COOPERATIVE AND SUPERVISORY CONTROL FOR PAYLOAD MANIPULATION

A Senior Scholars Thesis

by

KRISTEN HOLMSTROM

Submitted to the Office of Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

April 2009

Major: Aerospace Engineering

COOPERATIVE AND SUPERVISORY CONTROL FOR PAYLOAD MANIPULATION

A Senior Scholars Thesis

by

KRISTEN HOLMSTROM

Submitted to the Office of Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by:

Research Advisor:	John E. Hurtado
Associate Dean for Undergraduate Research:	Robert C. Webb

April 2009

Major: Aerospace Engineering

ABSTRACT

Cooperative and Supervisory Control for Payload Manipulation. (April 2009)

Kristen Holmstrom
Department of Aerospace Engineering
Texas A&M University

Research Advisor: Dr. John E. Hurtado
Department of Aerospace Engineering

There are many tasks done by humans today that could be done by robots. One environment where robots are especially useful is space. Because of the limitations of astronauts, robots could be sent to a planetary environment to prepare a habitat. This thesis considers two problems that arise when considering sending robots to a planetary environment. The first problem is the cooperative control of two robots manipulating flexible payloads. The second problem investigates the communication between humans and robots using vision techniques. The goal of each problem is to produce a hardware demonstration in a laboratory environment to demonstrate some of the skills necessary to implement the ideas in a planetary environment.

Several subsystems were developed by the Space Engineering Institute's Robotics Space Colonization Team including an overhead camera system, a wireless communication network, a Kalman filter, and Central PC System Architecture. Without these systems, neither project could be accomplished. The first project goal was completed through several phases beginning with theoretical development of the robot and flexible object models. Simulation results proved the theory to be true and hardware demonstrations proved that the equations were robust. The second project goal was completed by introducing more subsystems into the robotics lab including a webcam with image recognition software, battery information functions, path planning algorithms, and trajectory tracking control laws. A hardware demonstration was produced that showed the robot performing the desired the user communicated through patterns.

DEDICATION

I would like to dedicate my Undergraduate Research Scholar's thesis to my family. My parents, Bert and Lisa Holmstrom, and my sister, Alison Holmstrom, have shown me immeasurable support throughout my undergraduate experience.

ACKNOWLEDGMENTS

First, I would like to thank my Lord and Savior, Jesus Christ. He is the one and only true God who died so that I could be saved and live with Him forever. I would also like to thank my fiancé, Adam Johnson, for encouraging me. I owe a lot to my family including my parents, Bert and Lisa Holmstrom, and my sister, Alison Holmstrom. They have provided me with lots of support for my entire life.

I would like to acknowledge my advisor, Dr. John Hurtado. He has provided me with lots of support and technical guidance. I also received help from James Doebbler, Lesley Weitz, and Roy Palacios. Without their help, I would not have been able to complete the projects. Finally, I would like to thank the Space Engineering Institute's Robotics Team. The Robotics Team gave me the necessary experience to become the student I am today.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

There are many dangerous, dull, and dirty tasks that humans perform every day. In order to minimize human risk, robots can be designed to complete these tasks. NASA's vision for space exploration states that man will return to the moon by 2020 with the goal of living on the moon for extended periods of time [1]. Robots could be sent before the manned missions arrive to build a planetary habitat. Some solutions to complex problems, such as habitat construction, can require a team of robots. Considering this topic, two possible areas of study are: how do robots operate together to complete a task and how does a human communicate with a subset of a robot team. The problems solved in this thesis are solutions to problems that might occur while designing robots for planetary habitation construction. For the transportation of large construction objects, teams of simple robots could be used rather than a single complex robot. If an astronaut was conducting an experiment, a message could be sent to the robot to perform a desired task that would help maximize the limited time on a planetary surface.

Cooperative robotic transportation

In order to investigate the topic of robots operating together to complete a task, this project considers the cooperative control of two robots manipulating flexible payloads.

This thesis follows the style of *Journal of Guidance, Control, and Dynamics*.

This problem was explored by the Space Engineering Institute's Robotics Space Colonization Team [2]. The Robotics Space Colonization Team derived the equations of motion and developed the control laws used to govern the robots. The robotic platform is the iRobot Create® which is similar to the iRobot Roomba® vacuuming robot [3]. The robots are controlled by a central computer which sends commands to the robots, and the robots send back sensor information. Inertial state information is measured by an overhead camera, and local state information is measured by wheel encoders on the robots. The focus of this project is to improve and generalize the math modeling currently being used. This will increase the application to multiple situations. Control methodologies have been developed such that classic control methods used in structural mechanics problems are applied to the cooperative movement of two robots. For instance, the robots could be modeled as masses connected by a spring. The solution to this control problem is widely known, but the application of it to two robots transporting a flexible construction object has never been done.

Human-robot communication

In order for humans and robots to work together, there has to be human-robot communication. The three ways humans and robots can communicate are through voice, vision, and remote control. This project investigates the communication between humans and robots using vision techniques. The human communicates to the robot through a webcam that is connected to the central computer. This webcam uses pattern recognition software. As specified in the central computer, different patterns represent different tasks for the robot to perform. The user holds up a pattern in front of the webcam, and the message is translated to the robot through the central computer.

Another important piece of information that can be communicated is the current battery charge and capacity of the robot. After the user commands a task to the robot through the use of a pattern, the robot measures its battery charge and capacity. If the levels are above a desired amount, the robot completes the task it was assigned. If the levels are low, the robot returns to a home base location for recharging. If the levels are so low that the robot cannot drive to the home base, the user is notified by a message that appears on the screen of the central computer.

CHAPTER II

METHODS

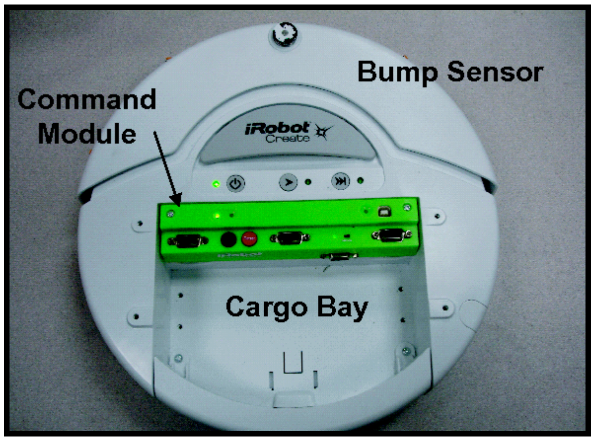
In order to produce hardware demonstrations of human-robot communication and cooperative robotic transportation, several tasks had to be accomplished. Some of these tasks include deriving equations, simulating motion, and developing software algorithms. This chapter will discuss the system components, the techniques of the human-robot communication, and the theory and simulation results of the cooperative robotic transportation.

System components

The autonomous robotic system is made up of a robotic platform, an overhead camera, a wireless communication network, a Kalman filter, and the Central PC. Each part plays an integral role in the success of the system. The system is setup to test autonomous control algorithms for up to three robots. This section will discuss the roles of each component of the system.

Robotic platform

The robotic platform for the project is the iRobot Create®. This robot is equipped with embedded wheel encoders and a command module. The command module includes an 8-bit microcontroller that allows for quick and robust programming in C/C++. It is a differentially driven robot, therefore permitting 360 ° rotational movements. Figure 1 below shows the iRobot Create® and the locations of the sensors.



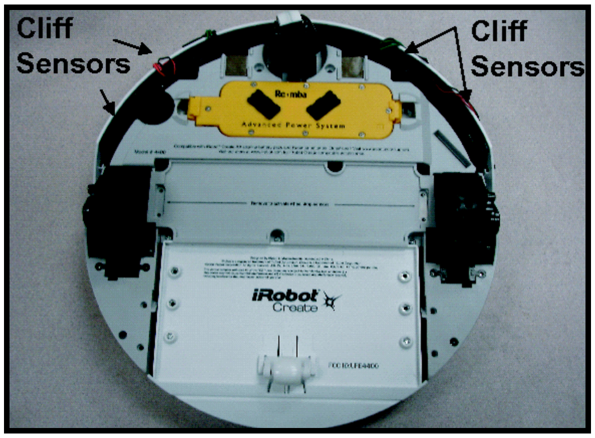


Fig. 1 iRobot Create® Component Locations.

A mechanical-claw assembly was originally designed for the iRobot Create® by the Space Engineering Institute's Robotic Space Colonization team. The unique design incorporates two motions, grasping and lifting, with one motor. The claw is attached to the motor by a steel cable and pulley system.

To lift an object, the motor turns the cable, which pulls the two claw arms towards each other. The arms continue to squeeze together until the tension is more than the gravitational force. Once this happens, the arms begin to lift the object by way of a hinge mounted on the robot. To drop an object, the process is reversed. Limit switches were incorporated to know when the claw was at the maximum or minimum position. Figure 2 shows the claw assembly.

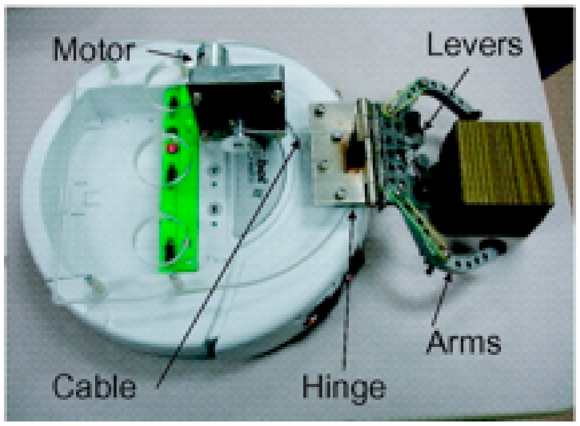
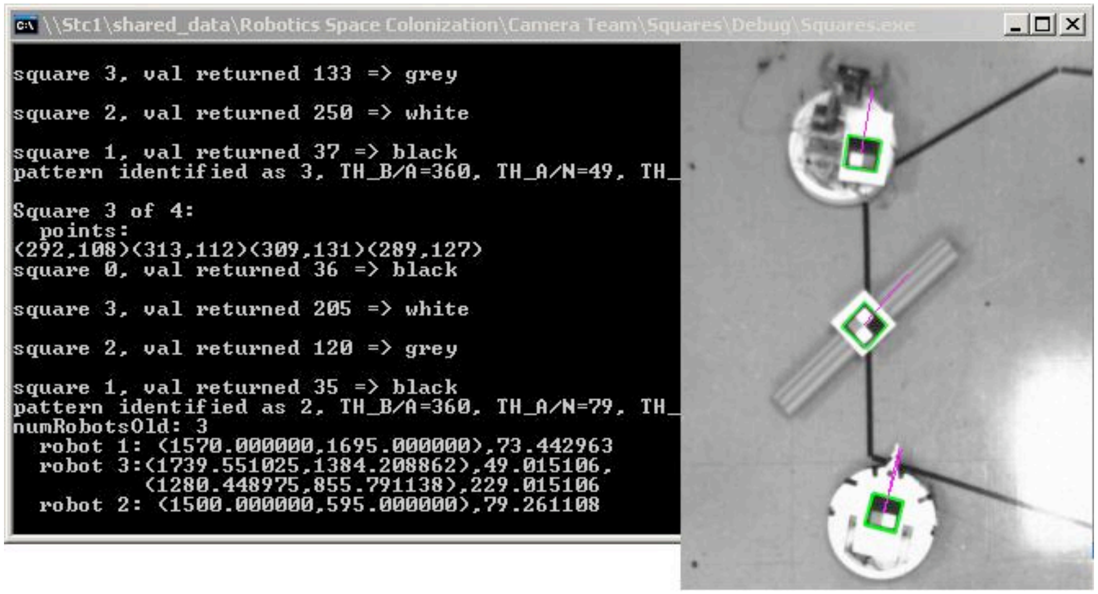


Fig. 2 Robotic Claw Assembly.

Overhead camera

The overhead camera system in the lab is used for inertial state measurement. A MDCS2 monochrome IEEE1394 camera is mounted above the lab and connected to the image recognition computer. Software that was developed by Texas A&M University graduate students James Doebbler and Kevin Daugherty is used to capture images and process them to find patterns. Each pattern is a square cut into four pieces with a different color on the gray scale in each corner. First, the program searches for square shapes with some tolerance. Then, the area of the shape is calculated and the program filters through to find shapes that are within a specified range. The program determines which color each section of the pattern is and compares that to known patterns in the system. Once the pattern is recognized, the position and orientation is determined using a geometric relationship based on how high the camera is mounted. The software outputs the *x* and *y* positions and *θ* orientation of each pattern recognized. This information is sent to the Central PC through UDP communication. The output information is shown in Fig. 3.


\\Stc1\shared_data\Robotics Space Colonization\Camera Team\Squares\Debug\Squares.exe

```
square 3, val returned 133 => grey

square 2, val returned 250 => white

square 1, val returned 37 => black
pattern identified as 3, TH_B/A=360, TH_A/N=49, TH_

Square 3 of 4:
  points:
(292,108)(313,112)(309,131)(289,127)
square 0, val returned 36 => black

square 3, val returned 205 => white

square 2, val returned 120 => grey

square 1, val returned 35 => black
pattern identified as 2, TH_B/A=360, TH_A/N=79, TH_
numRobotsOld: 3
  robot 1: (1570.000000,1695.000000),73.442963
  robot 3:(1739.551025,1384.208862),49.015106,
          (1280.448975,855.791138),229.015106
  robot 2: (1500.000000,595.000000),79.261108
```

Fig. 3 Overhead Camera Output.

The overhead camera system is a preferred option for a laboratory setting; however there will not be an overhead camera on the moon or Mars. This system could be replaced by a set of global positioning satellites that orbit the moon or Mars. This would provide the necessary inertial state information to the robots and astronauts. Although the global positioning satellites system would be expensive, the benefits would outweigh the costs for extended stay on a planetary surface.

Wireless communication

The robots and the central computer communicate with each other through a Zigbee communication network. Each robot is equipped with a Zigbee module that allows it to send and receive data packets. The Zigbee module uses one of the ports on the command module for serial input and output. MaxStream XBee chips and development boards were selected due to small power consumption and ease of use.

Although the Zigbee module communicates slower than a Bluetooth system, a continuous stream of data will not be sent and therefore the Zigbee is sufficient. Testing proved that the maximum reliable communication rate is 25 Hz. The maximum necessary is 15 Hz, which is the speed of packets sent from the image recognition PC; therefore the speed of the Zigbee modules is more than enough for the system.

Kalman filter

A Kalman filter is an algorithm that determines the best state estimates of a system using both the measurements and the model. This algorithm is necessary when complex tasks are being performed by a system. The problem with taking information directly from the sensors is the corruption of noise. When noise interferes with the system, the robots may think they are in one place when in actuality, they are in another place.

The Kalman filter implemented in the software of the system includes an initialization, propagation, and update phase. The propagation phase takes the last known position of the robot and extrapolates it using the robot model and the velocity commands. This is compared to the current measurement by a relationship based on a value called the covariance. The covariance determines the accuracy of the state estimate. The new estimate is retained for the next propagation phase of the algorithm. Figure 4 is a flow chart of the Kalman filter process.

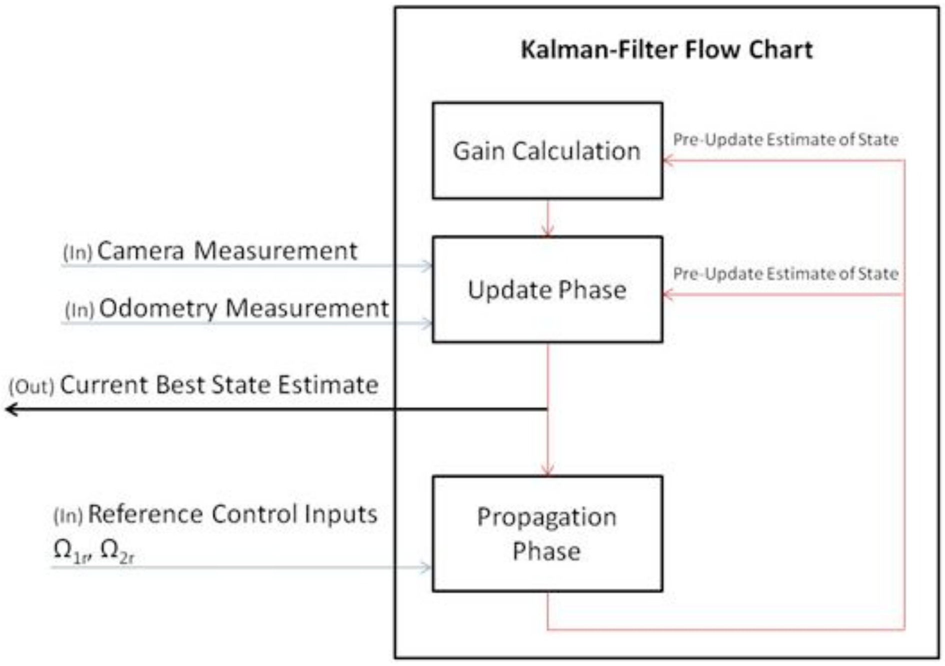

Kalman-Filter Flow Chart
Gain Calculation
Pre-Update Estimate of State
(In) Camera Measurement
Update Phase
Pre-Update Estimate of State
(In) Odometry Measurement
(Out) Current Best State Estimate
(In) Reference Control Inputs
Ω1r, Ω2r
Propagation Phase

Fig. 4 Kalman Filter Flow Chart.

System architecture

The Central PC is the brains behind the centralized system. All of the packets from the image-processing computer and from the robots are processed in the Central PC. The Central PC is divided into different classes that contain data and functions. Sensor information and state estimation are manipulated to determine the velocity commands that are sent to the robot.

Figure 5 shows the relationships between the classes in the Central PC. The Zigbee Class handles the communication between the Central PC and the robot. The Robot Class contains all of the information about the robot and receives velocity commands from the controls algorithms. The Measurement Class contains information from the sensor measurements and sends the information to the controls algorithms. The UDP Communication Class communicates with the Image Recognition PC and receives and sends camera data. All of the classes work together in order to complete the desired task.

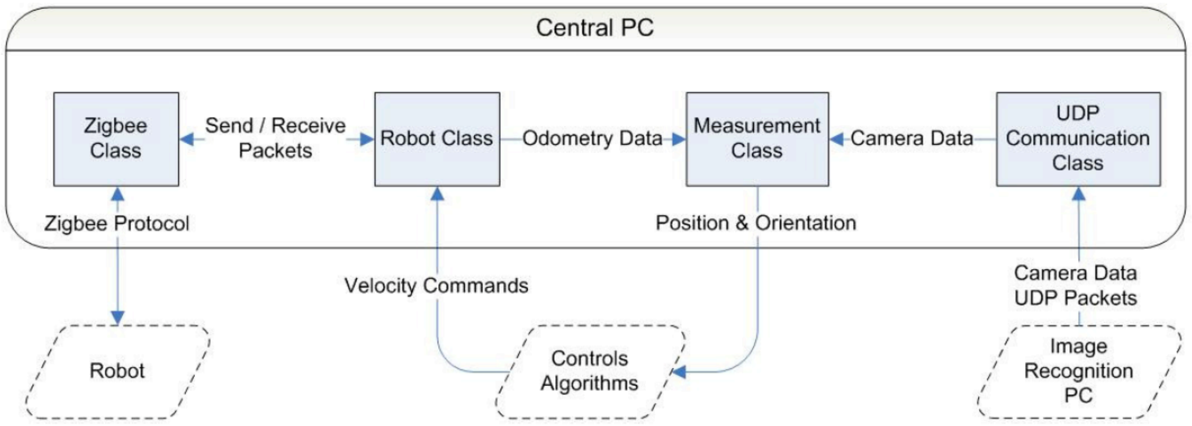


Fig. 5 System Architecture Relationship Chart.

Cooperative robotic transportation developments

In order to produce a hardware demonstration of cooperative robotic transportation, several theoretical developments had to be accomplished. This section details the theoretical developments including the model of the robot, the model of the flexible structure, and the trajectory tracking control law. A flow chart is presented that shows the steps to completing a hardware demonstration of cooperative robotic transportation. This is primarily based on previous work about cooperative robotic transportation of a flexible object [4].

Robot model

In order to derive the equations of motion for the robot, the inertial and body-fixed reference frames are defined. The body reference frame is centered on the robot and the inertial reference frame is at a fixed point in space. Let $\mathbf{r}_c$ be the inertial position vector to the center of the robot, as shown in Fig. 6.

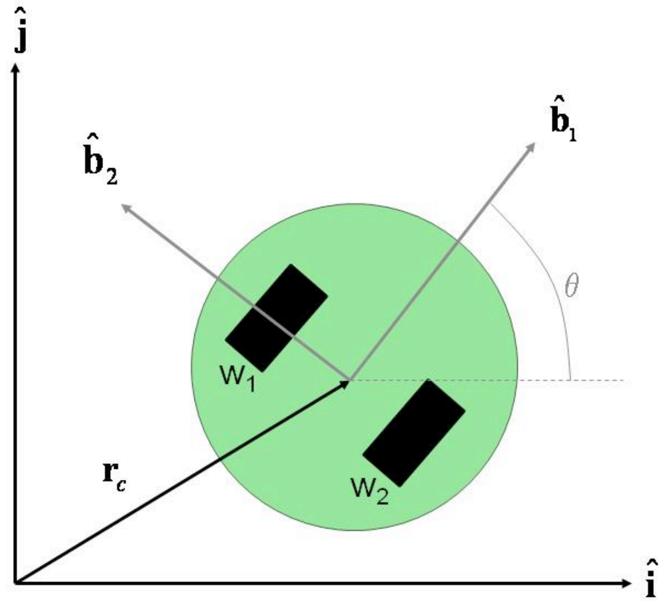


Fig. 6 Inertial and Body Fixed Reference Frames.

From the figure, it can be seen that the robot's inertial position, $\mathbf{r}_c$, and velocity, $\dot{\mathbf{r}}_c$, can be written in the inertial reference frame as shown below.

$$\mathbf{r}_c = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} \tag{1}$$

$$\dot{\mathbf{r}}_c = \dot{x}\hat{\mathbf{i}} + \dot{y}\hat{\mathbf{j}} \tag{2}$$

Equations can be written for the positions of each robot wheel.

$$\mathbf{r}_{w_1} = \mathbf{r}_c + d\hat{\mathbf{b}}_2 \tag{3}$$

$$\mathbf{r}_{w_2} = \mathbf{r}_c - d\hat{\mathbf{b}}_2 \tag{4}$$

The scalar quantity *d* is the distance from the center of the robot to the center of each robot wheel. The time derivative of the wheel positions are derived using the transport theorem, which is used to calculate inertial derivatives for vectors described in non-inertial reference frames [5].

$$\frac{{}^{I}d}{dt}\left(\mathbf{r}_{w_1}\right) = \frac{{}^{B}d}{dt}\left(\mathbf{r}_{w_1}\right) + \omega_{\frac{B}{I}} \times {}^{B}\mathbf{r}_{w_1} \tag{5}$$

$$\dot{\mathbf{r}}_{w_1} = \dot{\mathbf{r}}_c - d\dot{\theta}\hat{\mathbf{b}}_1 = \dot{x}\hat{\mathbf{i}} + \dot{y}\hat{\mathbf{j}} - d\theta\hat{\mathbf{b}}_1 \tag{6}$$

$$\dot{\mathbf{r}}_{w_2} = \dot{\mathbf{r}}_c + d\dot{\theta}\hat{\mathbf{b}}_1 = \dot{x}\hat{\mathbf{i}} + \dot{y}\hat{\mathbf{j}} + d\theta\hat{\mathbf{b}}_1 \tag{7}$$

Writing the wheel velocity vectors in the body-fixed reference frame produces the following two equations.

$$\dot{\mathbf{r}}_{w_1} = \left(\dot{x}\cos\theta + \dot{y}\sin\theta - d\dot{\theta}\right)\hat{\mathbf{b}}_1 + \left(-\dot{x}\sin\theta + \dot{y}\cos\theta\right)\hat{\mathbf{b}}_2 \tag{8}$$

$$\dot{\mathbf{r}}_{w_2} = \left(\dot{x}\cos\theta + \dot{y}\sin\theta + d\dot{\theta}\right)\hat{\mathbf{b}}_1 + \left(-\dot{x}\sin\theta + \dot{y}\cos\theta\right)\hat{\mathbf{b}}_2 \tag{9}$$

Assuming a no-slip wheel condition, the wheel velocities are defined.

$$\dot{\mathbf{r}}_{w_1} = v_L\hat{\mathbf{b}}_1 \tag{10}$$

$$\dot{\mathbf{r}}_{w_2} = v_R\hat{\mathbf{b}}_1 \tag{11}$$

Setting the equations for $\dot{\mathbf{r}}_{w_1}$ and $\dot{\mathbf{r}}_{w_2}$ equal to each other produces three unique equations. The first equation is the nonholonomic constraint associated with zero velocity in the $\hat{\mathbf{b}}_2$ direction. The second and third equations are associated with the wheel velocities in the $\hat{\mathbf{b}}_1$ direction. The matrix form of the three equations is given below.

$$\begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \cos\theta & \sin\theta & -d \\ \cos\theta & \sin\theta & d \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} 0 \\ v_L \\ v_R \end{Bmatrix} \tag{12}$$

Computing the inverse gives the equations as a function of left and right wheel velocities.

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} -\sin\theta & \dfrac{\cos\theta}{2} & \dfrac{\cos\theta}{2} \\ \cos\theta & \dfrac{\sin\theta}{2} & \dfrac{\sin\theta}{2} \\ 0 & -\dfrac{1}{2d} & \dfrac{1}{2d} \end{bmatrix} \begin{Bmatrix} 0 \\ v_L \\ v_R \end{Bmatrix} \tag{13}$$

The following relationships transform the robot equations into the commonly-used form in the literature on nonholonomically-constrained vehicle motion. The kinematic equations are shown below.

$$v = \frac{v_L + v_R}{2}; \quad \omega = \frac{v_R - v_L}{2d} \tag{14}$$

$$\dot{x} = v\cos\theta; \quad \dot{y} = v\sin\theta; \quad \dot{\theta} = \omega \tag{15}$$

If the control inputs are the motor force and torque, two additional equations are necessary that show the relationship of the dynamics. In order to produce these equations, a free body diagram is defined below in Fig. 7.

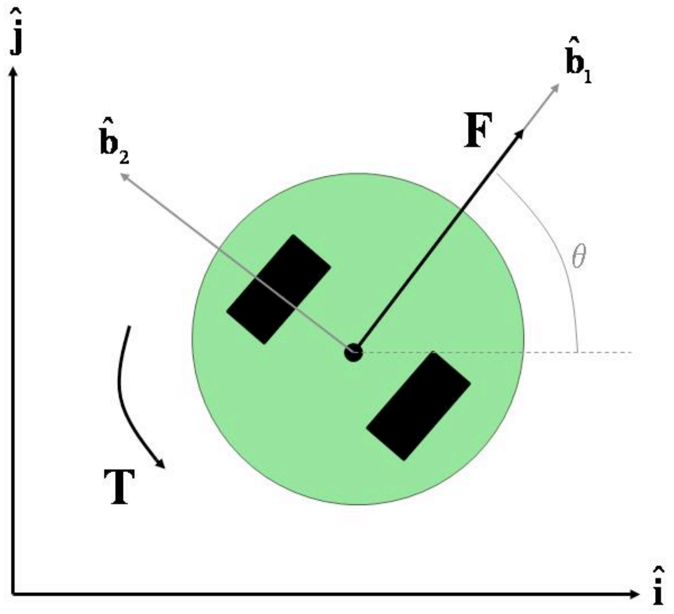


Fig. 7 Free Body Diagram of Robot.

Summing the forces and torques produces the following two equations, where F is the force of the motor and T is the torque of the motor.

$$m\dot{v} = F\ ; \quad I\dot{\omega} = T \tag{16}$$

These equations combined with the previous three equations produce a kinematic and dynamic model of the one-robot system. This model is necessary for the development of the trajectory tracking control law for one robot.

Flexible structure model

When considering prescribed paths and controlling a two-robot-construction-object formation, a kinematic and dynamic model of the entire system is needed. Figure 8 shows the two-robot team grasping the construction object, where ρ is the distance between the center of the robot and the grasping point of the robot's claw. This distance is assumed to stay constant throughout the motion.

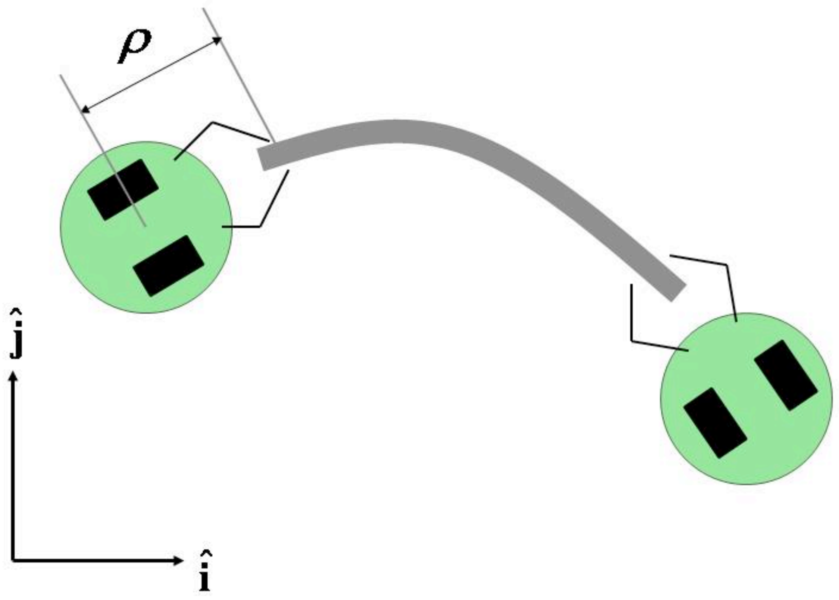


Fig. 8 Two-Robot Team Grasping the Flexible Structure.

The flexible structure is modeled as a translational spring with two torsional springs at each end of the object. The springs are used to represent the force and torque that is put on the robot from the structure. Figure 9 shows the placement of the springs and the position variables that relate the two robots. The translational spring with length r represents axial deformation and the bending resistance of the flexible structure.

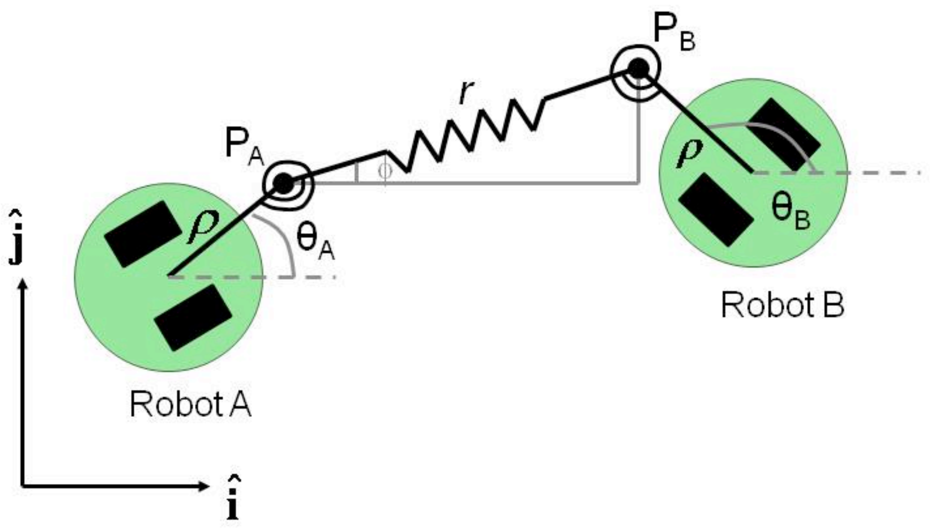


Fig. 9 Spring Model of the Flexible Structure.

The distance r and the angle ϕ are the variables that relate the position and orientation of the robots. The equations that define these variables in terms of the robot coordinates are shown below.

$$r = \left[\left(x_B + \rho\cos\theta_B - x_A - \rho\cos\theta_A\right)^2 + \left(y_B + \rho\sin\theta_B - y_A - \rho\sin\theta_A\right)^2\right]^{\frac{1}{2}} \tag{17}$$

$$\phi = \tan^{-1}\left(\frac{y_B + \rho\sin\theta_B - y_A - \rho\sin\theta_A}{x_B + \rho\cos\theta_B - x_A - \rho\cos\theta_A}\right) \tag{18}$$

When the flexible structure is modeled as a set of springs, a new free body diagram is necessary to impose the spring forces on the robots. Once the free body diagram is defined, new dynamic equations can be derived that incorporate the entire system of two robots and a flexible structure. Figure 10 shows the new free body diagram.

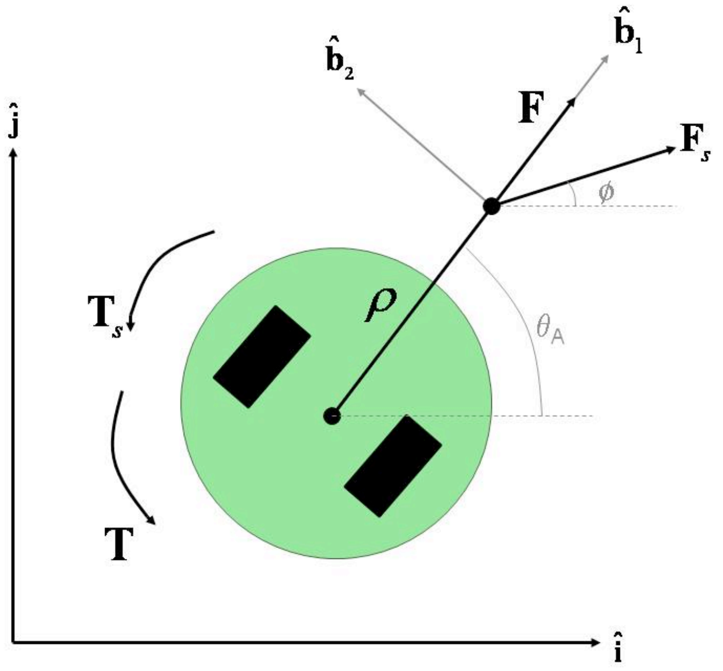


Fig. 10 Free Body Diagram with Spring Forces and Torques.

The equations for the new force $\mathbf{F_s}$ and the new torque $\mathbf{T_s}$ are defined below for Robot A.

$$\mathbf{F}_s = k_1 \left(r - r_0 \right) \tag{19}$$

$$\mathbf{T}_s = k_2 \left(\phi - \theta_A \right) \tag{20}$$

Before the forces and torques are summed, the equations have to be projected onto the $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ axes respectively. This gives the equations for the entire system. Below are the equations governing Robot A and Robot B.

$$\dot{x}_A = v_A \cos\theta_A$$
$$\dot{y}_A = v_A \sin\theta_A$$
$$\dot{\theta}_A = \omega_A \tag{21}$$
$$m\dot{v}_A = F_A + k_1(r - r_0)\cos(\phi - \theta_A)$$
$$I\dot{\omega}_A = T_A + k_2(\phi - \theta_A) + \rho k_1(r - r_0)\sin(\phi - \theta_A)$$

$$\dot{x}_B = v_B \cos\theta_B$$
$$\dot{y}_B = v_B \sin\theta_B$$
$$\dot{\theta}_B = \omega_B \tag{22}$$
$$m\dot{v}_B = F_B + k_1(r - r_0)\cos(\phi - \theta_B + \pi)$$
$$I\dot{\omega}_B = T_B + k_2(\phi - \theta_B + \pi) + \rho k_1(r - r_0)\sin(\phi - \theta_B + \pi)$$

This model assumes that the flexible structure is massless. This is a reasonable assumption because all of the forces and torques that are acting on the robot are accounted for. The equations are coupled together through the variables of *r* and the angle ϕ.

Optimal trajectory design and trajectory tracking control law

A trajectory design for a team of robots cooperatively transporting an object can include relative trajectories for each robot. It is necessary for the trajectories to be functions of time so that the relative-motion constraints between the robots are met. Because of the complexity of the robotic platform model and path constraints, a direct trajectory optimization problem was transformed into a nonlinear programming problem. The trajectory design developed by Texas A&M University graduate student Lesley Weitz allows for restrictions to be placed on the object being transported such as bending and stretching limits.

The trajectory tracking control law is developed for the fifth-order equations of motion shown for each robot in equations (21) and (22). First, position errors are defined in the x and y directions.

$$e_x(t) = x(t) - x_r(t); \quad e_y(t) = y(t) - y_r(t) \tag{23}$$

Error dynamics are defined to be equal to some gain times the position error to drive the errors to zero.

$$\dot{e}_x(t) = -k_x e_x(t); \quad \dot{e}_y(t) = -k_y e_y(t) \tag{24}$$

From equations (23) and (24), commanded velocities in the x and y directions along with the magnitude of the velocity can be found.

$$\dot{x}_c(t) = -k_x e_x(t) + \dot{x}_r(t); \qquad \dot{y}_c(t) = -k_y e_y(t) + \dot{y}_r(t); \qquad v_c = \sqrt{\dot{x}_c^2 + \dot{y}_c^2} \tag{25}$$

The commanded orientation equation is in terms of the commanded velocity components. This equation is directly related to the nonholonomic constraint equation.

$$\theta_c = \tan^{-1}\left(\frac{\dot{y}_c}{\dot{x}_c}\right) \tag{26}$$

The error equations for the heading angle and the commanded angular velocity are defined below.

$$e_\theta(t) = \theta(t) - \theta_c(t); \quad \omega_c(t) = -k_\theta e_\theta(t) \tag{27}$$

The commanded velocity is aligned with the commanded heading angle. Therefore, the velocity has to be projected onto the $\hat{\mathbf{b}}_1$ axis.

$$v_{c_p} = v_c \cos(\theta - \theta_c) = v_c \cos(e_\theta) \tag{28}$$

Velocity and angular velocity errors are defined below, along with the selected error dynamics to drive the errors to zero.

$$e_v = v - v_{c_p}; \quad \dot{e}_v = \dot{v} - \dot{v}_{c_p} = -k_v e_v \tag{29}$$

$$e_\omega = \omega - \omega_c; \quad \dot{e}_\omega = \dot{\omega} - \dot{\omega}_c = -k_\omega e_\omega \tag{30}$$

Dynamic inversion is used to find the control inputs after substituting $\dot{v}$ and $\dot{\omega}$ into the dynamics equations.

$$F = m\left(\dot{v}_{c_p} - k_v e_v\right) - f(r,\phi,\theta) \tag{31}$$

$$T = I(\dot{\omega}_c - k_\omega e_\omega) - g(r,\phi,\theta) \tag{32}$$

The functions *f* and *g* are the nonlinear terms from the dynamics equations shown previously. The equations for $\dot{v}_{c_p}$ and $\dot{\omega}_c$ are determined by taking derivatives of v_{c_p} and ω_c.

$$\dot{v}_{c_p} = \frac{\dot{x}_c \ddot{x}_c + \dot{y}_c \ddot{y}_c}{\sqrt{\dot{x}_c^2 + \dot{y}_c^2}} \cos(\theta - \theta_c) - v_c(\omega - \omega_c)\sin(\theta - \theta_c); \quad \dot{\omega}_c = -k_\theta(\omega - \omega_c) \tag{33}$$

System flow chart

In order to produce a hardware demonstration of cooperative robotic transportation, a flow chart had to be made to plan a way to integrate the system components. Although the system components work individually, getting them to work as a unit can be very challenging. Figure 11 is the flow chart of the system process for the hardware demonstration of cooperative robotic transportation.

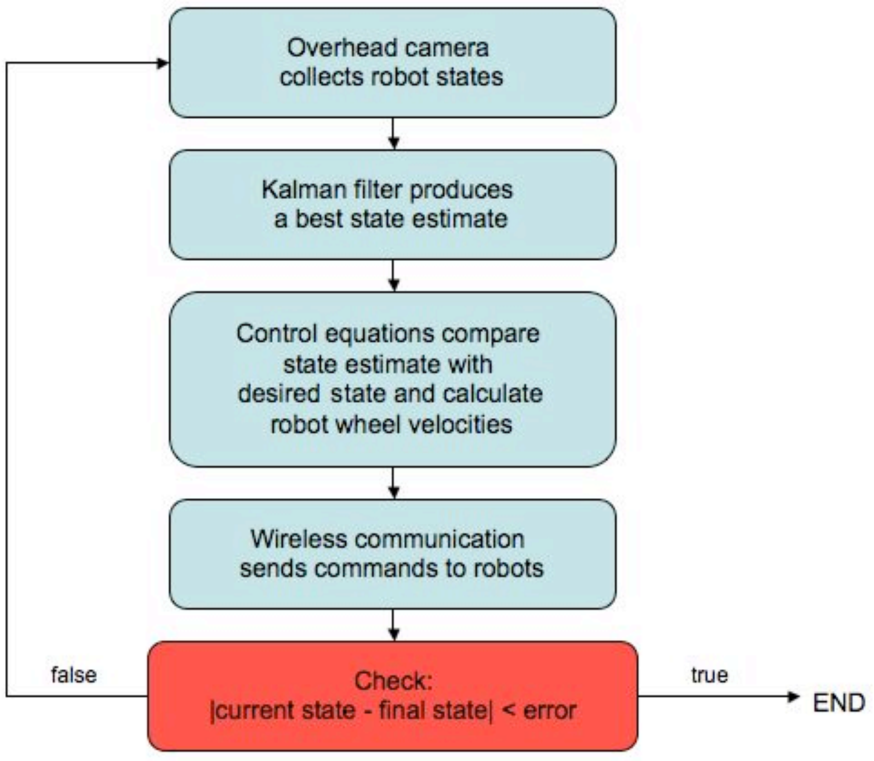

Overhead camera collects robot states
Kalman filter produces a best state estimate
Control equations compare state estimate with desired state and calculate robot wheel velocities
Wireless communication sends commands to robots
Check: |current state - final state| < error
false
true
END

Fig. 11 System Process Flow Chart for Cooperative Robotic Transportation.

First, the overhead camera collects data and measures the states of both robots. This information is sent to the Kalman filter which combines the sensor information with information about the robot model to produce a best state estimate. The state estimate is sent to the control equations for comparison with the desired state, which comes from the reference trajectories. Robot wheel velocities are calculated and the wireless communication network sends the commands to the robots. A check is then calculated by taking the absolute value of the difference between the current state and the desired final state of the reference trajectory path. This difference is compared to a specified error value to give the robot some final footprint to end up in. If the check is true, then the algorithm exits and the robots are finished transporting the object. If the check is false, the algorithm repeats until the robot is in the desired final footprint.

Human-robot communication subsystems

This section discusses the subsystems needed to produce a hardware demonstration of human-robot communication. Path planning and control equation developments are presented that are implemented on the robot. A flow chart shows the process the system goes through during a hardware demonstration of the human-robot communication demonstration.

Webcam

The robot's vision system consists of a Logitech QuickCam® Communicate Deluxe. This webcam connects to the Central PC through a USB connection. The image output from the webcam has 1280 x 1024 pixels at 30 frames per second in color. These specifications make the webcam qualify for the purposes of this project.

The job of the webcam is to scan for patterns that the user holds up in front of it. These patterns represent different tasks for the robot to perform. This permits a way for the user and the robot to communicate. The software used to recognize the pattern is only a slight variation to the software the overhead camera uses to recognize patterns. The squares the software searches for are much larger because the user holds the pattern directly in front of the webcam. Also, the square is identified based on the grayscale block pattern. Once the square is identified, the software stops searching and sends a message containing which pattern was recognized to the main program.

Battery information

Gathering health information from the robot has many benefits. Battery information falls under the robot health umbrella. For this project, battery information is gathered before the robot completes the desired task to make sure that the task can be accomplished. This directly applies to a situation on the moon or Mars where it is important to know the battery information of the robots the astronauts are working with.

In order to gather battery information, software is implemented on the robot that utilizes two functions that are built in to the iRobot Create®. The first function collects the battery charge in milliamp-hours (mAh). The second function collects the battery capacity in mAh. These two values are compared as a percentage and that value is evaluated against a desired battery level. The capacity is also compared to a nominal battery capacity to make sure the user know when a new battery is needed.

Path planning

A simple path-planning algorithm plans a trajectory from an arbitrary initial position of the robot to a desired final destination. The initial position of the robot is measured from the overhead camera. Cubic-polynomial functions are used to generate a reference trajectory from the initial position to the final desired position. The initial and final velocities are assumed to be zero. The polynomial functions are time-based trajectories, which makes the total-trajectory time one of the design variables. The equations below are the cubic-polynomial functions for the *x* and *y* trajectories.

$$\begin{aligned} x(t) &= a_1 t^3 + a_2 t^2 + a_3 t + a_4 \\ y(t) &= b_1 t^3 + b_2 t^2 + b_3 t + b_4 \end{aligned} \tag{34}$$

The following matrix equation is solved to determine the “a” coefficients for the *x* trajectory.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ T^3 & T^2 & T & 1 \\ 3T^2 & 2T & 1 & 0 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{Bmatrix} \tag{35}$$

Here, $x(0) = x_0$, $\dot{x}(0) = \dot{x}_0 = 0$, $x(T) = x_f$, $\dot{x}(T) = \dot{x}_f = 0$, and *T* is the desired time to reach the final position. The coefficients of the *y* trajectory are found in the same way.

Trajectory tracking control law

The Trajectory Tracking Control Law is necessary to drive the position errors of the robot to zero. Using error equations and proportional gains, commanded velocities are calculated that keep the robot on the desired path. The following development derives these equations for a single robot.

Error for the robot's position states is defined in the equations below. The estimated states of the robot are received from the Kalman filter, and are denoted as: $\{x, y\}$. The reference position states are calculated based on the path planning algorithm defined previously, and are symbolized as: $\{x^*, y^*\}$.

$$e_1 = x - x^* \tag{36}$$

$$e_2 = y - y^* \tag{37}$$

Looking at the x-direction state, the derivative of the error is the difference in velocities of the actual and reference states.

$$\dot{e}_1 = \dot{x} - \dot{x}^* \tag{38}$$

Because the desired limit of the error is zero, the derivative of the error is also defined as

$$\dot{e}_1 = -k_1 e_1 \tag{39}$$

This utilizes a proportional controller, k_1. Setting these equations equal to each other, and solving for the actual robot velocity yields

$$\dot{x} = \dot{x}^* - k_1 e_1 \tag{40}$$

This velocity is now defined as the commanded velocity, $\dot{x}_c$. The commanded velocity is the velocity needed for the robot to return to the desired path.

$$\dot{x}_c = \dot{x}^* - k_1 e_1 \tag{41}$$

Similarly, the y-direction commanded velocity equation is as follows.

$$\dot{y}_c = \dot{y}^* - k_2 e_2 \tag{42}$$

Error in the orientation is defined as the difference between the actual orientation and the commanded orientation.

$$e_3 = \theta - \theta_c \tag{43}$$

In order to calculate the commanded orientation, the following equation is used.

$$\theta_c = \tan^{-1}\left(\frac{-\dot{x}_c}{\dot{y}_c}\right) \tag{44}$$

This equation is derived in the equations of motion for the robot. Here, we define the time-rate of change of the commanded heading as shown below.

$$\dot{\theta}_c = \omega_c = -k_3 e_3(t) \tag{45}$$

The desired forward velocity can be found from the commanded velocities in the *x* and *y* directions: $v_d = \sqrt{\dot{x}_c^2 + \dot{y}_c^2}$. This velocity is projected onto the $\hat{\mathbf{b}}_1$ axes through the actual heading angle in order to find the "permissible" commanded velocities.

$$\dot{x}_p = v_d \cos\theta; \quad \dot{y}_p = v_d \sin\theta \tag{46}$$

The individual wheel velocities to track the trajectory can be found using the inverse of the kinematics relationship presented previously.

$$\begin{Bmatrix} 0 \\ v_L \\ v_R \end{Bmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \cos\theta & \sin\theta & -d \\ \cos\theta & \sin\theta & d \end{bmatrix} \begin{Bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta}_c \end{Bmatrix} \tag{47}$$

Note that the first equation is the nonholonomic constraint, which will always be satisfied.

System flow chart

The system goes through the process shown in Fig. 12 in order to produce a hardware demonstration of human-robot communication. This flow chart integrates all the subsystems so that the system runs autonomously. The logic presented includes decision-making the robot does after checking its battery information.

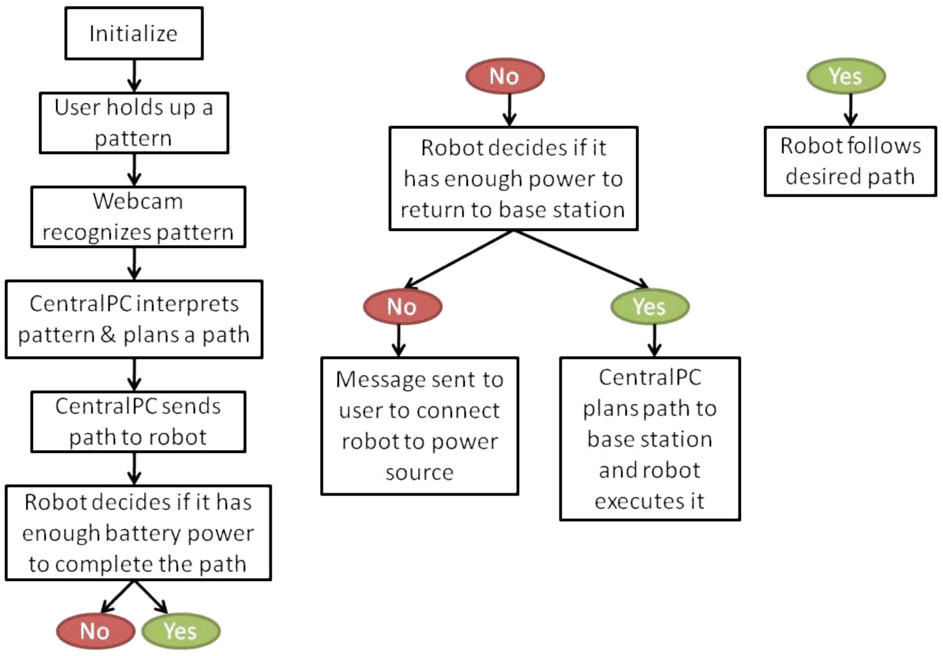

Initialize
User holds up a pattern
Webcam recognizes pattern
CentralPC interprets pattern & plans a path
CentralPC sends path to robot
Robot decides if it has enough battery power to complete the path
No
Yes
No
Robot decides if it has enough power to return to base station
No
Yes
Message sent to user to connect robot to power source
CentralPC plans path to base station and robot executes it
Yes
Robot follows desired path

Fig. 12 Human-Robot Communication Flow Chart.

First the user initializes the software, or in other words the user runs the program. Then the user holds up a pattern in front of the webcam while the webcam is scanning for patterns. Once the webcam recognizes the pattern, the pattern number is sent to the Central PC. The number sent corresponds to a path for the robot to follow. This path represents a task that the robot would accomplish. The Central PC plans the path and sends the path to the robot. The robot measures its battery level against a nominal amount and determines if it can traverse the path. If the robot has enough battery power, then it follows the desired path. If the robot does not have enough battery power, it determines if it can drive itself to the home base, which might include a charging station. If it can drive to the home base, the Central PC plans a path for the robot to the home base and the robot executes the path. If the robot cannot drive to the home base, a message is sent to the user by way of the Central PC that the robot is completely out of batter power and needs to be collected manually.

CHAPTER III

RESULTS

The main goals of the projects have been accomplished over the course of one summer and one fall semester. The goals are to produce hardware demonstrations of cooperative robotic transportation and human-robot communication. In this chapter, simulation and hardware results are presented for the two projects.

Cooperative robotic transportation results

In order to produce hardware results of cooperative robotic transportation, simulation results are necessary to verify the equations derived in Chapter II. The difference between the simulation results and hardware results is some errors in the solution. When working with simulation, there is no noise to create error in your solution unless it is introduced. Therefore, the hardware results are somewhat different than the simulation results because they incorporate real life errors. These errors include sensor noise, measurement noise, communication delay and others. Although the hardware results do not match up perfectly with simulation, they do verify the theory behind the equations which is the main goal. In this section, simulation results and hardware results are presented for the cooperative robotic transportation project.

Simulation results

Simulations are first produced for the perfect case without any errors. The robot paths are in a straight line as shown in Fig. 13.

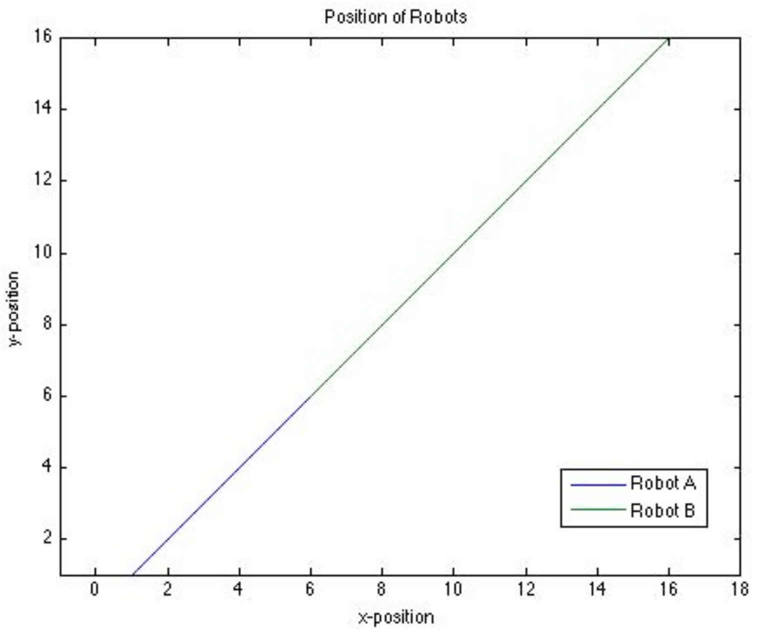


Fig. 13 Simulation Results of Robot Paths.

In order to complete this path, the robots are prescribed reference trajectories based on some desired boundary conditions for position, velocity, and acceleration. The error between the reference path and the actual robot position is calculated and velocity commands are determined based on this error. The robot velocities are shown in Fig. 14.

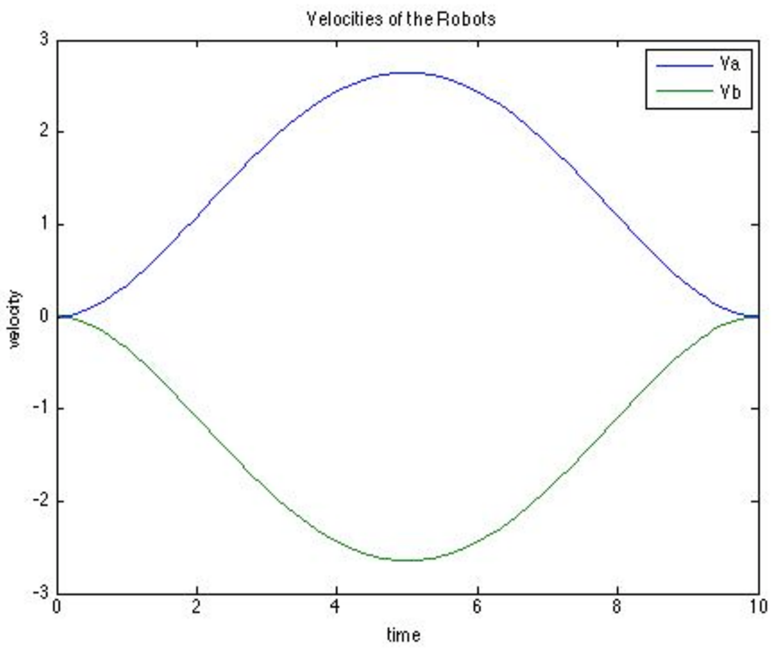


Fig. 14 Simulation Results of Robot Velocities.

When an initial error is introduced in the simulation, the control laws are tested for robustness. In this case, a small initial condition error is introduced in the position of both robots. The same process is followed to calculate the commanded velocity to drive the robots on the desired path. Figure 15 shows the path the robots took with the initial position error, and Fig. 16 shows the position error values for both robots. The important issue to notice in Fig. 16 is that the errors converge quickly for both robots. Figure 17 shows the robot velocities.

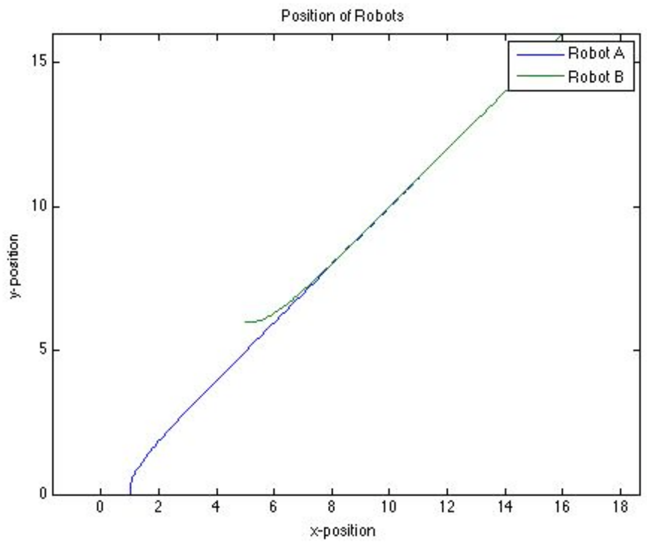


Fig. 15 Simulation Results of Robot Paths with Introduced Error.

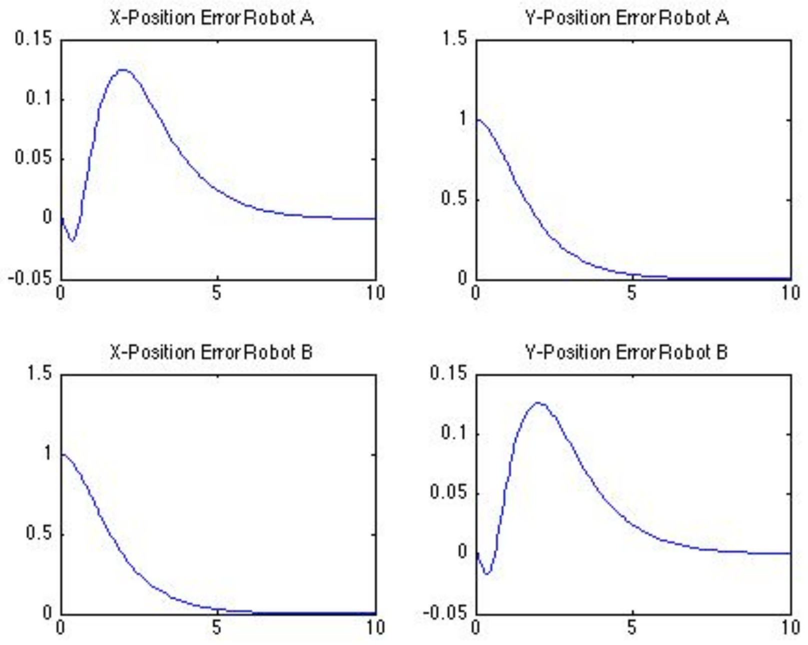


Fig. 16 Simulation Results of Robot Position Errors.

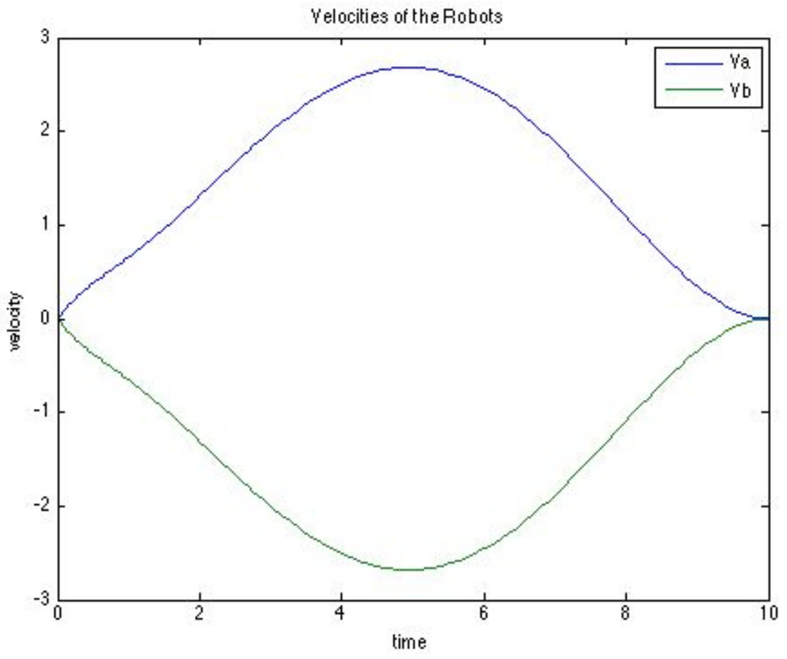


Fig. 17 Simulation Results of Robot Velocities with Introduced Error.

Hardware results

For the hardware implementation, the autonomous robotics system is used as described previously in Chapter II. A simpler form of the equations of motion is used, although the full set of the equations is used in the NLP calculation of the prescribed path. There are larger errors in the results from the hardware implementation that may be from sensor noise, measurement noise, and wireless communication error. The robots start with an initial position error as shown in Fig. 18. The claw assembly on the robot allows for some slipping in the grip of the flexible object. This allows one robot to push the other robot off the path. This creates some error, but overall the robots stay true to their paths. Figure 18 shows the hardware test results of the robots positions. Test results from three different runs are shown in the figure. The black square around the paths is the field of view of the overhead camera.

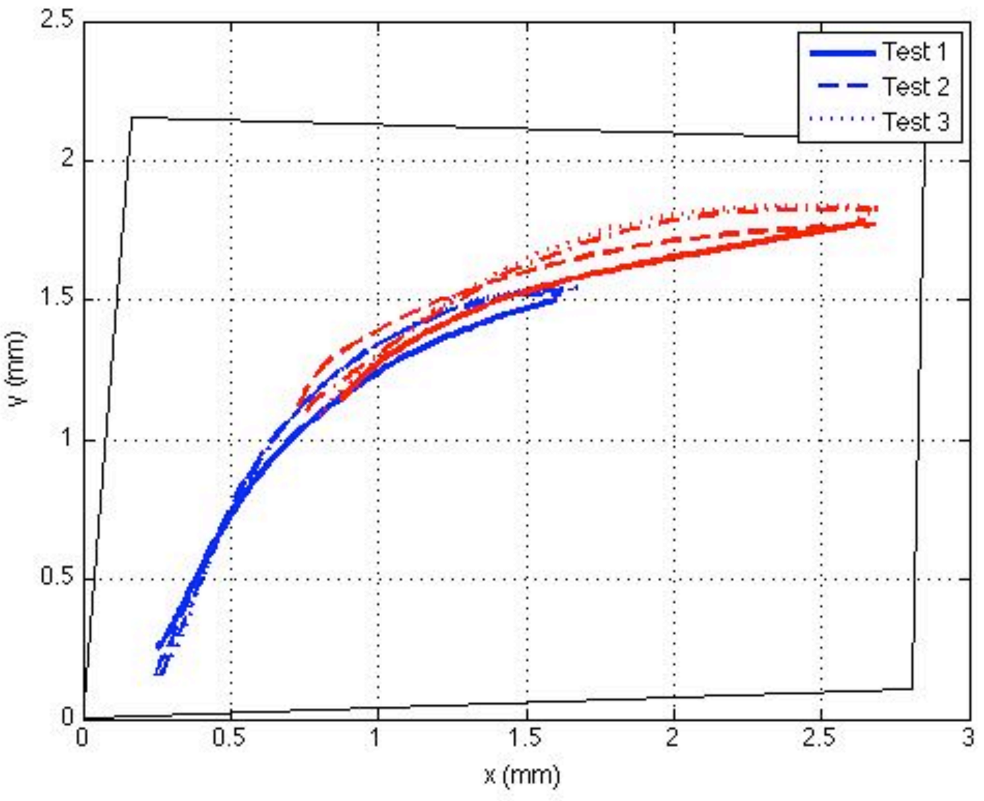


Fig. 18 Hardware Results of Robot Paths.

Figure 19 shows the distance between the robots and the angular deformation for each test run. The time scale changes for some of the runs because of the iRobot's performance at small velocities. The iRobot Create's wheels have trouble when the commanded velocity is less than 20 mm/s. Once the velocity reaches that limit, the wheels begin to stick and the robot does not perform as desired.

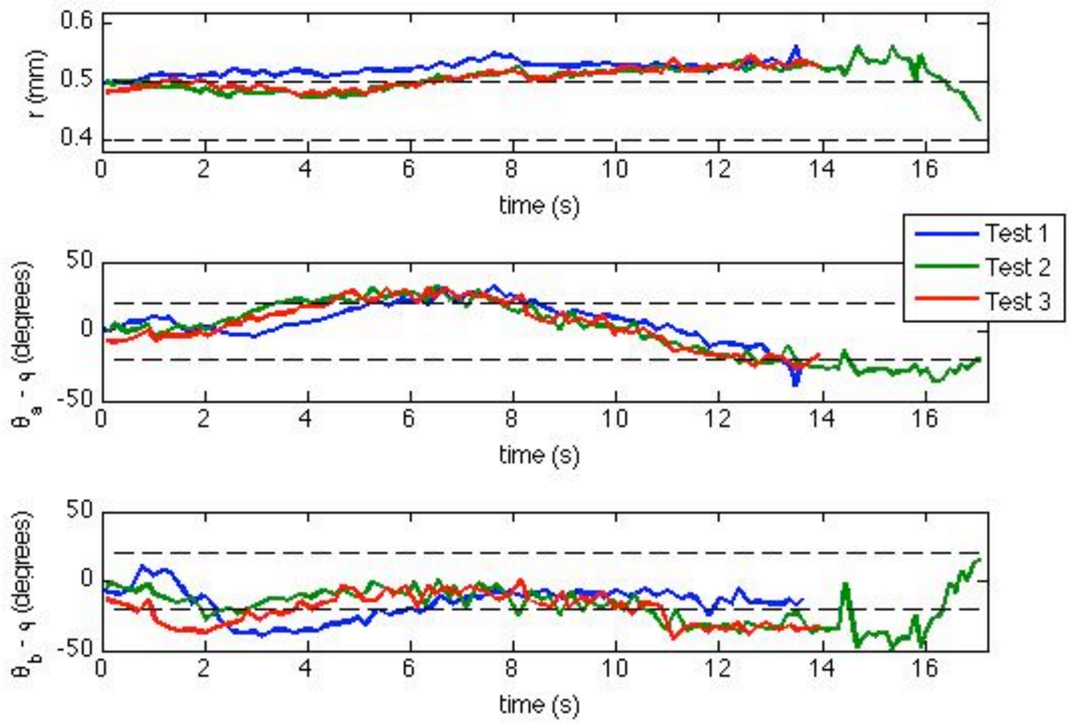


Fig. 19 Hardware Results for Time Histories of Variables.

Human-robot communication results

To demonstrate human-robot communication, in this project a webcam scans for patterns that represent different tasks for the robot to perform. The tasks are represented by destinations for the robot to travel to. When analyzing the results, the important thing to notice is not how accurately the robot followed the desired path, but the fact that the robot traveled to the desired destination rather than another destination.

Figure 20 is an image of the user holding up a pattern in front of the webcam. The pattern represents a final destination for the robot to travel to. Ideally, the webcam would be connected to the robot and the robot would have enough processing power to evaluate the algorithms on its own. The robotic platform used for this project does not have high processing power and therefore all of the processing is done on the Central PC.

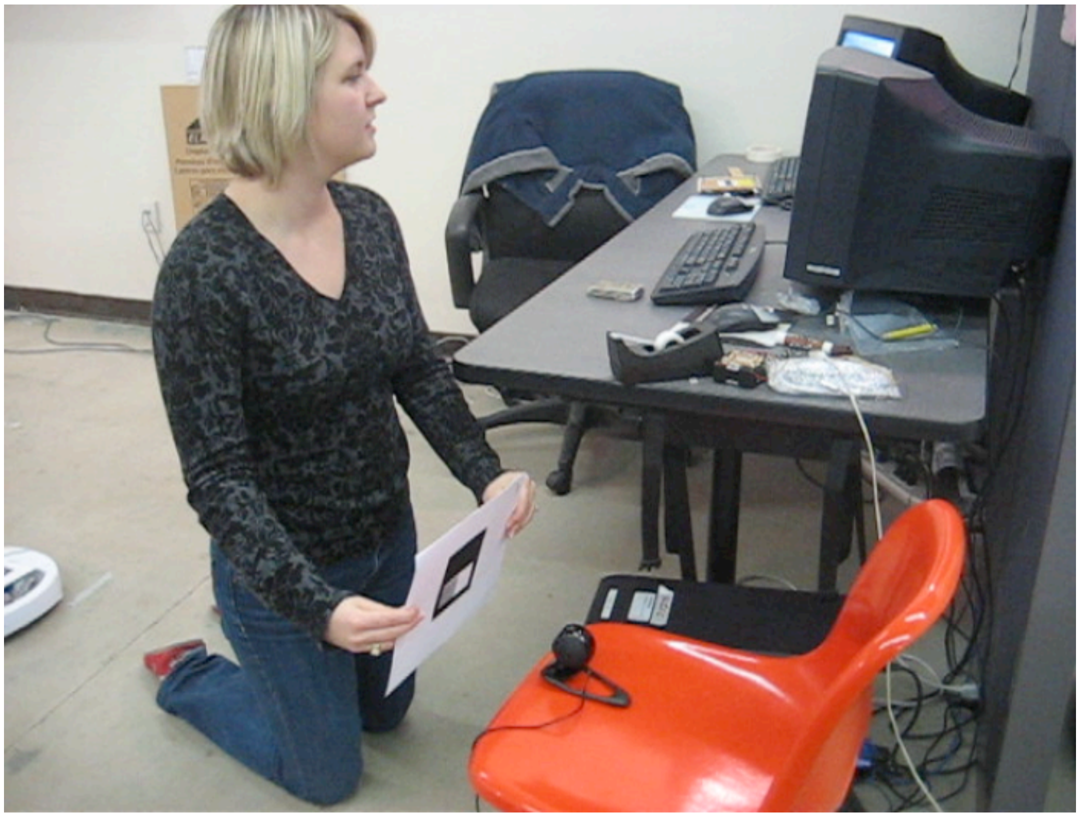


Fig. 20 Human-Robot Communication Demonstration.

The hardware demonstration was produced successfully. When pattern 1 is held up in front of the webcam, the robot travels to location 1. When pattern 2 is held up in front of the webcam, the robot travels to location 2, likewise. The battery information that was gathered was the charge and capacity of the battery as stated in Chapter II. The batter level was presented as a percentage of the battery capacity. If the percentage was less than 40%, then the robot returned to the home base. If the percentage was less than 10%, then the user was notified that the robot needed to be recharged manually. These values were found based on experimenting with the robot and if the software was implemented on another robotic platform, then different percentages might be necessary. Figures 21 and 22 show the planned paths for the robot and the actual paths the robot took when they were commanded to perform task 1 or task 2. As stated earlier, the important part is not that the robots followed the path perfectly, but that the robot traveled to the desired final destination.

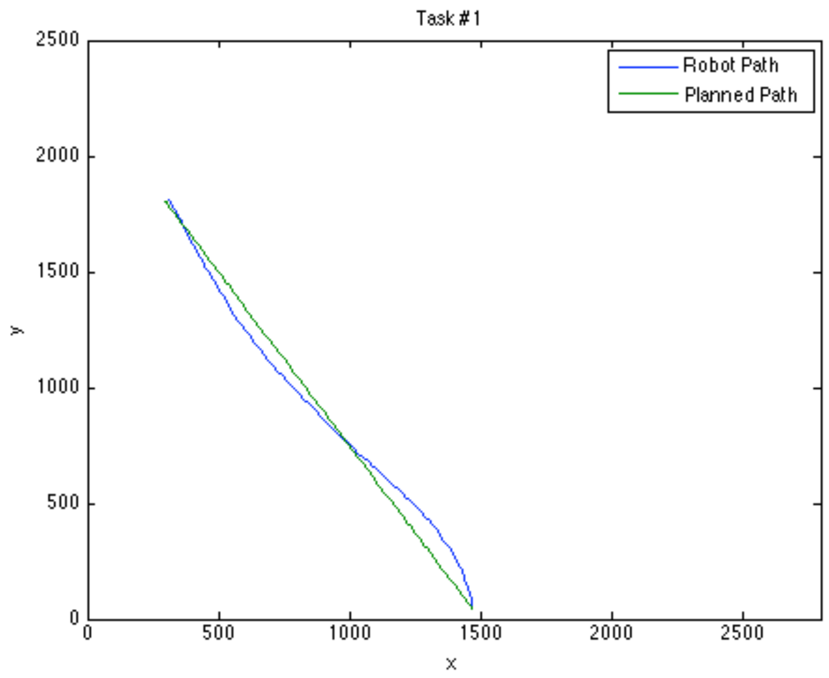


Fig. 21 Planned Path vs. Robot Path for Task 1.

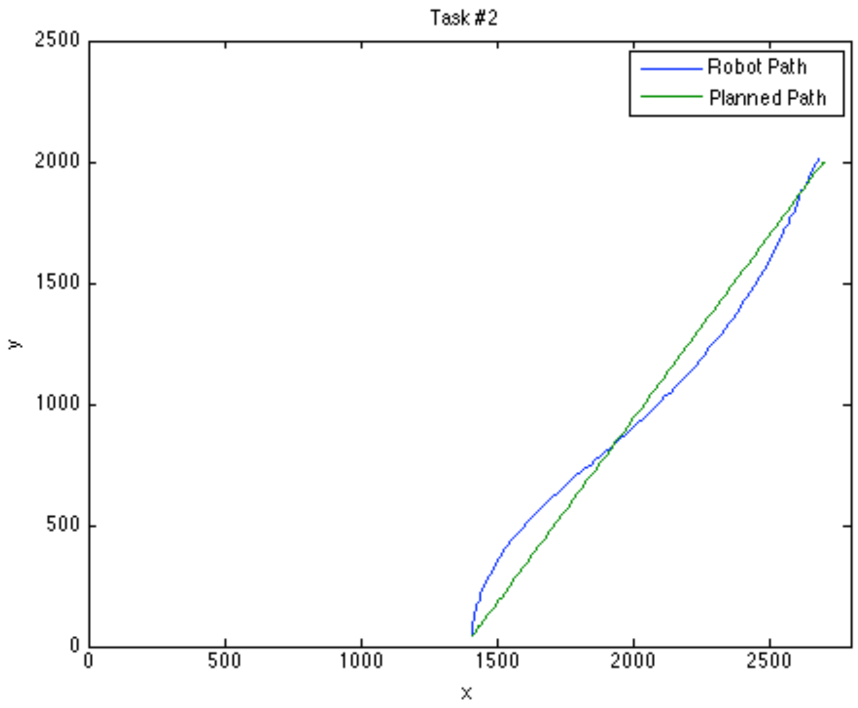


Fig. 22 Planned Path vs. Robot Path for Task 2.

CHAPTER IV

SUMMARY AND CONCLUSIONS

Autonomous robots are very useful tools to have while exploring a planetary environment. They can perform tasks in harsh conditions with minimal risk. Some of these tasks include habitat or facility construction. They can also act as an assistant to an astronaut that is collecting samples out in the field. To produce these robots, the first step is to make prototypes in a laboratory environment that can demonstrate some of the skills necessary for autonomous robotic tasks. This thesis discussed two projects that are pieces of this issue. The first project goal was to produce a hardware demonstration of cooperative robotic manipulation of an object. Some of the subsystems developed would be useful for autonomous habitat or facility construction. The second project goal was to produce a hardware demonstration of human-robot communication through vision techniques. An astronaut that is trying to communicate with the robots out in the field could use the ideas incorporated in the project.

To complete the project goals, an autonomous robotics system was used that was developed by the Space Engineering Institute's Robotics Space Colonization Team. The autonomous robotics system includes the iRobot Create© robotic platform, overhead camera and image recognition software, wireless communication network, Kalman filter, and Central PC system architecture. These subsystems were used in both projects and were necessary to complete the project goals.

In order to complete the cooperative robotic transportation project goals, several theoretical developments had to be made. The robot model and the flexible structure model produced the equations of motion for the robots and the object. The trajectory was designed for the two-robot flexible-structure system. The inputs to calculate the trajectory included stretching and bending limits of the flexible object along with the robotic platform constraints. The trajectory tracking control law allowed each robot to track its trajectory with minimal errors. After the theoretical developments were accomplished, a flow chart was designed so that every system would integrate together to complete the project goal.

In order to complete the human-robot communication demonstration, several subsystems had to be integrated into the autonomous robotics lab. A webcam was implemented along with image recognition software that scanned for square patterns that represented tasks for the robot to accomplish. Battery information was collected from the robot so that the Central PC would know if the robot could accomplish the desired task. The path-planning algorithm planned a path from an arbitrary initial position to a desired final position. The trajectory tracking control law is similar to the one in the cooperative robotic transportation project, in that the robot follows its own path minimizing position errors. A flow chart was designed to integrate these subsystems with the autonomous robotics lab to accomplish the project goals.

The overall results of the project were successful. The cooperative robotics transportation project produced successful simulation and hardware results. Plots were presented that compared the planned robot paths to the actual robot paths. The demonstrations proved that the theory presented is true. The human-robot communication project produced a successful hardware demonstration. When the user held up a pattern that represented task 1, the robot successfully completed task 1. The same was true for task 2. This proves that communicating to a robot through vision techniques can be successful.

These projects are a small step towards autonomous robots in space. Future work in this area might include implementation on a more sophisticated robotic platform in order to achieve more accurate data for the robotics problems. Other human-robot communications devices could be investigated including remote control and voice techniques.

REFERENCES

[1] "Vision for Space Exploration," NASA, http://www.nasa.gov/externalflash/Vision/index.html [retrieved 28 March 2008].

[2] Bolon, A., Holmstrom, K., Nguyen, J., Palacios, R., Soto, A., Spratlen, B., and Weitz, L., "Cooperative Transportation of a Flexible Object Towards Construction of a Martian Habitat," *2008 AIAA Region IV Student Paper Competition*, Houston, Texas, April, 2008.

[3] "iRobot Create Homepage," iRobot, http://www.irobot.com/create/explore/ [retrieved 28 March 2008].

[4] Weitz, L., Doebbler, J., Holmstrom, K., and Hurtado, J., "Cooperative Manipulation of a Flexible Object by Nonholonomically-Constrained Robots," *Journal of Intelligent and Robotic Systems* (Submitted 2008).

[5] Schaub, H. and Junkins, J., *Analytical Mechanics of Aerospace Systems*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Washington, DC, 2003.

CONTACT INFORMATION

Name:	Kristen Holmstrom
Address:	c/o Dr. John Hurtado Department of Aerospace Engineering MS 3141 Texas A&M University College Station, TX 77843
Email Address:	kristen44210@tamu.edu
Education:	B.S., Aerospace Engineering Texas A&M University, December 2008 Undergraduate Research Scholar