# Controlling a Unicycle Ensemble with a Single Command Input

**Gregory Bales**[*]
University of California Davis
Davis, CA 95618
glbales@ucdavis.edu

## Abstract

This report summarizes the work performed for the final project in MAT258a. A method of controlling an ensemble of unicycle robots is presented along with some of the insight gained in the exercise.

## 1 Contextual Overview

My research attempts to address problems that arise in interactions between humans and robotic agents. Information exchanges between humans and large swarms of autonomous robots is known as human swarm interaction (HSI) problem. HSI poses a number of unique challenges. For example, an ensemble of agents must communicate its vast amount of sensor information to the human in a way that is tractable and meaningful. Conversely, the human must have the capacity to interact with the swarm through his or her limited number of sensory channels. This problem does not scale linearly as our attention is quickly overwhelmed when presented with large amounts of information. Such a scenario would preclude any decision making advantages that we might be able to offer. As a first step in an effort to simplify human-to-swarm communication, I will explore an open loop policy that will limit the control input to a single channel for a set of non-holonomic, unicycle robots. As a consequence, the ensemble of $n$ agents will be commanded by a single control signal, broadcast to the entire swarm.

## 2 Ensemble Control

One way of simplifying the effort needed to control the swarm is to manipulate the global nature of the ensemble without controlling the specific configuration of every element. This will allow the use of a small amount of information (a single control signal) to effect a large array of $n$ robots. We must understand however, that there is something to be lost in this effort. Analogously, we can control the global temperate of some mass by injecting heat into a single small area and letting conduction take care of the rest. In doing so, we loose the means to control the evolution of the temperature at all other points in the system.

## 3 The Unicycle

A unicycle is a single wheel which rolls on a planer surface. Direction is controlled by changing the orientation of forward movement. It is by its nature non-holonomic. A description of the equations of motion in terms of two degrees of freedom can be misleading. A practical system generally consists of a robot with two drive wheels and a rolling caster for stability. While the unicycle behavior of this configuration is the same, the embodied configuration has three degrees of freedom:

[*]Department of Mechanical Engineering

position and orientation, any two of which are independent. A point in the $x - y$ plane can only be reached by movement through $v - \omega$ space. While we can adjust $v$ and $\omega$ independently, we cannot do so with $x$ and $y$. Rather, $x = x(v, \omega)$ and $y = y(v, \omega)$.

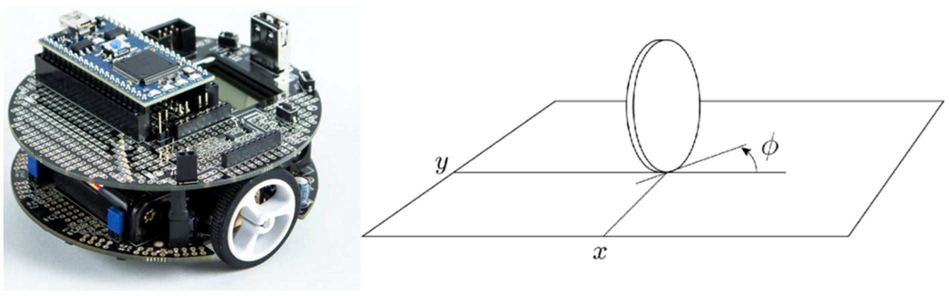


Figure 1: A typical differential drive robot modeled as a unicycle rolling on the $x - y$ plane

The description of the linear and angular velocities of the unicycle is given by the following non linear expression.

$$\dot{q}(t) = \begin{bmatrix} v(t)\cos(\omega t) \\ v(t)\sin(\omega t) \\ \omega \end{bmatrix} \tag{1}$$

## 3.1 Discretizing the Motion of the Unicycle

In general, motion of the unicycle is achieved through the continuous variation of $v$ and $\omega$. We can simplify the analysis, and consequently the control, by introducing an artificiality in the way in which we allow the motion to proceed. Let us assume that changes in position will be lumped into discrete steps. We can break this overall motion into a forward movement, and an independent rotational movement. Therefore, the robot is either turning in place, or moving forward. For clarification, the forward movement is described by the $u_1(k)$ term, while the turn in place is described by $u_2(k)$.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + u_1(k) \begin{bmatrix} \cos(\theta_0 + \theta(k)) \\ \sin(\theta_0 + \theta(k)) \\ 0 \end{bmatrix} + u_2(k) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

In this work, we are not interested in the controlling the orientation of the robot between the start and the goal. We only need that it get from one point to the other. As such, we are free to use the angle $\theta(k)$ to control the means by which we approach the goal without the need to orient the robot along the path. Therefore, we can eliminate the $u_2(k)$ term and assume that at each step, the robot will turn an amount $\Delta\theta = \epsilon_i \phi$. This leaves our discrete state equation as

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + u(k) \begin{bmatrix} \cos(\theta_0 + \epsilon_i k\phi) \\ \sin(\theta_0 + \epsilon_i k\phi) \end{bmatrix}. \tag{3}$$

## 3.2 Compact Expression of the Equations of Motion

Now that we have broken the motion down into two steps, a pattern arises that we can take advantage of. For every robot $i$, the first step beyond the initial condition $(x_0, y_0)^T$ takes us to

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + u(1) \begin{bmatrix} \cos(\theta_0 + \epsilon_i \phi) \\ \sin(\theta_0 + \epsilon_i \phi) \end{bmatrix}.$$

From equation 3, we can see that the $k + 1^{th}$ step takes us to

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + u(1) \begin{bmatrix} \cos(\theta_0 + \epsilon_i \phi) \\ \sin(\theta_0 + \epsilon_i \phi) \end{bmatrix} + u(2) \begin{bmatrix} \cos(\theta_0 + \epsilon_i 2\phi) \\ \sin(\theta_0 + \epsilon_i 2\phi) \end{bmatrix} + \cdots + u(k) \begin{bmatrix} \cos(\theta_0 + \epsilon_i k\phi) \\ \sin(\theta_0 + \epsilon_i k\phi) \end{bmatrix},$$

which we can represent as

$$x_i(k) = x_i(0) + \sum_{j=1}^{k} u(j)B_i(j) \quad \text{where } B_i(k) = \begin{bmatrix} \cos(\theta_0 + \epsilon_i k\phi) \\ \sin(\theta_0 + \epsilon_i k\phi) \end{bmatrix}.$$

We can condense the individual $B_i(k)$ terms into a matrix, and we are left with the $k$ step evolution of the $i^{th}$ robot.

$$C_{ik} = [B_i(1), B_i(2), \cdots, B_i(k)] \in \mathbb{R}^{2\times k} \tag{4}$$

$$x_i(k) = x_i(0) + C_{ik}u, \quad u \in \mathbb{R}^k \tag{5}$$

Finally, we can combine the $C_{ik}$ row vectors into a matrix that encodes the movement for the entire ensemble of $n$ robots through $k$ steps.

$$C_{nk} = \begin{bmatrix} B_1(1) & B_1(2) & \cdots & B_1(k) \\ B_2(1) & B_2(2) & \cdots & B_2(k) \\ \vdots & \vdots & \ddots & \vdots \\ B_n(1) & B_2(2) & \cdots & B_n(k) \end{bmatrix} \in \mathbb{R}^{2n\times k}$$

This means that the total motion of the ensemble of $n$ robots, from the initial position $x(0)$ to the $k^{th}$ step is given by the following expression where $C_{nk}$ grows by one column for each step $k$.

$$x(k) = x(0) + C_{nk}u \tag{6}$$

The motion obtained in this manner consists of forward jumps and turns in space. If need be, the robots are allowed to turn in place by $\epsilon_i\phi$ for each step until a profitable direction is found.

## 4 Control Properties

### 4.1 Controllability

Next we would like to show that the system of $n$ unicycle robots can be controlled to any point in the $x - y$ plane. We need to prove that the system of unicycle robots is completely state controllable. In order to achieve this, we must ensure that each robot in the ensemble responds to the control signal in a unique way. The parameter $\epsilon_i \in (0, 1]$ encodes this property. It scales the effective angular velocity, which decouples the turning characteristics of each robot. Therefore, a single command $u(k)$ applied to every robot results in $n$ unique movements. Mathematically, we can state this by showing that the matrix $C_{nk}$ is full rank. In order to do so, we must recognize that so long as $\epsilon_i \neq \epsilon_j \quad \forall i, j \in I = \{1, 2, \cdots, K\}$, we have sinusoids that are orthogonal over $\theta = [-\pi, \pi]$. Every row of $C_{nk}$ is composed of a sequence of sinusoids of different frequencies and therefore, every row is linearly independent from the other rows.

### 4.2 Stability

As we will see shortly, the stability constraint becomes a natural feature of the optimal solution. However, the system is only $K$ step stable. It is guaranteed to converge to the equilibrium position from the initial condition only, and not between from other point in the domain. Any disturbance encountered by any one of the robots cannot be corrected by $u$. It is in effect, open loop.

## 5 Forming the Optimization Problem

Much of the work in forming the problem comes in the way in which the system is simplified. Each of the $n$ robots moves in two degrees of freedom. As long as $k > 2n$, our system us under-determined and we are left with the ability to minimize the control effort $u$ over the total of $K$ steps. In my initial formulation of the problem, the series of constraints included an adherence to the dynamics of the system $x(k + 1) = f(x(k), u(k))$; an adherence to the terminal constraint $x(K) = x_f$; a guarantee of global stability.

Two of these constraints are captured in equation 5 which describes the evolution of the system over all $k$ with the terminal condition $x(K) = x(0) + C_{nK}$ at the final step $k = K$. The third is a consequence of forming the optimal solution over $K$ steps.

Thus, the optimal value of the control effort can be expressed

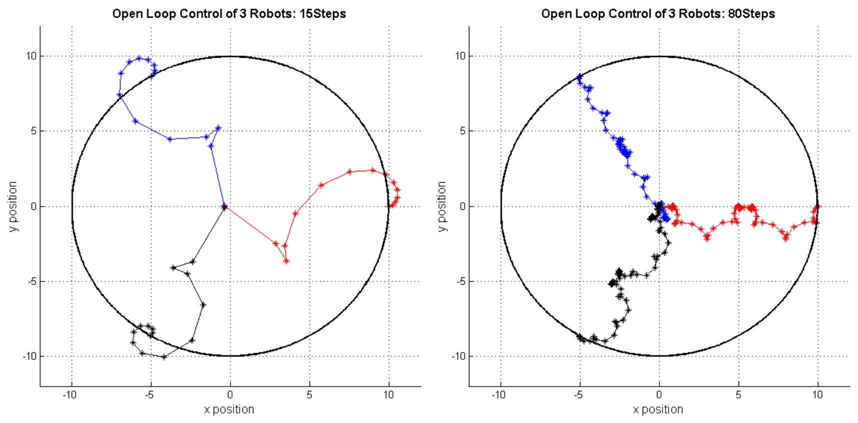
$$u^{opt} = argmin \frac{1}{2} \sum_{i=1}^{K} \|u_i\|^2 \quad s.t. \quad C_{nK} u + x(0) - x(K) = 0 \tag{7}$$

Solving the constrained optimization problem gives us, $u^{opt} = C_{nK}^T (C_{nK} C_{nK}^T)^{-1} (x(0) - x(K))$.

## 6 Results

The resulting behavior of the system under this control policy is difficult to capture in a small number of plots. In the following figures, we can see how the paths evolve as a function of total steps $K$. These systems were modeled in Matlab. Despite the fact that the method is advantageous for a large array of robots, a limited number are presented here for ease of comparison. The distances presented here have no physical meaning per se. A step of 1 unit represents the distance traveled in the time duration framed by changes in $k$. This is modeled as a discrete process where the command velocity $u(k)$ is given in whatever units we choose. For clarity, we will choose the distance units to be meters.

In Figure 2, we can see that the total number of steps $K$ changes the nature of the path. In these configurations, the shortest path to the goal is a straight line. However, the robots must turn by $\Delta\theta$ for each step. As the total number of steps increases, the paths become straighter since there are more opportunities for the controller to turn. One of the other natural features of this method is the



(a) (b)

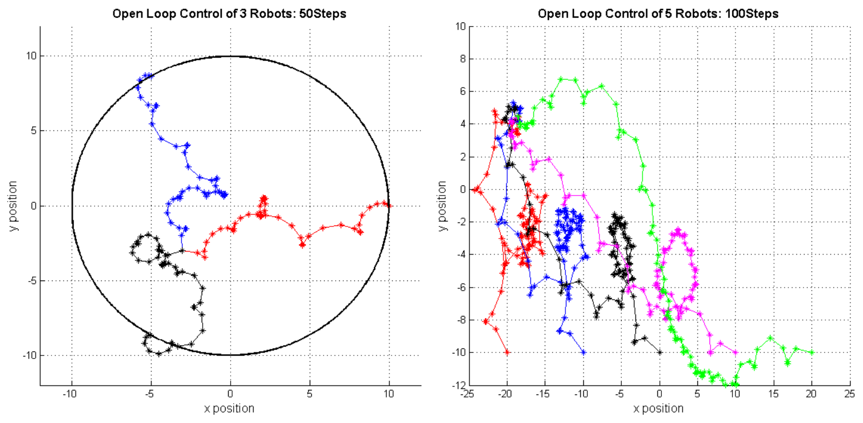
Figure 2: (a) Path of 3 robots, starting equally spaced on a circle and finishing at the origin, $K = 15$. (b) Path of 3 robots, starting equally spaced on a circle and finishing at the origin, $K = 80$.

fact that the optimal solution forces the path lengths of every robot to be the same. Therefore, if the target state is close to the initial state for a particular element of the ensemble, it must move far afield in order to return. This behavior is illustrated in Figure 3.

## 7 Discussion

### 7.1 The Problem with Model Predictive Control

This work originally started with the notion of combining closed loop control of an ensemble of robots within an Model Predictive Control framework that would take a human input into account.



(a) (b)

Figure 3: (a) Path of 3 robots, starting equally spaced on a circle and finishing at (-3,-3), $K = 50$ (b) Path of 5 robots, starting equally spaced on a line and finishing at (-20,4), $K = 100$

This objective included the difference between a human input $v_i$, in the form of a location, and the positions of all other robots in the ensemble. The idea is that the human would have limited authority over the robotic ensemble by directing it to a specific point along the trajectory. This optimization problem is given by equation 8.

$$u^{opt} = argmin \frac{1}{2} \sum_{i=1}^{k} \left\| \sum_{j=1}^{N} v_i - u_i B_j(i) \right\|^2 \quad s.t. \quad C_{nK} u + x(0) - x(K) = 0 \tag{8}$$

Again, a solution to this problem can be obtained in a closed form, where $A_\alpha$ and $\gamma_K$ are terms that encapsulate the sum of the $B_j(k)$ terms in the norm.

$$u^{opt} = A_\alpha^{-1} C_{nK}^T (C_{nK} A_\alpha^{-1} C_{nK}^T)^{-1} (\delta - n C_{nK} A_\alpha^{-1} \gamma_K) + n A_\alpha^{-1} \gamma_K$$

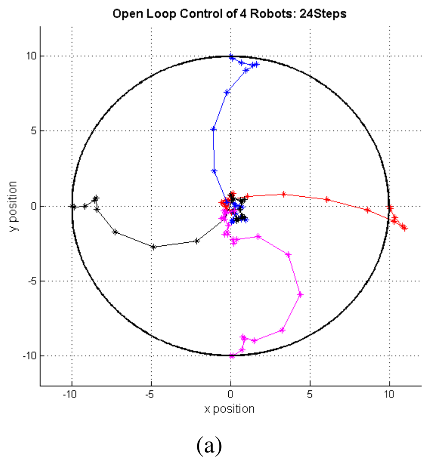
It turns out that there is no benefit to implementing this in a general way. The human input does not affect the positions of the robots locally in space, but globally over all the paths in the ensemble. The input $v_i$ only acts to redistribute the way in which the solution evolves over the $k$ steps. Furthermore, implementing an MPC by taking the first optimal step and then recomputing the optimization is ineffective. The full effect of $u^{opt}$ is not captured in one step, but as an aggregate of $K$ steps. The paths modeled under the MPC policy would not converge to any point, but rather they spun around in a circle as the robots oriented themselves during that first step $u^{opt}(1)$.

## 7.2 The Minimum Path Length Controller

While we have shown that the controller will converge for values of $k > 2n$, the resulting paths are not always efficient. We can also find values of $K$ for which the path lengths are shortest for a given configuration. Since the calculation of the optimal path is so cheap, we can create a second optimization loop for in which the path length is compared for different values of $K$. The value of $K_{min}$ that produces the shortest path is used to calculate the $u^{opt}$. Thus we have simultaneously minimized the control effort, as well as the path length.

$$K_{min} = argmin \frac{1}{2} \sum_{i=1}^{K-1} \|x(k+1) - x(k)\|^2 = argmin \frac{1}{2} \sum_{i=1}^{K-1} \left\|C_{n(k+1)} u^{opt}(k+1) - C_{nk} u^{opt}(k)\right\|^2$$

An example of path length minimization is illustrated in Figure 4. Here we have 4 robots the start equally spaced on a circle. In Figure 4a, the robots converge to (0,0) in 24 steps with a path length of 17.42m. In Figure 4b, the target is moved to (5,0) and robots converge in 47 steps with a path length of 21.37m.



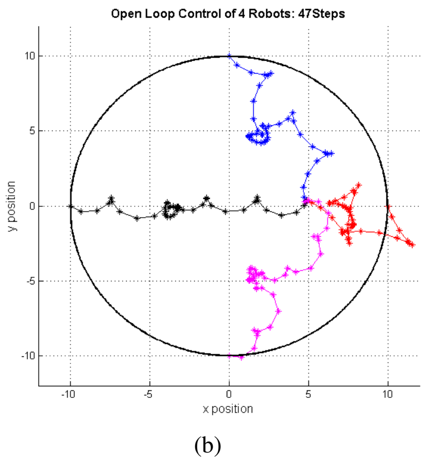


Figure 4: Minizing path lengths (a) Path of 4 robots, starting equally spaced on a circle and finishing at (0,0), $K = 24$ (b) Path of 4 robots, starting equally spaced on a line and finishing at (5,0), $K = 47$

### 7.3 The Human in the Loop

The natural progression of this work is to place the human in the loop. Given the control method presented here, there are several ways in which we could do this. The human could select one or more final positions and the swarm will calculate how to get there. Furthermore, the human can stop the progression of the controller at any point, reassert one or more of these new targets and allow the ensemble to proceed along its new course.

## 8 Conclusion

I have presented a means by which a single, optimal open loop command vector may be derived to steer an array of $n$ robots from an initial to final position. The start and end goals for the individual robots need not be the same. Provided that the number of steps $K > 2n$, we are guaranteed to converge. Much of this work is based on [1],[2]. While my initial attempt to combine MPC with the multi-agent controller in [3] has been unsuccessful, it has not been fruitless and I will make further attempts to re-frame the problem as my research progresses.

While these optimal solutions were derived as a closed form, all of the results were compared against those obtained using the Matlab Optimization toolbox. In fact, the problem presented in equation 8 took approximately 200 iterations, while the closed form solution was solved in a fraction of a second.

### References

[1] A. Becker and T. Bretl, Approximate Steering of a Unicycle Under Bounded Model Perturbation Using Ensemble Control, IEEE Transactions on Robotics, vol. 28, no. 3, pp. 580591, Jun. 2012.

[2] A. Becker, C. Onyuksel, and T. Bretl, Feedback control of many differential-drive robots with uniform control inputs, in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, 2012, pp. 22562262.

[3] R. Chipalkatty, G. Droge, and M. B. Egerstedt, Less is more: Mixed-initiative model-predictive control with human inputs, Robotics, IEEE Transactions on, vol. 29, no. 3, pp. 695703, 2013.

[4] A. De Luca, G. Oriolo, and M. Vendittelli, Stabilization of the unicycle via dynamic feedback linearization, in 6th IFAC Symp. on Robot Control, 2000, pp. 397402.

[5] J. Nocedal and S. J. Wright, Numerical optimization, 2nd ed. New York: Springer, 2006.