KDC Final Report

Jennifer King, Igor Napolskikh, Aaron Walsman
The Robotics Institute, Carnegie Mellon University
{*jeking,inapolk,awalsman*}@andrew.cmu.edu

May 8, 2014

1 Abstract

In this project we implemented a control system for the nonholonomic drive system in the HERB robot. The HERB robot uses a Segway base consisting of two wheels in a differential drive configuration, with an additional caster wheel to provide balance and support. We represent this using a unicycle kinematics model, where the desired trajectory is described by an ordered path of discrete waypoints. Therefore we address the following problems: *path following and control, and vehicle modelling via system identification.* To address these challenges we implemented control software that integrates with the simulated and phsyical HERB robotics system.

2 Related Work

The subject of single-track nonholomic drive systems has been extensively discussed in the robotics literature. The lattice based $A^\star$ planner proposed in [1] provides a planning method that utilizes discrete action sets to accomodate the constraints inherent in a nonholonomic system. The orbital tracker proposed in [3] provides much of the desired control behavior but with a slightly more complex bicycle steering model. Therefore we also look to the linearized control model proposed in [2] which more closely resembles the kinematics present on the HERB robot.

3 Approach

We designed a controller for base movements for the HERB robot. Figure 1 shows a high-level overview of the complete control system. We define the state of the vehicle $q = (x, y, \theta) \in SE(2)$. Given a path ξ defined by a sequence of poses, $q_i = (x_i, y_i, \theta_i) \in SE(2)$ for $i = 1...n$, we wish to generate control commands, $u_c = (v_c, \omega_c)$, in the form of forward and rotational velocity which drive the robot along the path. The control system utilizes pose feedback, q_a, provided

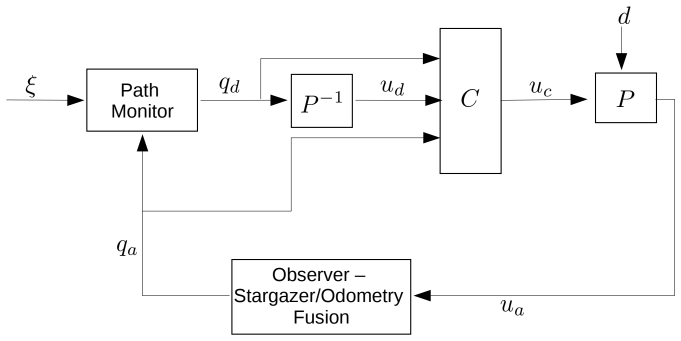

Figure 1: A high-level view of the control system.

by an onboard localization system to compensate for unmodeled disturbance, d, in the system. In the following sections we define each of the components of the control system.

3.1 Feedforward Command Generation

We model the robot as a differential drive vehicle. The kinematics of the differential drive vehicle allow us to relate (v, ω) to the change in state through the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

Given a desired pose q_d on the path we wish to generate a feedforward control $u_d = (v_d, \omega_d)$ by inverting our model of the differential drive vehicle. For feedforward control generation we ignore the desired heading, θ_d, and utilize only the defined position, (x_d, y_d). First we consider foward velocity v_d. From Equation 1:

$$\dot{x}_d^2 = v_d^2 \cos^2\theta \tag{2}$$

$$\dot{y}_d^2 = v_d^2 \sin^2\theta \tag{3}$$

Combining Equations 2 and 3 we get:

$$\dot{x}_d^2 + \dot{y}_d^2 = v_d^2(\cos^2\theta + \sin^2\theta)$$

$$v_d = \pm\sqrt{\dot{x}_d^2 + \dot{y}_d^2} \tag{4}$$

For angular velocity we again use Equation 1 to derive the following relationship:

$$\theta_d = atan2(\dot{y}_d, \dot{x}_d)$$

Differentiating this with respect to time we get the following:

$$\dot{\theta}_d = \omega_d = \frac{d}{dt} atan\left(\frac{\dot{y}_d}{\dot{x}_d}\right) = \frac{\ddot{y}_d\dot{x}_d - \ddot{x}_d\dot{y}_d}{\dot{x}_d^2 + \dot{y}_d^2} \tag{5}$$

3.2 Feedback Control

To design our feedback control system we follow the method in [3] but adapt the model to apply to our differential drive vehicle. The first step is to define a state transform from our vehicle state to an error state space. We define two sources of error:

1. d - the distance between the vehicle and the closest point on the path, (x_d, y_d)
2. $\Delta\theta$ - the difference between our current heading, θ, and the tangent to the path at (x_d, y_d)

We can derive the state transform as follows. Let $\vec{r} = (x, y)^T$ be the current position of the robot. Let $\vec{r_d} = (x_d, y_d)^T$ be the point on the path closest to $\vec{r}$. Then:

$$e = \begin{bmatrix} d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \|\vec{r} - \vec{r_d}\|_2 \\ \theta - \theta_d \end{bmatrix}$$

Note that $\vec{r}$ is parameterized by the arclength, s, of the path the robot is driving. Similarly, $\vec{r_d}$ is parameterized by the arclength, s_d, of the desired path. To define the dynamics of the error we must derive $\dot{e}$:

$$\dot{e} = \frac{d}{dt}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} v\sin\Delta\theta \\ \omega - \kappa_d\dot{s}_d \end{bmatrix} \tag{6}$$

Where κ_d is the curvature of the desired path, which relates to the normal and tangent via Frenet's Forumula: $\frac{d\vec{t}_d}{ds_d} = \kappa_d\vec{n}_d$. Equation 6 still has the term $\dot{s}_d$. We can define the value of this as follows:

$$\dot{s}_d = \frac{\cos\Delta\theta}{1 - d\kappa_d} \tag{7}$$

Because we are performing path following, and not trajectory following, we do not care about time. Thus, we wish to form our state space model relative to $\frac{d}{ds_d}() = \frac{d}{dt}() \cdot \frac{dt}{ds_d} = \frac{d}{dt}()\frac{1}{\dot{s}_d}$. Dividing $\dot{e}$ by $\dot{s}_d$ we get our final model of the error dynamics:

$$\frac{d}{ds_d}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \tan\Delta\theta(1 - d\kappa_d) \\ \omega\frac{1-d\kappa_d}{\cos\Delta\theta} - \kappa_d \end{bmatrix} \tag{8}$$

We can assume our controller is performing accurately, and linearize this model about $d = 0$ and $\Delta\theta = 0$. This yields the following:

$$\frac{d}{ds_d}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\omega + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\kappa_d \tag{9}$$

This equation takes the form of a linear state space model: $\dot{e} = Ae + Bu$, where e is our error state. We use a standard control law for a linear system: $u = -Ke + c$. Letting $c = \kappa_d$ and substituting our control law in we get:

$$\frac{d}{ds_d}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_0 & -k_1 \end{bmatrix}\begin{bmatrix} d \\ \Delta\theta \end{bmatrix} \tag{10}$$

Which is stable when $k_0 > 0$ and $k_1 > 0$. Our control law is then:

$$\omega = -k_0 d - k_1\Delta\theta + \kappa_d \tag{11}$$

We note that this gives us a value for only ω. We also have control of the velocity v. This model assumes $v = 1$. To incorporate our feedforward parameters we scale this control by v_{ff} giving us our final combined control law:

$$\begin{gathered} v = v_{ff} \\ \omega = (-k_0 d - k_1\Delta\theta + \kappa_d)v_{ff} = (-k_0 d - k_1\Delta\theta)v_{ff} + \omega_{ff} \end{gathered} \tag{12}$$

3.3 Path Projection

In order to steer the robot onto the desired path, we must choose a point along the path that we associate with the robot's current position. This point determines the feedforward position, velocity and acceleration. In most situations, this can be chosen as the point on the curve that is closest to the robot's current position. However in certain cases where there are either loops in the path or significant disturbances near tight corners, this approach risks jumping between disconnected path segments, as in Figure 2. To address this issue we limit the search of the closest position to a small region centered on the last projected position, and choose a region size that balances the robot's speed with the curvature of the path.

4 Experiments and Results

4.1 System Identification

The parameters of the plant used in our control system were estimated by performing a series of trials in which we recorded the real-world linear and angular velocities relative to specified reference velocities. MATLAB's System Identification Toolbox was

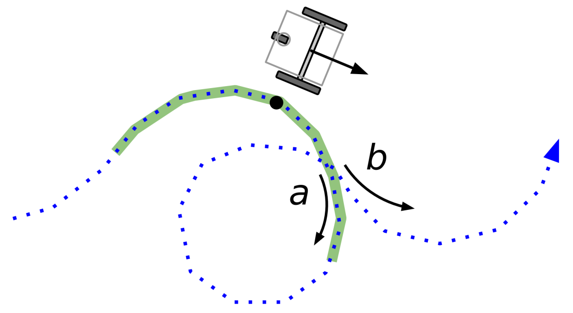

Figure 2: Searching over a limited region of the path (green) ensures that the correct path *a* will be taken rather than the incorrect path *b*.

employed to best characterize the MIMO system's behaviour, using 30 trials of recorded experimental data that was denoised using a low-pass filter. The model that best approximated the system was four second-order discrete transfer functions with a time delay. The transfer function model was then translated into a state space equivalent implemented as a ROS Python node using a second-order Padé approximation for the time delay. Including this in the simulation process gave us the ability to more accurately estimate the system's reaction to changes in controller gains and allowed access to more effective automated tuning processes.

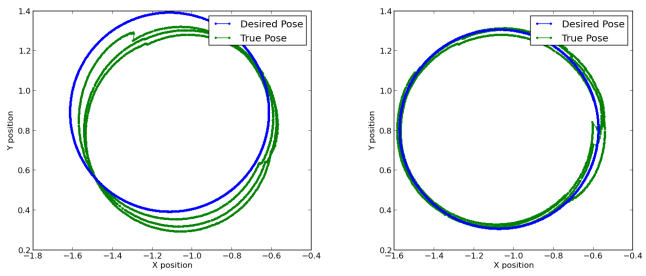

Figure 3: Plot comparison between the feedforward alone (left) and the the controller with feedback (right).

4.2 Trajectory Following

We tested our implementation of the feedback controller on ten trajectories of varying length and complexity. Figure 3 and Figure 4 show the results from a single trial. As can be seen, the controller significantly improved the robot's ability to follow desired paths. The maximum deviations from the path, expressed in terms of distance and heading error, decreased to approximately 26% and 40% of the original error, respectively.

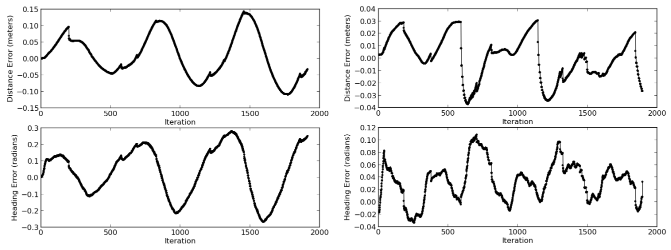

Figure 4: Error plot comparison between the feedforward alone (left) and the controller with feedback (right). The plots show Herb's pose with respect to a given trajectory while following a reference circular trajectory.

5 Discussion

Our feedback control system successfully improved the robots ability to navigate an environment. The planner generated feasible trajectories and the controller effectively steered the robot along the paths, even when the robot was not in an optimal starting pose. In particular, a significant decrease in position and heading error was observed with the introduction of the new controller.

6 Supplementary Material

The following videos are examples of the controller running on the HERB robot.
https://www.youtube.com/watch?v=h1r_KtExAe0
https://www.youtube.com/watch?v=Fqx3rZh6Yu8
https://www.youtube.com/watch?v=cfYZX27Ki60

References

[1] M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[2] G. Oriolo, A. De Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: design, implementation, and experimental validation. *Control Systems Technology, IEEE Transactions on*, 10(6):835–852, 2002.

[3] M. Werling and L. Groll. Low-level controllers realizing high-level decisions in an autonomous vehicle. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1113–1118. IEEE, 2008.