Optimal Path Following of a Differential Drive Mobile Robot

Riley Bridges and Ali Ryckman

***Abstract*— To control a differential drive mobile robot driving along a path, we have developed both open and closed loop optimal control solutions that minimize traversal time and maintain proximity to the path. Direct collocation was used for an open loop solution which resulted in a reasonably high accuracy, but is unlikely to be reproducible in the real world. Model Predictive Control (MPC) was used as a closed loop solution and resulted in relatively high deviation from the desired path as well as higher traversal times.**

I. Introduction

Differential drive is a type of drive train that involves a number of wheels arranged in parallel and fixed locations along each side of a vehicle, with the sets of wheels on each side of the vehicle being driven at different speeds to achieve both linear and angular motion. Mobile robots driven by differential drive systems are popular for use in various mobile robotics applications, largely due to their mechanical simplicity. Path following is a common task for mobile robots to perform, and can be difficult to carry out accurately while maintaining a reasonable speed. We propose to control a two wheeled differential drive mobile robot using optimal control techniques.

II. Simulation

The dynamics of this differential drive robot will be simulated in MATLAB using the following equations governing the system's motion, provided by Künhe et al. [1]:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \end{cases}$$

where x, y are the x and y positions of the robot, θ is the heading of the robot, and v refers to linear velocity of the robot. The system will have state vector $q(t) = \begin{bmatrix} x & y & v & \theta & \dot{\theta} \end{bmatrix}^T$.

III. Control Inputs

The system is actuated by motors on each of the two wheels. Accordingly, these actuators are commanded by a control vector $\mathrm{U} = \begin{bmatrix} \tau_L & \tau_R \end{bmatrix}^T$ containing the motor torques of the left and right wheels of the robot. This control vector can be related to the linear and angular accelerations used in the dynamics equations and state vector via the following kinematics provided by Censi and Tani [2]:

$$\begin{cases} \dot{v} = c\dot{\theta}^2 + \frac{\tau_L + \tau_R}{MR} \\ \ddot{\theta} = \frac{L(\tau_R - \tau_L)}{R(Mc^2+J)} - \frac{Mcv\dot{\theta}}{Mc^2+J} \end{cases}$$

Where R is the radius of each wheel, L is the distance between the two wheels, M is the mass of the robot, c is the displacement of the robot's center of mass from the origin of its base frame, and J is the robot's yaw moment of inertia with respect to the center of mass.

IV. Problem Description

Suppose we have a mobile robot with a two-wheeled differential drive system, which drives along a 2D plane. Given a 2D path consisting of the linear interpolation of a series of (x, y) positions, we want to find an optimal sequence of control inputs that causes this robot to traverse the path with minimal deviation in the shortest time possible.

A. Objectives and Constraints

Our cost function will be the time taken to fully traverse the path, and our constraints will be the following:

1) Initial and final positions of the robot are at the beginning and end of the path, respectively.
2) Initial and final velocities are zero.
3) Maximum distance of the robot from the path is below a certain threshold.
4) The average deviation from the line, measured as the line integral along the path of distance to the robot, divided by the length of the path, is below a certain threshold.
5) Commanded torques of each wheel are within the bounds of a certain set of torque limits.

We do not plan to enforce any computational or run time requirements on our solutions.

V. Proposed Solutions

Since we each have to develop our own optimal control, one of us will develop a closed loop feedback controller to solve this problem, and the other will solve the problem using an open loop trajectory optimization solution.

For our open loop optimal control, we plan to use either multiple shooting or direct collocation to optimize our trajectory. For our feedback controller, we are not yet sure what type of controller will be ideal for this problem, but based on background research it seems like MPC (Model Predictive Control) may be a viable option.

VI. Dynamics Simulation

Given the system's state vector and the kinematics, the dynamics function is defined as follows:

$$\dot{q}(t) = \begin{bmatrix} \dot{x} & \dot{y} & \dot{v} & \dot{\theta} & \ddot{\theta} \end{bmatrix}^T = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ c\dot{\theta}^2 + \frac{\tau_L+\tau_R}{MR} \\ \ddot{\theta} \\ \frac{L(\tau_R-\tau_L)}{R(Mc^2+J)} - \frac{Mcv\dot{\theta}}{Mc^2+J} \end{bmatrix}$$

To determine the validity of the dynamics equations used for the differential drive mobile robot, a simulation is necessary. Using MATLAB's ode45 numerical integration function we can determine the resulting kinematics of the robot system given left and right wheel torque inputs and the constant parameters mentioned above, such as robot mass and wheel radius.

For a simple baseline example, the control input is given as an eight-point torque input spline, where the left wheel torque input is a constant 1.0 Nm and the right wheel torque input is -1.0 Nm, as shown in Figure 1.

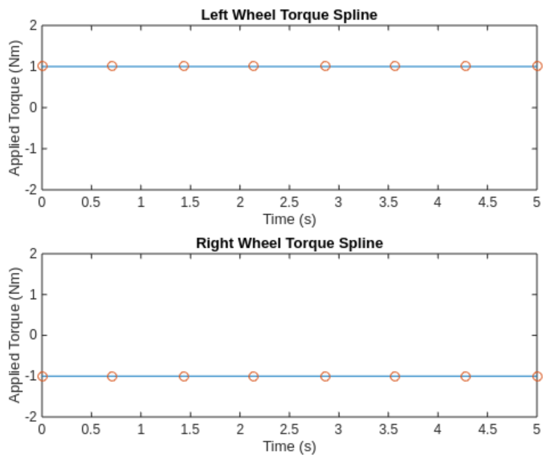


Fig. 1. Left and Right Wheel Torque Splines for Spinning in Place

The mass of the robot is set to 5 kg, the radius of each wheel is 0.1 m, the distance between wheels is 0.5 meters, and the center of mass is located at the origin of the robot's base frame, eliminating the need for a defined robot yaw moment of inertia. The robot's initial state $q(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. As the robot's center of mass is in the center of the robot and the wheels have equal and opposite force inputs, the expected robot behavior would be rotation in place with a constant acceleration and thus linearly increasing velocity. Some of the resulting state outputs are displayed in Figure 2, graphed versus time.

As expected, x position and y position do not change, but the robot has linearly decreasing angular velocity, meaning the robot is spinning in place. This is further supported by implementing an animation that draws the frame-by-frame visualization of robot position over time, which is attached

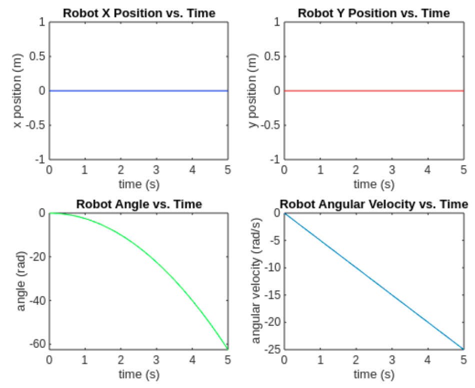


Fig. 2. State Outputs vs. Time for Torque Inputs in Fig. 1

to this report.

When providing an 8-point 1.0 Nm positive force spline to both wheels as shown in Figure 3, the resulting dynamics are also as expected. In this simulation, the robot's constant parameters are the same as above and the initial state is $q(0) = \begin{bmatrix} 0 & 0 & 0 & \frac{\pi}{4} & 0 \end{bmatrix}^T$. Because the wheels are controlled with equivalent torque, it is logical that the robot would drive directly forward with no angular velocity and with linear acceleration in the direction it is initially positioned. An animation for these dynamics can also be found attached to this report.

VII. Model Analysis & Limitations

When initially considering the dynamics of this system, we had the control vector $\text{U} = \begin{bmatrix} v_L & v_R \end{bmatrix}^T$ such that the control inputs were wheel velocities instead of torques. However, this would have been a less realistic model, as robots are not controlled directly by velocities. Using wheel torques results in more complicated dynamics but a more realistic system. Even so, there are still a handful of key simplifications of the system. For instance, there is not friction force present in the system which is sufficient for ideal simulations but would not model real-world differential drive robots very realistically. Additionally, the simulation examples used above simplified the robot's structure by making its center of mass the origin of the base frame, thus eliminating any need for a moment of inertia calculation. This moment of inertia is nontrivial to calculate and must be taken into consideration for any realistic robot whose mass is not perfectly centered.

VIII. Direct Collocation Solution

Our first optimal control solution utilizes direct collocation to optimize the trajectory of our system. Our solution finds the optimal design vector X^* that minimizes our cost function:

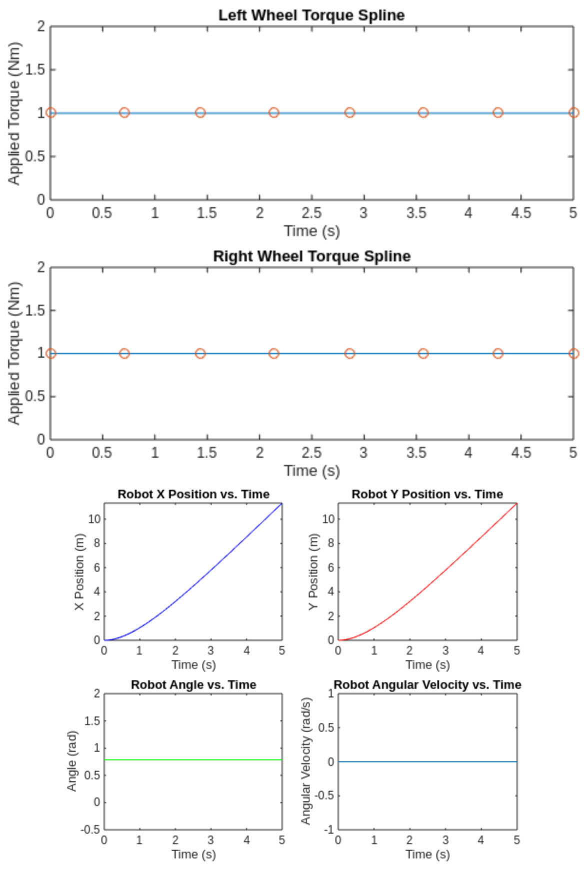


Fig. 3. Torque Splines and State Outputs for Equal Wheel Torques

$$\mathrm{X} = \begin{bmatrix} \tau_L & \tau_R & x & y & v & \theta & \dot{\theta} \end{bmatrix}$$
$$\mathrm{X}^* = \arg\min_{\mathrm{X}} C_1(\mathrm{X})$$
$$C_1 = \sum_{i=1}^{N} d_i + k t_f$$

where N is the number of nodes, d_i is the shortest distance of each node from the given path, t_f is the time taken to traverse the path, and k is a proportional tuning constant. The first term of this cost function prioritizes proposed paths that are closer to the true path, and the second term prioritizes paths that take a shorter time to traverse. Our inequality constraints are

$$\begin{cases} -\tau_{max} \leq \tau_L \leq \tau_{max} \\ -\tau_{max} \leq \tau_R \leq \tau_{max} \\ t_f \geq 0 \end{cases}$$

where τ_{max} is the maximum torque limit. Our equality constraints are

$$\begin{cases} q_0 = q_{0target} \\ q_f = q_{ftarget} \\ q_{i+1} = q_i + \dot{q}_i \cdot \Delta t \end{cases}$$

where q_0 and q_f are the initial and final states of the solution and $q_{0target}$ and $q_{ftarget}$ are their desired values, respectively. The last constraint refers to the defect constraints where $1 \leq i \leq N$. These constraints ensure the solution obeys the differential drive dynamics between each state, approximated by forward Euler integration. In our original direct collocation solution, we intended to have the robot's distance from the desired path be constrained using inequality constraints. The cost function would then be $C_2 = t_f$. However, simply constraining the dynamics to remain within a set distance of the provided path made it very difficult for the optimizer to converge on a correct path. By moving this distance target from the constraints to the cost function, we made it easier for the optimizer to find a reasonable solution that both stayed close to the path and traversed it.

To demonstrate the direct collocation solution, we use the cost function C_1 that considers both distance from path and completion time, with $N = 30$, $k = 0.1$, and the equality and inequality constraints with $_{max} = 0.5Nm$, $q_{0target} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, and $q_{ftarget} = \begin{bmatrix} 10 & 5 & 0 & 0 & 0 \end{bmatrix}$. The path the robot must follow is composed of points $q_{0target}$, (5,0), (5,5), and $q_{ftarget}$. The solution route taken by the robot for this problem is shown in Figure 4, with the path shown in red and the robot's actual trajectory shown in blue. The optimization takes an average of 30.25 seconds.

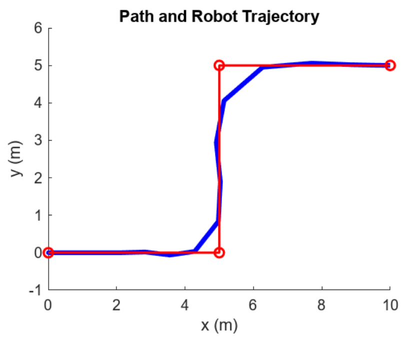


Fig. 4.

The resulting wheel torques and state outputs are shown in Figure 5.

For a second example, we used the same cost function but initial state of $q_{0target} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, final state of $q_{ftarget} = \begin{bmatrix} 5 & 10 & 0 & 0 & 0 \end{bmatrix}$, and path points $q_{0target}$, (5,2), (0,4) (5,6), (0,8), and $q_{ftarget}$. The resulting path and robot trajectory are shown in Figure 6.

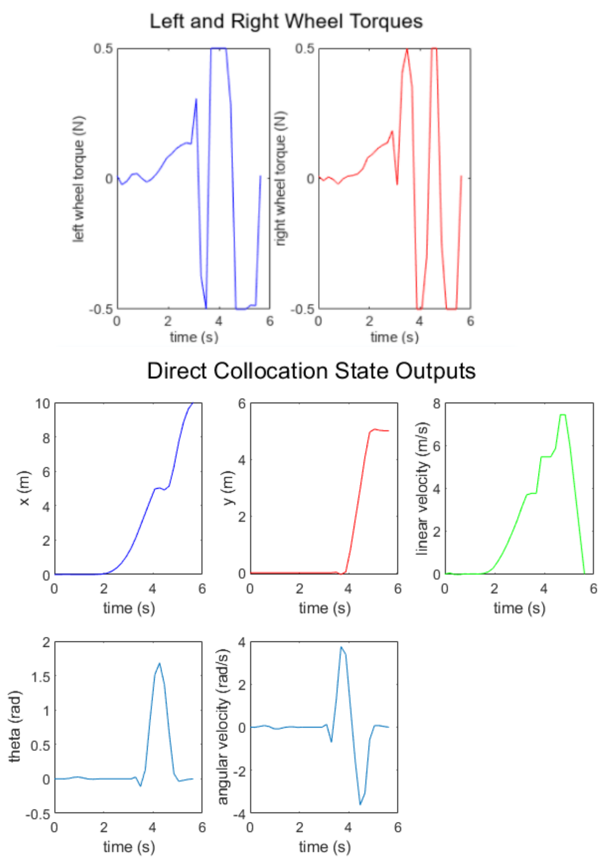


Fig. 5.

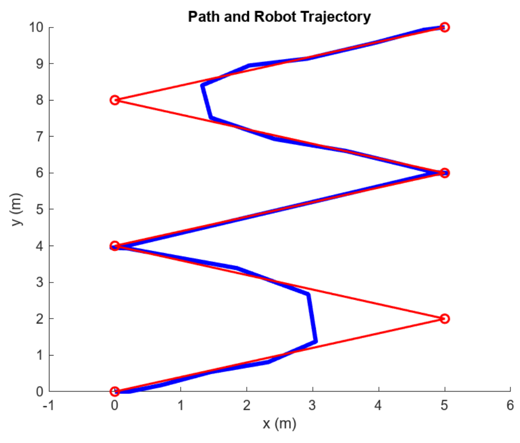


Fig. 6.

The left and right wheel torques and state outputs are shown in Figure 7.

Though the controller is able to start and end with the states specified by $q_{0target}$ and $q_{ftarget}$, it is unable to reach the waypoints in the middle of the path without additional modification of the cost function. This is a disadvantage of putting the distance to path requirement in the cost function instead of the constraints, because the controller must balance the cost of staying close to the path versus completing the solution in a timely manner.

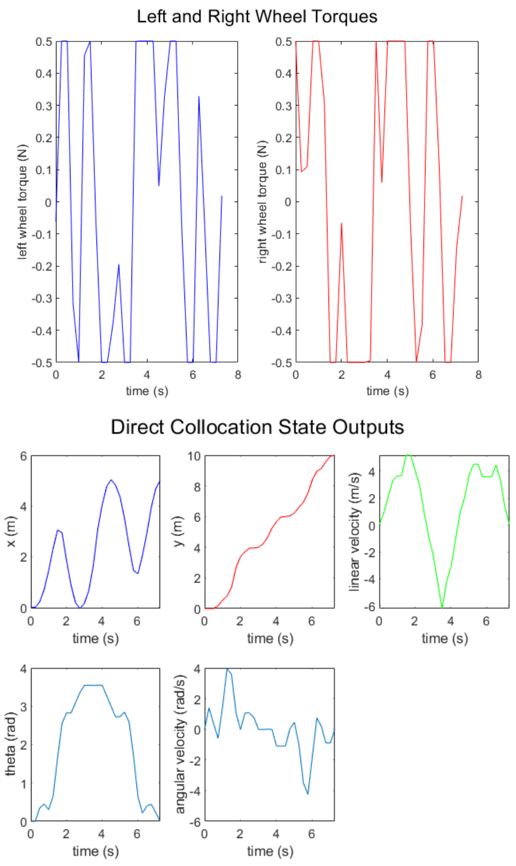


Fig. 7.

IX. Model Predictive Control Solution

Our second optimal control solution utilizes model predictive control. This involves optimizing a short time horizon trajectory, executing a portion of it, and using feedback to re-optimize the short term trajectory. In order to reduce the run time of the trajectory optimization, we linearize our dynamics about a different operating point, (q_0, u_0), during each trajectory optimization, by taking the Jacobian of our dynamics equation:

$$\begin{aligned}
\dot{q} &= h(q,u) \\
A &= \frac{\partial h}{\partial q}_{q0} \\
B &= \frac{\partial h}{\partial u}_{u0} \\
\dot{q}_{lin} &= A \cdot (q - q_0) + B \cdot (u - u_0)
\end{aligned}$$

These A and B matrices can be used for our trajectory defect constraints, allowing us to formulate the trajectory optimization as a convex quadratic program, meaning it can be solved very quickly. We used the same design vector as in the direct collocation solution. Our trajectory optimization finds the optimal X star that minimizes our cost function like so:

$$X^* = \arg\min_{X} \sum_{i=1}^{N} q_i^T Q q_i \Delta T$$

Subject to the same torque limits and defect constraints used in the direct collocation solution. We also enforce an initial condition equal to the current state during each iteration of MPC.

Our cost function would have an addition term if our R matrix was not set to zero. To maintain accuracy of our linearized dynamics, we re-linearize before the model prediction step in each iteration of the control loop, using the current state as the operating point. To make the robot follow the desired path, the ideal solution would incorporate the distance between the robot and the path into the cost function. However, this minimum distance is not a closed form function of the inputs, so it can not be used in a purely quadratic cost function required by a quadratic program. Based on our testing, using a nonlinear solver such as fmincon would cause the trajectory optimizer to take an unrealistically long time to be used for feedback control in MPC. Our solution to this was to divide up our path into line segments and iterate through each segment setting the end point of the segment as the desired state of our optimizer. Once each end point is reached, we continue to the next segment until we reach the end of the path.

For an example of MPC controller, we use the same path points pictured in Figure 4. Our time horizon and time step are 1 second, our total time is 100 seconds, and our number of nodes is 20. The trajectory formulated by MPC is shown in Figure 7, and the corresponding wheel torques and state outputs can be found in Figure 8. The trajectory optimization in each iteration of MPC took an average of 0.0148 seconds.

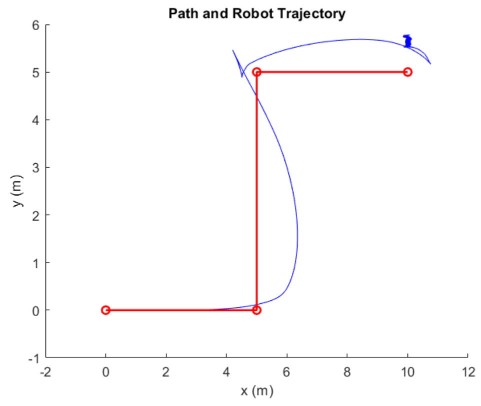
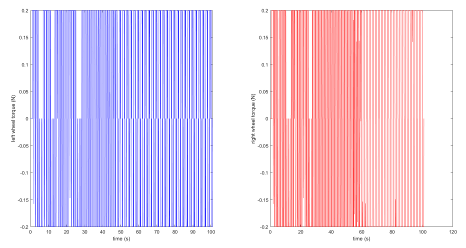


Fig. 8.

In another example of the MPC controller, we used the path points shown in Figure 6.



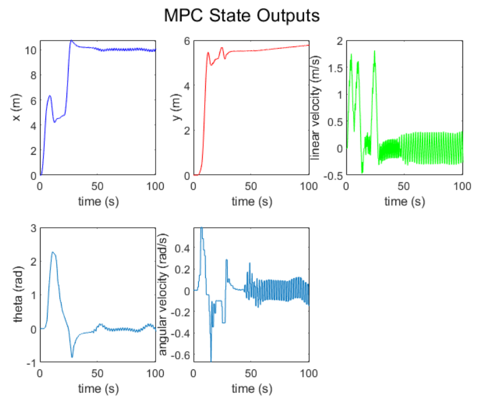


Fig. 9.

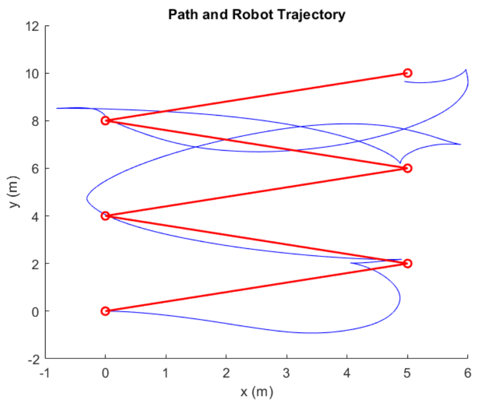


Fig. 10.

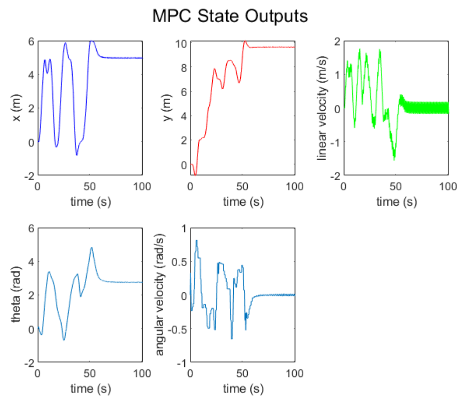


Fig. 11.

X. Conclusions

As can be seen in Figures 5 and 7, our MPC solution was not able to stay as close to the path as our direct collocation solution and took a much longer time to reach the end of the path. One key insight that makes these two cases difficult to compare is that the direct collocation solution never simulates the trajectory whereas the MPC solution is continuously using forward integration to simulate the dynamics. This means that while the direct collocation solution may do well in theory, it is unlikely to perform well in the real world with no feedback control. Furthermore, the direct collocation solution was somewhat sensitive to the initial design vector, which had to be adjusted slightly to get optimization to converge on a reasonable solution.

There are several ways to improve the performance of our MPC solution. Firstly, we could interpolate additional points along our desired path so that each segment completed by the controller is shorter, thus making it stay closer to the desired path. Another way the performance could be improved would be through linearizing the dynamics at each node of the trajectory optimization instead of once per MPC iteration.

The inaccuracy of our linearized dynamical model used in our MPC solution suggests that the dynamics of the differential drive robot are highly nonlinear. This implies that either a nonlinear optimizer must be used with the true dynamics, or linearized dynamics can only be used at a high frequency of linearization. The discrepancy between the accuracy of our direct collocation planned trajectory and our simulated trajectory in MPC suggests that this control problem requires a high degree of feedback to get accurate results.

References

[1] Künhe, F., J. Gomes, and W. Fetter. ”Mobile robot trajectory tracking using model predictive control.” II IEEE latin-american robotics symposium. Vol. 51. 2005.

[2] A. Censi and J. Tani. Modeling of a Differential Drive Vehicle. [PDF document]. ETH Zurich, 2010.