Differential Drive Robot Motion

Motion Models for Differential Drive Mobile Robots

In any mobile robotics project knowing how to estimate your robot's position is fundamental for any application. Determining where your robot is within its environment requires some kind of model to estimate its position based on control inputs to the robot. Most commonly, we use the robot's linear velocity and rotational velocity as the inputs to the kinematic models. These velocities can come from a variety of sources, such as commanded velocities to the robot, inertial measurement units, or wheel encoders.

For this application, we'll consider one of the most common scenarios: tracking the 2D pose of a differential drive robot using wheel encoders.

The Basics

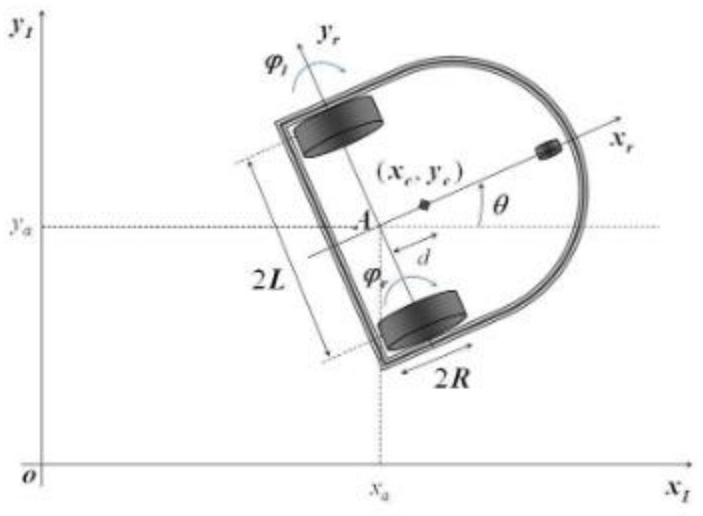
First, let’s take a moment to define some terms and give some basic background information you have probably seen before. But for the sake of completion and for readers who may be encountering this information for the first time, I will be rehashing the fundamentals here.

The first thing to define is the "global frame" or the "world frame". This is the coordinate system used to define the environment in which the robot exists. For this project we are considering a 2D robot, so it will exist in a 2D *x, y* global coordinate system.

We will define the 2D pose of the robot as the following:

$$p_r = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

The pose of the robot corresponds to its *x, y* position and theta orientation within the global frame. The robot is modeled as a single point with some orientation. The diagram below taken from [2] shows a differential drive robot within the world frame:



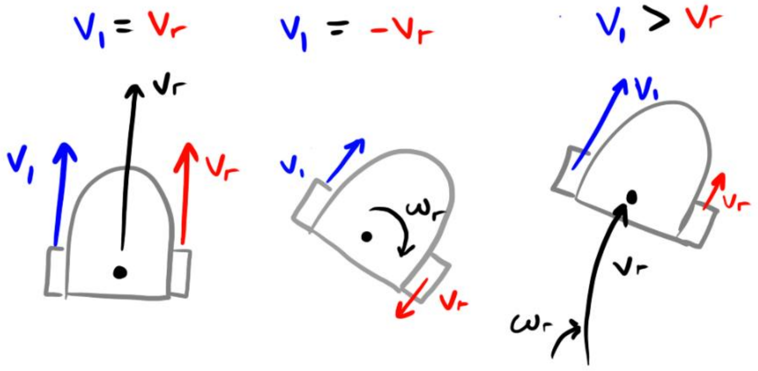
Where point A represents the position of the robot, theta denotes the orientation. The coordinate frame where the robot exists is its world/global frame.

The control input for the robot will be derived from its wheel encoders. The wheel encoders measure the rotations of the robot's wheels, which can be used to determine the velocity of each wheel. Typically, this value is produced in radians per second. We will want to transform this value so we get meters per second for each wheel:

$$v_{m/s} = v_{rad/s}\frac{d}{2\pi}$$

Where d is the wheel diameter. This equation gives us the wheel velocity in mps. This will need to be performed for each tire of the robot.

Once we have the linear velocity of the left and right wheel, we then need to consider the dynamics of a differential drive system to determine its linear and rotational velocity of the overall vehicle. For example, if both wheels are moving forward at the same velocity, the vehicle itself will only have a linear velocity directed along the robot's local coordinate frame's x axis. If each wheel moves at equal velocities in opposite directions, the robot will have a rotational velocity and no linear velocity as it will be moving in place. If the robot moves with both wheels with different, positive velocities, the robot will move with some curvature, resulting in a linear and rotational velocity.



To compute linear and rotational velocities, the following equations are used:

$$v_{m/s} = \frac{v_{r\ m/s} + v_{l\ m/s}}{2}$$

and

$$\omega_{rad/s} = \frac{v_{r\ m/s} - v_{l\ m/s}}{W}$$

where $v_{r\ m/s}$ and $v_{l\ m/s}$ are the right and left wheel velocities, respectively, and W is the track width of the robot. Trackwidth is the distance between each wheel. The point that defines the origin of the robot frame is the point halfway between each wheel.

The Forward Kinematic Model

To determine our robot's position after getting the linear and rotational velocities of the robot, we need a mathematical model representing the kinematics of the differential drive system. This model will allow us to input the velocities and the amount of time between readings to determine the robot's change in position. The simplest model for the kinematics of a differential drive robot is a linear model that simply performs the linear translation and rotation afterwards. This model begins by creating a rotation matrix based on the robot's current orientation.

$$R(\theta) = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix}$$

Where theta is the robot's current orientation. We will multiply this matrix by a vector containing our linear and rotational velocities.

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

By multiplying R by u, we get a 3x1 vector containing the x and y linear velocities in the world frame and the rotational velocities of the robot.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \omega \end{bmatrix}$$

by multiplying these by the change in time between iterations, δ_t, and adding it to the original pose, we get the updated pose of the robot.

The Forward Kinematic Model using ICR

While the previous model is simple, and when used at a very high sampling rate, it provides a "good enough" estimation of the robot's position. However, this model does not account for the curvature of the robot's motion as it only performs a linear motion THEN rotates. In reality, the robot is turning while making its linear motion. The model introduced here will account for this and provide a more accurate estimation of the robot's pose (assuming constant velocity).

If the robot is driving at a constant linear and rotational velocity, that motion can be described as a circle with some radius. The equation for an arbitrary objection moving on a circular trajectory with radius r can be described by:

$$v = \omega r$$

Rearranged, we get:

$$r = abs(\frac{v}{r})$$

Where v and omega are the robot's linear and rotational velocities. This gives us the radius of the circle described by the robot's motion. The point in the center of this circle is called the instantaneous center of rotation. "Instantaneous" as we are only considering the velocity of the robot at that current moment. The equations to find the x,y point that is the center of rotation are as follows:

$$x_c = x - \frac{v}{\omega} sin(\theta)$$

$$y_c = y - \frac{v}{\omega} cos(\theta)$$

Where xc and yc is the center point of the circle described by the robot's motion. The following equation computes the arclength of the robot's motion along the circle to find the new pose of the robot after delta t seconds of travel:

$$\begin{pmatrix} x_c - \frac{v}{\omega} sin(\theta + \omega \Delta t) \\ y_c + \frac{v}{\omega} cos(\theta + \omega \Delta t) \\ \theta + \omega \Delta t \end{pmatrix}$$

Which after substituting for xc and yc:

$$\begin{pmatrix} x - \frac{v}{\omega} sin(\theta) - \frac{v}{\omega} sin(\theta + \omega \Delta t) \\ y + \frac{v}{\omega} cos(\theta) - \frac{v}{\omega} cos(\theta + \omega \Delta t) \\ \theta + \omega \Delta t \end{pmatrix}$$

This equation gives us the robot's new pose while considering the curvature of the motion. However, this model has one caveat: pure linear motion. If the robot is driving on a straight line, the circle describing its motion has an infinite radius. So, we need to add a conditional statement to handle this. If the robot's angular velocity is sufficiently small, then we will assume it is following a straight line and we will use the following model:

$$\begin{pmatrix} x + v\Delta t cos(\theta + \omega \Delta t) \\ y + v\Delta t sin(\theta + \omega \Delta t) \\ \theta + \omega \Delta t \end{pmatrix}$$

Where we simply compute the linear motion of the robot and divide it into x and y components based on the robot's final orientation.

References

[1] http://www.cs.cmu.edu/~rasc/Download/AMRobots3.pdf

[2] https://www.researchgate.net/publication/271098735_Dynamic_Modelling_of_Differential-Drive_Mobile_Robots_using_Lagrange_and_Newton-Euler_Methodologies_A_Unified_Framework

[3] https://mitpress.mit.edu/books/probabilistic-robotics

[4] https://dspace.mit.edu/bitstream/handle/1721.1/36832/16-412JSpring2004/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring2004/A3C5517F-C092-4554-AA43-232DC74609B3/0/1Aslam_blas_report.pdf