Distributed Vision-Based Target Tracking Control Using Multiple Mobile Robots

Anthony Le and Ryan Clue

Abstract

Coordination control of multiple robots has been an active research topic in recent years. The fundamental question is how to control individual robots for object tracking using limited local sensing/communication information. In the following, review of the literature, system, and engineering efforts are discussed.

Contents

1 Introduction

Coordination control of multiple robots has been an active research topic in recent years. There are a lot of potential applications in this field - for example, environmental monitoring, search and rescue missions, threat/obstacle evasion are useful applications. The fundamental question is how to control individual robots for object tracking using limited local sensing/communication information.

2 Review of literature and prior work

Previous work from the Class of 2016 Senior Project *Cooperative Control of Heterogenous Mobile Robots Network* was reviewed. The exact same platform of robots using the QBot2 are used for our project. Cooperative control was tested using QBot2 platform. Camera for localization and Wifi communication to exchange position information. Consensus and formation stabilization problems were studied.

Vision-based interception of a moving target with a nonholonomic mobile robot by Luigi Freda and Giuseppe Oriolo investigates control law based on angles of a camera image to track a target. In their proposed theory, a simplified explanation results in the angular velocity of the robot,

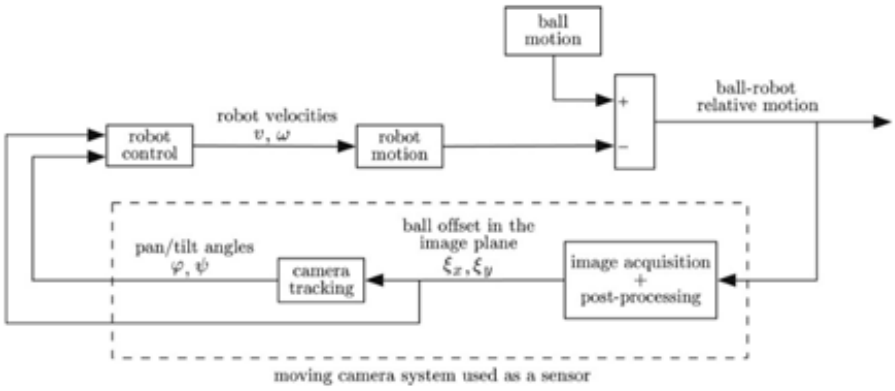
$$\omega = -K * tan^{-1}(\xi_x / f) \tag{1}$$

where ξ_x is the x-coordinate of the ball on the image plane, f is the focal length; Driving velocity of the robot:

$$V = K * d \tag{2}$$

where d is the distance to the target obtained from the kinect infrared sensor. The relation between ξ_x and f can be seen in Appendix-Figure 10.

The following block diagram is a summed up explanation of kinematic controls:



Decentralized multi-robot encirclement of a 3D target with guaranteed collision avoidance by Franchi, Stegagno, and Oriolo is one work referred to for object tracking. This article delivers a control framework for achieving encirclement of a target moving in 3D using a multi-robot system with fully decentralized and only requires local communication among robots. Each robot locally estimates all the relevant global quantities and allows for the applicability to both 3D(spatial) and 2D(planar) encirclement without modications. An integrated scheme for estimating in a decentralized wayin all the global quantities is needed by the control law. It also has a provably effective strategy for inter-robot collision avoidance. The article presents an extensive numerical validation showing applicability of the method to both holonomic point robots and underactuated UAVs and an experimental implementation on nonholonomic ground vehicles using only onboard sensors (i.e., without any external localization system) Refer to Appendix-Figure 11 and 12 to see how encirclement works.

3 Standards Applicable

In this project, the objectives are: to design distributed vision-based control algorithms for mobile robots and to implement and validate the proposed algorithms. The main tasks of the project is for target identification using camera-based images, target following using camera-based images, and multiple robot formation control with target following. Some constraints may need to be met. For example, environment will be a well-lit, indoor area - no direct sunlight; target will remain on the same level plane as the robots (2D motion tracking); target will not move faster than the Qbot 2s maximum velocity (0.7 m/s); target will be a solid colored, basketball-sized sphere; the environment will be reasonably free of similarly colored objects; and multiple robots must use same control strategy.

4 Subsystem Level Function Requirements

4.1 Top Level Design

In our top level system designs, the project is split into two distinct stages; image processing and trajectory control. The image processing stage is used to determine the centroid of not only the object being tracked but also the location of objects that the Qbot2 might collide with. The RGB Image will be used primarily to determine the target object location, while the IR Image will be used to corroborate that target image location as well as detecting obstacles in the Qbot2s immediate path, including other Qbot2s.

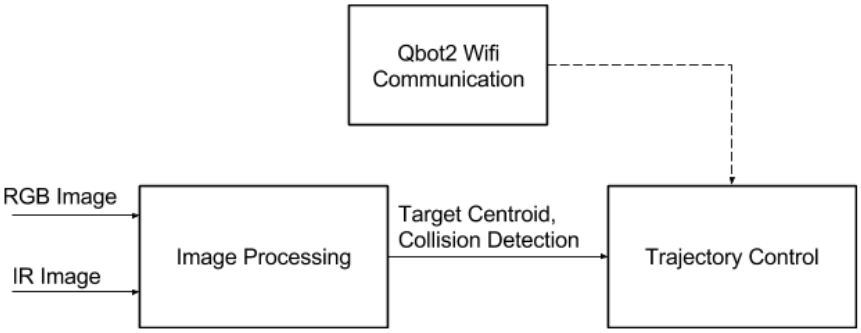


Figure 1: Top Level Design

4.2 Subsystem Level Design

We obtain our image values from the Kinect camera mounted on the Qbot2. The Kinect also performs the very significant processing work of converting the infrared image to a depth image.

Image thresholding involves checking whether or not an elements values fall within a certain range. Each pixel of a color image is a combination of several magnitudes of colors – in our case red, green, and blue. We set a range of acceptable values, and any pixels that do not meet all of these values are replaced by a 0. Pixels that do meet this value are replaced by a 1. This gives us a binary image in which (ideally) our target object is represented by the largest grouping of pixels

Blob Detection is a method of determining which pixels are grouped together and which are not. Different tolerances of the space between pixels may be selected (ie pixels must be directly adjacent to neighboring pixels, alternatively pixels must be adjacent or diagonal to neighboring pixels) when constructing the blobs. We then select the largest blob to calculate our target centroid. This means that color thresholding does not need to eliminate all non-target pixels from the environment. It does, however, need to remove enough such that the target blob is the largest in the image. Otherwise our results will be off.

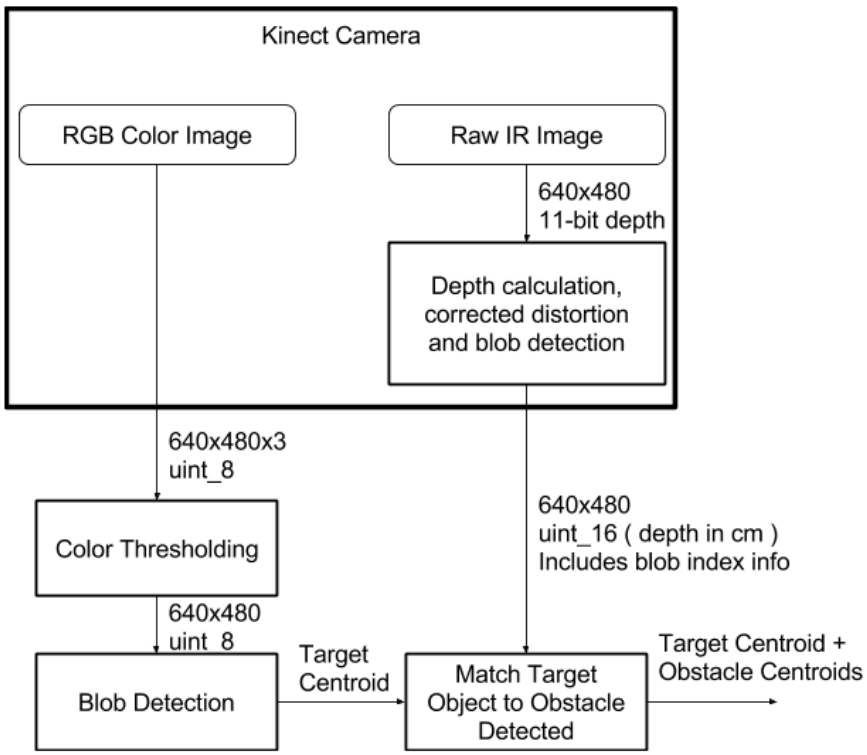


Figure 2: Subsystem Level Design

4.3 Modes of Operation

Startup Mode- The QBot2 enters this mode after being powered on. Here the QBot2 will perform all necessary initialization steps. After the basic steps have been completed, the Qbot2 will calibrate its camera and connect to a host computer via wifi. This host computer will handle all the image processing and control logic. After performing some self diagnostics the Qbot will proceed to search for a target by switching to Target Detection Mode.

Target Detection Mode - The Qbot will enter this mode if it has gone more than a couple seconds without seeing the target. In this mode the Qbot will rotate in place until it has successfully located a valid target. When a target is found, the Qbot will switch to Target Following Mode. However if after a minute of searching the Qbot has still not located a target, it will switch itself to re-calibrate mode to adjust the camera.

Target Follow Mode - In this mode the QBot2 will move to a position near the target object (note the object may be either stationary or moving at a rela-

tively low speed). During this stage the Qbot will attempt to detect and avoid collisions with its surroundings. The initial movement of a Qbot entering this stage will be determined by data from the previous Target Detection Mode results. The Qbot will continue to process updated camera images while moving towards the position of the target. If the target is not in view, however, the Qbot will switch back to Target Detection Mode and try to re-acquire the target.

Recalibration Mode - During this mode the Qbot will remain stationary while attempting to automatically calibrate its camera to current light of its environment. After this mode is completed, the Qbot will return to whichever operational mode it was in before switching to Recalibration Mode.

5 Engineering efforts completed to date

5.1 Testing Results

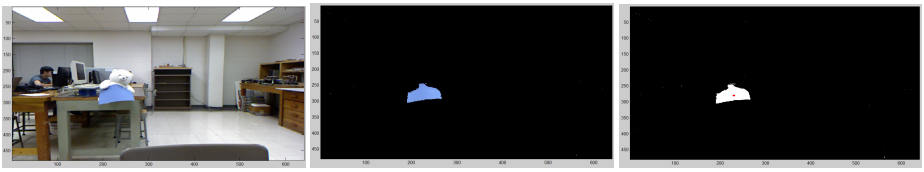
We have completed a very rudimentary design for a Qbot2 to visually track an object. The image processing portion of this experiment involved the use of color thresholding and blob detection. The kinematic model was similar to the one described earlier, which uses the target centroid to determine direction. We use the target centroid obtained from this RGB image to determine the wheel velocities. Forward velocity is calculated by omitted from the command to set wheel velocity, so that the robot will remain in place while turning to face the object.

Below this a link to a demonstration:

https://youtu.be/aCppCa-5ouU

Here is a step-by-step illustration of the sequence of image processing stages we used to calculate the target centroid.

We start with the unmodified RGB image (left image). Using a predetermined range of red, green, and blue intensities we ignore all values that do not fall within this range – this stage is called color thresholding (center image). Finally we apply a method known as blob detection (right image) to determine the largest grouping of pixels and then calculate the central location of that pixel. For the demonstration, we mark that central point of the pixel grouping on the processed image with a 3x3 square of red pixels. This red dot is a visual representation of the XY values that will be passed to the kinematic control.



The kinematics for this experiment were simply

$$v_{right} = -1 * sin(centroid_x - screen_x) \tag{3}$$

and,

$$v_{left} = sin(centroid_x - screen_x) \tag{4}$$

The variables v_{right} and v_{left} are what will be used to set the current wheel velocities. The variable $centroid_x$ is the X value of the centroid location, while $screen_x$ is the size of the X dimension of our image size.

5.2 The Kinetic Model

In a certain problem statement, a given trajectory and two robot postures are given. To track a given robot's trajectory is a problem that can be solved using nonlinear feedback and dynamic feedback linearization. A standard kinematic model must be used:

$$x_1 = \theta \tag{5}$$

$$x_2 = xcos\theta + ysin\theta \tag{6}$$

$$x_3 = xsin\theta - ycos\theta \tag{7}$$

And by using,

$$\ddot{x} = \ddot{z_1} = u_1 \tag{8}$$

$$\ddot{y} = \ddot{z_2} = u_2 \tag{9}$$

where,

$$\dot{x} = \xi cos\theta \tag{10}$$

$$\dot{y} = \xi sin\theta \tag{11}$$

and when the kinematic model is not moving,

$$\xi = v = 0 \tag{12}$$

We can construct a dynamic state feedback model for exact linearization purposes with nonholonomic constraints. Following the proposed model, we can implement trajectory tracking using the dynamic feedback linearization method.

The feedback design of the nonlinear controller based on dynamic feedback linearlization is fairly straightforward. Using a linear and decoupled system, the following feedback for desired trajectory has been postulated:

$$u_1 = \ddot{x_d} + k_{p1}(x_d - x)k_{d1}(\dot{x_d} - \dot{x}) \tag{13}$$

$$u_2 = \ddot{y_d} + k_{p2}(y_d - y)k_{d2}(\dot{y_d} - \dot{y}) \tag{14}$$

Under the proposed dynamic feedback linearization method, the second order linear model is obtained as given in equations(4,5). The PD controller in equations(9,10) is then applied to solve the trajectory tracking control problem. How should control gains k_p and k_d be selected in our simulation one may ask?

The gains can be set arbitrarily. Following,

$$k_1 = k_3 = 2\xi\sqrt{\omega_d^2(t) + bv_d^2(t)} \tag{15}$$

$$k_2 = b \tag{16}$$

where ξ is any number between 0 and 1, and b > 0.

To set better defined values for trajectory tracking gains, we set:

$$k_1 = K_3 = 2\xi\sqrt{a} \tag{17}$$

$$k_2 = (a^2 - (w^d(t))^2)/v^d(t) \tag{18}$$

where a > 0 and ξ is any number between 1 and 0. For our values of ξ and a, we use .7 and 1, respectively.

In the proposed PD control, velocities $\dot{x}$ and $\dot{y}$ are required. To generate those values in your simulation, we compute the values from the kinematic model given in equations (1-8). Generating the derivative values for the kinematic model, we can start to develop the trajectory path of the circle. This is done by using these equations:

$$x = center(1) + R * cos(\gamma * dt); \tag{19}$$

$$y = center(2) + R * sin(\gamma * dt); \tag{20}$$

where dt is the time-step for interation implementation, gamma is the parameter for reference trajectory, and center referring to the center of the circle.

The equation for the figure-8 path is as follows:

$$x = center(1) + R * sin(2 * \gamma * dt); \tag{21}$$

$$y = center(2) + R * sin(\gamma * dt); \tag{22}$$

Computing the $\ddot{y}$ and $\ddot{x}$ values are essential to determine the relativistic trajectories. The equations for $\dot{y}$ and $\dot{x}$ are just the derivatives of equations (15,16 and 17,18).

Matlab's ODE45 helps integrates and solve the system of differential equations given with initial conditions. To find appropriate values for a circle or figure 8 shape, differential equations of the desired trajectory are also generated within Matlab. Given the approximately-linearized tracking error dynamics,

$$\epsilon_x = x_d - x, \epsilon_y = y_d - y, \epsilon_\theta = \theta_d - \theta \tag{23}$$

we can develop a algorithm that minimizes error based on the kinematic and linearized model. The proposed error model (19) can be translated and turned into the matrix array:

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \omega^d & 0 \\ -\omega^d & 0 & \nu^d \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{24}$$

From here, another set of u_1 and u_2 values can be utilized.

$$u_1 = -k_1 e_1 \tag{25}$$

$$u_2 = -k_2 v^d (sine_3/e_3) e_2 - k_3 e_3 \tag{26}$$

With initial conditions for posture being set to,

$$x = 1.5, y = 1.5, \theta = pi/2 \tag{27}$$

we can work with a kinematic model with initial conditions for trajectory tracking control.

A wall or obstacle avoidance problem can also be solved assuming these contraints: Given a robot put inside a circular chamber, a trajectory that maps out the walls can be used in order to solve a wall avoidance problem. Here, a circular trajectory is given to guide the inside of the chamber. The simulated robot must be placed within bounds of the wall and must follow the trajectory in order to avoid wall collision.

5.3 Simulation of Kinetic Model

The following are simulation results for the given problem and solutions:

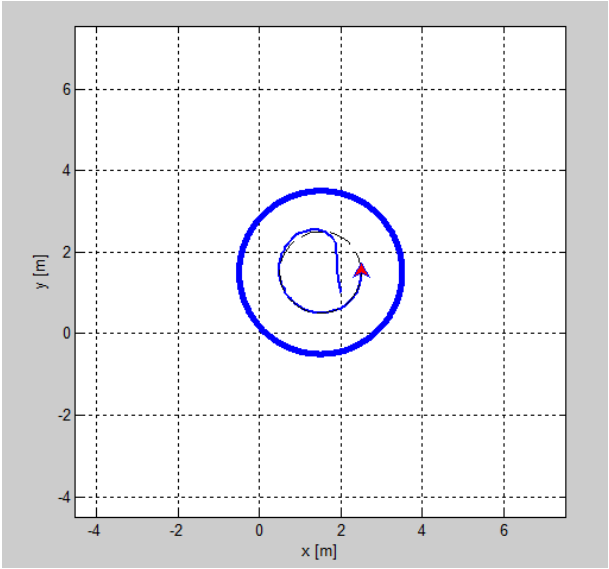


Figure 3: Demonstration of trajectory tracking avoiding a circular wall

A demonstration of how a following a trajectory can help avoid wall collision are seen in figure 3. The larger band implies a wall. We can see the robot starts at initial pose (1.5, 1.5) and follows the trajectory of a circle given. Based on simulation, obstacle or wall avoidance is possible if given a trajectory path avoiding the wall.

In the following figures, figure 4 and figure 5, we can see how the robot follows the given circular trajectory path.

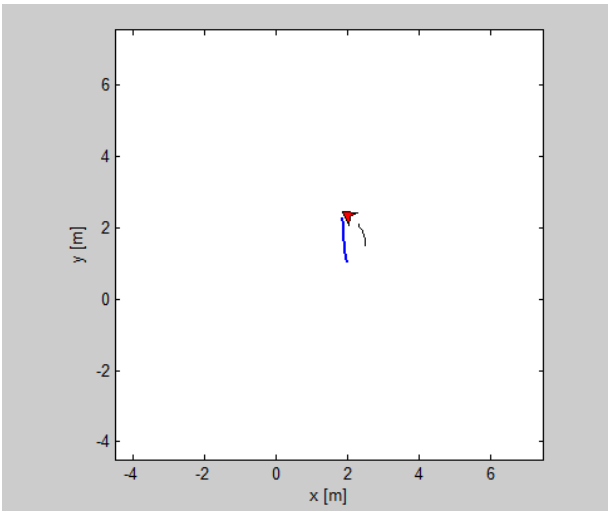


Figure 4: Initial demonstration of trajectory tracking avoiding a circular wall

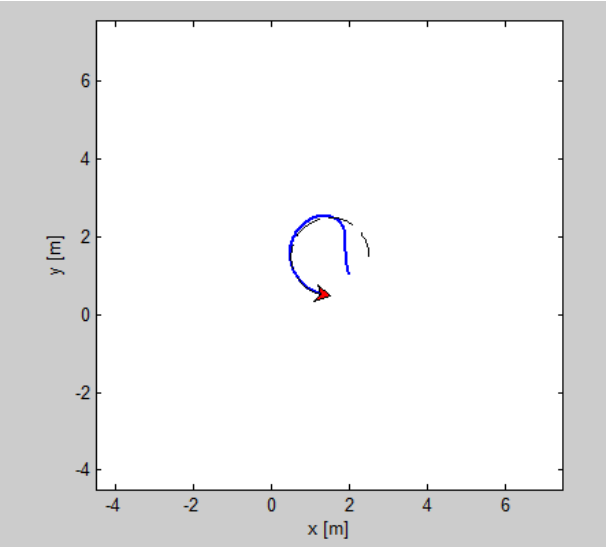


Figure 5: After several timesteps, trajectory tracking avoiding a circular wall

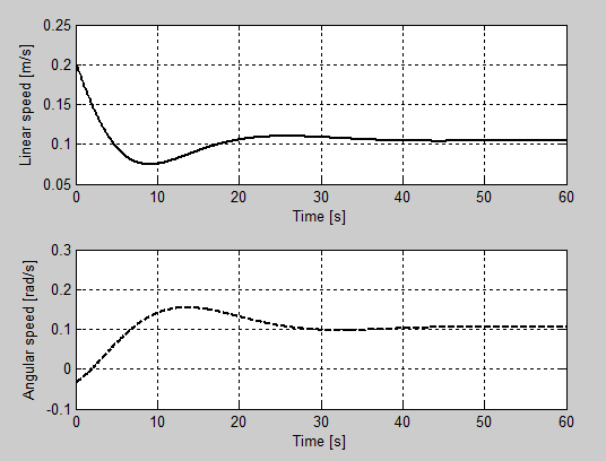


Figure 6: Linear and angular speeds of simulated control for circular path

As we can see, trajectory following can be implemented given a circular trajectory.
Following, trajectory tracking of a figure-8 path using a nonlinear feedback model was simulated.

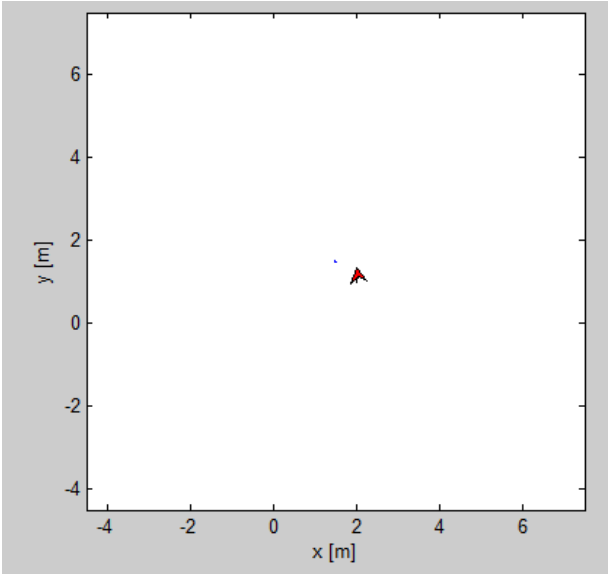


Figure 7: Initial conditions of trajectory tracking following a figure-8 path

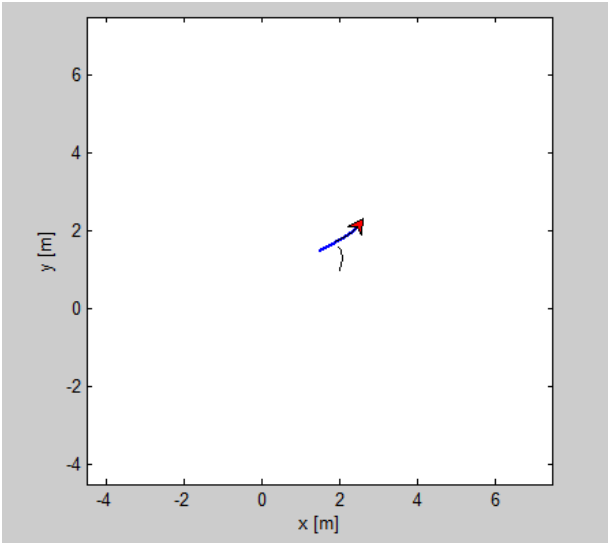


Figure 8: After several timesteps, trajectory tracking following a figure-8 path

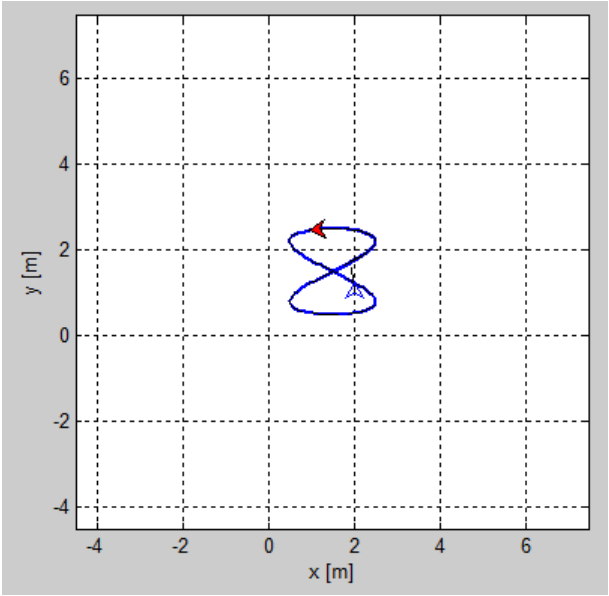


Figure 9: Trajectory tracking following a figure-8 path - finished

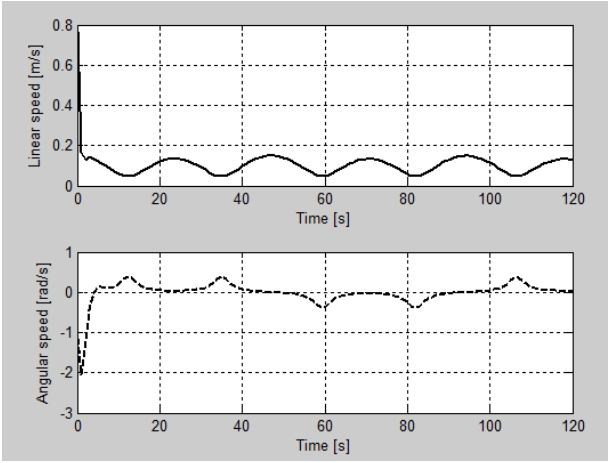


Figure 10: Linear and angular speeds of simulated control for figure-8 path

Based on figure 5-10, we can see trajectory control is reliably implemented. Using non-linear control from equations 13 and 14 transformed to 25 and 26, we can see the implementation for dynamic feedback linearization is possible.

6 Parts List

The only parts we are using for this project are the three Qbot2s in the Bradley University robotics lab. They were purchased for about $1,000. The Matlab software suite that we use to program the Qbot2s also came packaged with the original purchase I believe. We also must use a Wifi USB module so that we can connect to two networks at the same time; the Qbot2s are configured for an ad hoc network while the Matlab certificate is obtained through Bradley Universitys secure network.

7 Deliverables, Division of Labor, and Schedule for Completion

Ryan will be handling the image processing portion of the experiment, while Anthony will be focusing on the kinematic models.

8 Discussion and future directions

Originally we thought that the work of image processing would be limited to blob detection and color thresholding, but after getting some feedback on our project its looking like we will have to use some more sophisticated methods to detect the target object. Currently we are looking at edge detection and background subtraction as alternatives to color thresholding.

9. References

[1] G. Oriolo. Wmr control via dynamic feedback linearization: design, implementation and experimental validation. *IEEE Trans. on Control Systems Technology*, 10:835852, 2002.

[2] Franchi, Stegagno, and Oriolo. Decentralized multi-robot encirclement of a 3D target with guaranteed collision avoidance. 2016

[3] Freda and Oriolo. Vision-based interception of a moving target with a non-holonomic mobile robot. 2007

[4] S. Miah, Lab 2: Trajectory Tracking Using Differential Drive Mobile Robots. 2016 .

[5] Cap 02.pdf. [Online]. Available: http://mayerle.deps.prof.ufsc.br/private/eps6405/Cap 2002.pdf. [Accessed: 14-Nov-2016]

[6] MRNProposalDocument.pdf. [Online]. Available: http://ee.bradley.edu/projects/proj2016/mrn/MRNProposalDocument.pdf. [Accessed: 14-Nov-2016]

[7] Quanser - QBot 2 for QUARC. [Online]. Available: http://www.quanser.com/products/qbot2. [Accessed: 14-Nov-2016].

10 Appendix

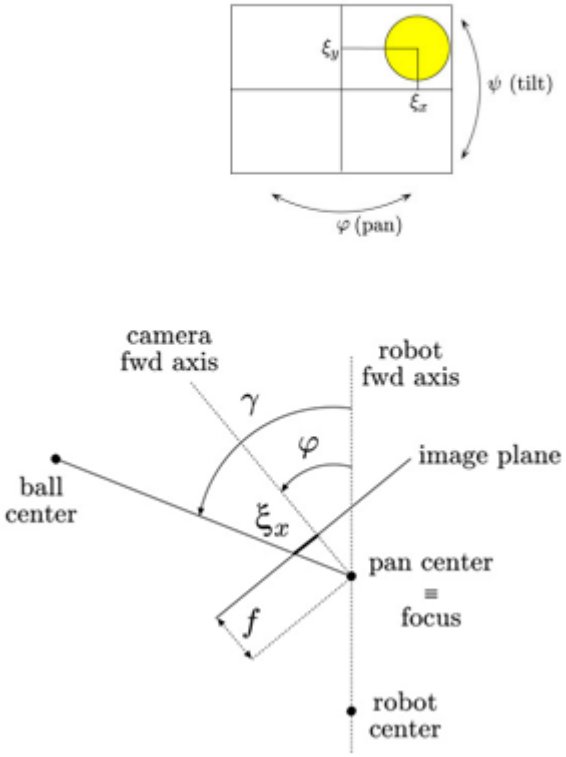


Figure 10: Example of ξ_x and f

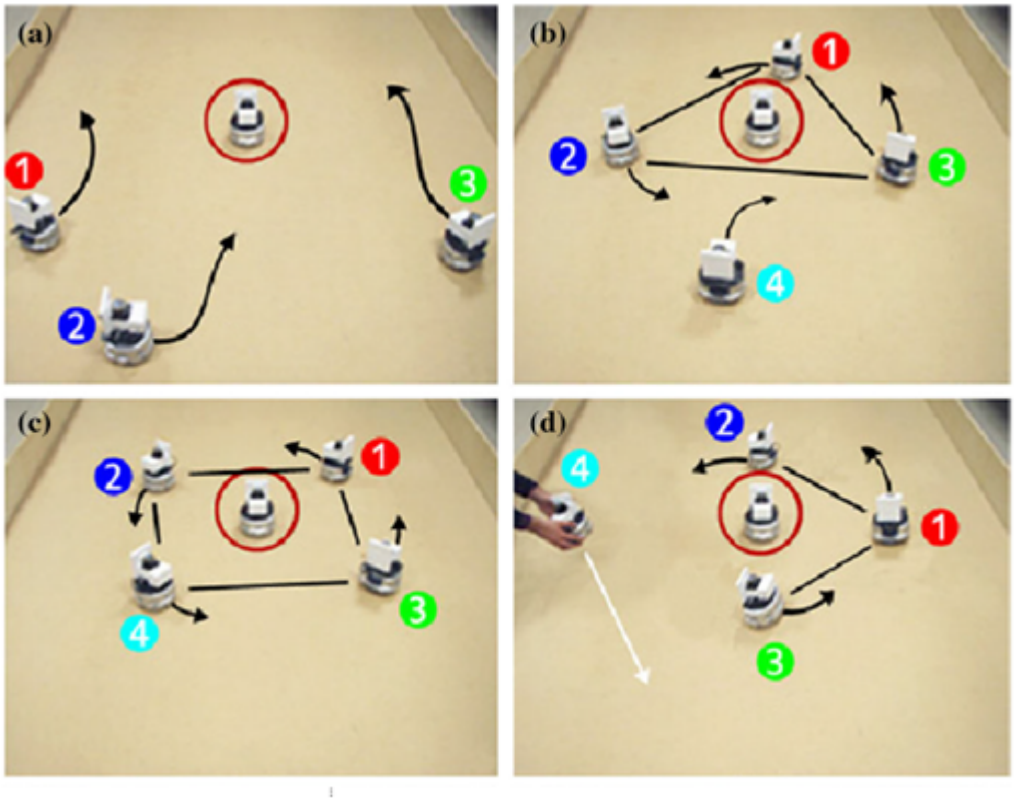


Figure 11: Encirclement Example 1 taken from Oriolo et al. [2]

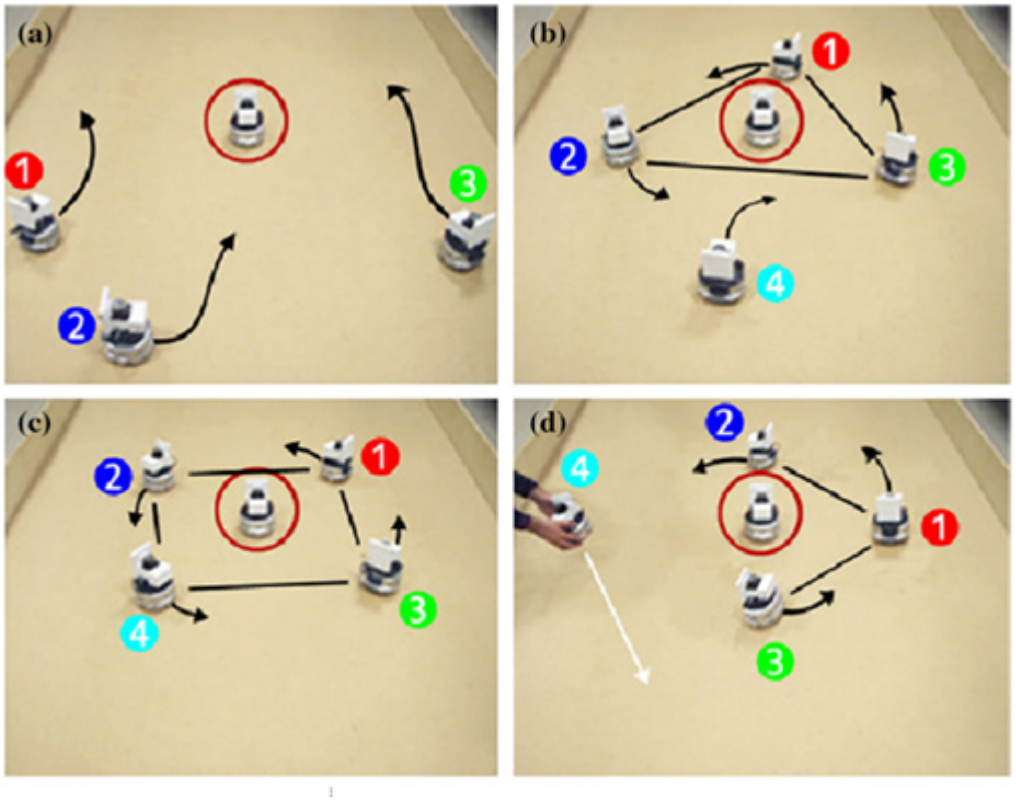


Figure 10: Encirclement Example 2 taken from Oriolo et al. [2]