was modified. The hardware demonstrations show that these cooperative control laws could be used to control a group of robots. Greater onboard computational power would enable a decentralized implementation, which would yield improved system stability and performance by reducing the effects of processing delays.

## Acknowledgments

The authors would like to thank the NASA Johnson Space Center for sponsoring the Space Engineering Institute, the Director of the Space Engineering Institute, Magdalini Lagoudas, for her support of the Autonomous Robotics Team, and the faculty mentor of the team, Dr. John E. Hurtado. The authors would also like to acknowledge some past undergraduate students on the Space Engineering Institute (SEI) robotics team for their work to develop the Autonomous Robotics Laboratory including Amy Bolon, Jesse Bowes, Jane Nguyen, Albert Soto, and Matt Wilson.

## References

- <sup>1</sup>National Aeronautics and Space Administration, “The Vision for Space Exploration,” Tech. rep., February 2004.
- <sup>2</sup>Leger, P. C., Trebi-Ollennu, A., Wright, J. R., Maxwell, S. A., Bonitz, R. G., Biesiadecki, J. J., Hartman, F. R., Cooper, B. K., Baumgartner, E. T., and Maimone, M. W., “Mars Exploration Rover Surface Operations: Driving Spirit at Gusev Crater,” *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, 2005, pp. 1815–1822.
- <sup>3</sup>Harrison, D. A., Ambrose, R., Bluethmann, B., and Junkin, L., “Next Generation Rover for Lunar Exploration,” *IEEE Aerospace Conference*, 2008, pp. 1–14.
- <sup>4</sup>Bolon, A., Holmstrom, K., Nguyen, J., Palacios, R., Soto, A., Spratlen, B., and Weitz, L. A., “Cooperative Transportation of a Flexible Object Towards Construction of a Martian Habitat,” *AIAA Region IV Student Paper Competition*, Houston, TX, April 2008.
- <sup>5</sup>Weitz, L. A., Hurtado, J. E., and Sinclair, A. J., “Decentralized Cooperative Control Design for Multivehicle Formations,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 970–979.
- <sup>6</sup>iRobot.com, *iRobot Command Module: User’s Manual*, Accessed: March 2009, <http://www.irobot.com/filelibrary/pdfs/hrd/create/Commandv2.pdf>.
- <sup>7</sup>Crassidis, J. L. and Junkins, J. L., *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC Applied Mathematics and Nonlinear Science Series, Boca Raton, Florida, 2004.