Systematic Design of a Differential Drive Mobile Robot

Creed Zagrzebski
Department of Computer Science
University of Wisconsin – La Crosse
La Crosse, Wisconsin
zagrzebski1516@uwlax.edu

***Abstract*—A systematic overview of the design of a differential drive robots. Covers the kinematics, dynamics, and the control system of the robot. Includes a detail overview of forward and inverse kinematics, and the equations to represent the positioning of the robot. Discusses SLAM techniques for generating a map for autonomous navigation and path planning.**

***Index Terms*—differential drive, kinematics, encoders, SLAM, localization, kalman filter, particle filter, odometry**

I. Introduction

Mobile robotics is one of the most prominent fields in the robotic family due to its usefulness and applicability in several fields like military defense, industry, and other commercialized applications [1]. There are many different applications for mobile robots that can be used to reduce human exposure to dangerous tasks and increase efficiency. A few popular applications for mobile robotics include the inspection dangerous of equipment, material handling in a computer-integrated manufacturing environment, and underwater exploration [2].

The differential drive is the simplest and most effective drivetrain for a ground-contact mobile robot [2]. The drivetrain for these robots is often low-cost and has a wide range of applications in both home and industrial settings for uses such as warehouse automation and home cleaning. This type of robot consists of two separate wheels that lie on a common axis [2]. Each wheel has its own independent motor, allowing for precise control of the robot [4]. Controlling the velocities of each wheel determines the robot's motion and is used to adjust the speed and direction of the robot's trajectory. By setting the speed of each wheel opposite to one another, the robot will spin in place. If the velocities of each wheel are equal, then the robot will move in a straight line. If one of the wheels has a greater velocity than the other, then the robot will move either left or right [2]. These robots are sometimes equipped with additional free-rolling wheels for stability but do not contribute to the steering of the robot and can be ignored in calculations [2].

Designing an autonomous differential drive robot involves several steps, including odometry using hall effect sensors, forward and inverse kinematics, path planning, and SLAM-based map generation [2,4,8]. This article will review how these components work together with one another to make a fully functional autonomous navigation system for a differential drive robot.

II. Odometry and Forward Kinematics

Odometry is a technique to determine a robot's current position within its global environment, based on its previously known positions [1]. If the robot's parameters (i.e., wheel size, the distance between wheels, acceleration, and velocity) are known, then the robot's current position can be calculated using kinematics. Kinematics is the study of the mathematics behind the motion of an object without considering the effects of forces that can affect the object's motion (i.e., friction) [2]. Namely, kinematics is more concerned with the geometric relationships that control the system, including energies and speeds (i.e., linear/angular velocity, acceleration) that affect the movement and position of a system. Dynamics differs from kinematics since it focuses more on the study of motion with the concern of external forces, and will be covered later in the article since it is important to consider when calculating the trajectory error of the robot for factors such as wheel slippage [2].

Using the kinematics of a differential drive robot, the velocities of the left and right wheels can be controlled to change the position and heading of the robot. Forward kinematics takes the velocities of each wheel as control parameters and determines the new position of the robot with respect to its previous location using mathematical models [2]. By specifying the proper velocities of each wheel, the robot can move from its current position (x, y, θ) to a goal position located in the global environment $(x'.y', \theta')$, where x and y are the cartesian coordinates for the center of the robot and θ is the angle of the robot with respect to the x-axis of the global reference frame.

Fig. 1 shows the cartesian coordinates and geometry for a differential drive robot. This diagram provides a geometric description of the robot where the ICR is the center of the arc for which the robot is turning about, R is the signed distance from the instantaneous center of rotation (ICR) to the center of the robot, ω is the angular velocity about the ICR, L is the distance between the center of both wheels along the axle, θ is the heading of the robot, and V_L and V_R are the velocities of the left and right wheel respectively [2].

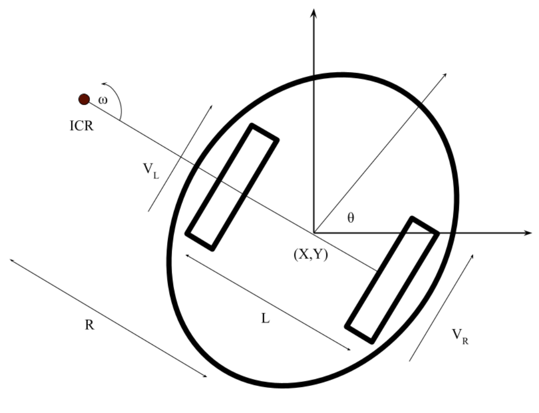

Fig. 1. Geometric Description of Differential Drive Robot.

To calculate the current position of the robot using kinematics, the velocities of the left and right wheels are required [1]. The wheel velocities can be found by first calculating the angular velocity of the driveshaft of the DC motor. The DC motors contain hall effect sensors (magnetic) to count the number of pulses per revolution of the wheel [1]. These pulses can trigger hardware interrupts on a microcontroller to increment a variable for each pulse. The angular velocity of the motor shaft can be derived by looking at the variable storing the number of pulses within a specific time interval and dividing that value by the encoder resolution (i.e., pulses per revolution) [1].

However, the calculations for forward kinematics require linear velocity, since the distance traveled is needed to find the robot's new position with respect to its initial position. Linear velocity is obtained from the angular velocity (RPM) with the circumference of the wheel using the following equation

$$\text{Velocity } = \frac{(2\pi * \text{wheel radius}) * \text{RPM}}{60} \text{ [1]}$$

At any instantaneous point in time, the robot must satisfy the property that the left and right wheels follow a path that moves around the ICR at a fixed angular rate ω, and thus gives the following mathematical properties using the left and right velocities of the wheels [2]:

$$\omega(R + L/2) = V_R \tag{1}$$

$$\omega(R - L/2) = V_L \tag{2}$$

It is important to note that ω, R, V_L, and V_R are all functions of time. At any point in time, R and w can be solved using algebra, yielding [2]:

$$R = \frac{L}{2}\frac{V_L + V_R}{V_R - V_L} \tag{3}$$

$$\omega = \frac{V_R - V_L}{L} \tag{4}$$

Suppose there is a differential drive robot located at some initial position (x, y, θ). The equation of the Instantaneous Center of Rotation (ICR) can be derived using (2), the current position of the robot, along with trigonometry [2]:

$$\begin{aligned} ICR &= (x - Rsin(\theta + \pi/2), y + Rcos(\theta + \pi/2)) \\ &= (x - Rsin(\theta), y + Rcos(\theta)) \end{aligned} \tag{5}$$

At some instantaneous time $t \to t + \delta t$, the future position and heading of the robot can be modeled using linear algebra, trigonometry, and the R_z transformation matrix from the special orthogonal group [2]:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICR_x \\ y - ICR_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICR_x \\ ICR_y \\ \omega\delta t \end{bmatrix} \tag{6}$$

This calculation can be thought of as translating the ICR to the origin of the global reference frame, performing a yaw rotation about the z-axis using the orthogonal rotation matrix $R_z \in SO(3)$, and then translating the rotation back to the ICR. This equation represents the motion of the robot rotating at a distance R from equation (3) about the ICR with an angular velocity of ω as the input from equation (4) [2]. This model can be used to find the final position of a differential drive robot given the wheel velocities with respect to the global reference frame. However, the objective of an autonomous robot is to automatically navigate to a goal position specified by the user. Forward kinematics can only calculate a goal pose given the left and right wheel velocities, so inverse kinematics is required to calculate the necessary wheel velocities to achieve a given goal position.

III. Inverse Kinematics

Inverse kinematics is the process of finding wheel velocities V_L and V_R to navigate to a target pose (x', y', θ') and is used for trajectory planning. Unfortunately, it is not possible to select an arbitrary target pose due to geometric constraints of the robot [2].

A differential drive robot is non-holonomic, in which the robot can not move laterally along its axis [3]. For example, a robot facing perpendicular to the x-axis must rotate before moving along the y-axis. The non-holonomic constraint is mathematically represented by:

$$\begin{bmatrix} -sin(\theta\delta t) & cos(\theta\delta t) \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix} = 0 \tag{7}$$

Consequently, simply inverting equation (6) to find the control inputs solutions (velocities) for each wheel is not always possible or desirable due to this constraint. Therefore, another method is needed to control the robot to navigate it to the desired position.

One possible option is to use a closed-loop kinematic controller to direct the robot's motion [4]. Using wheel encoders from the DC motor, the current position is compared with the desired position to determine the amount of error in the system (i.e., distance away from the desired position). The calculated error is used to control the linear and angular velocity of the robot to achieve the desired position within the global reference frame.

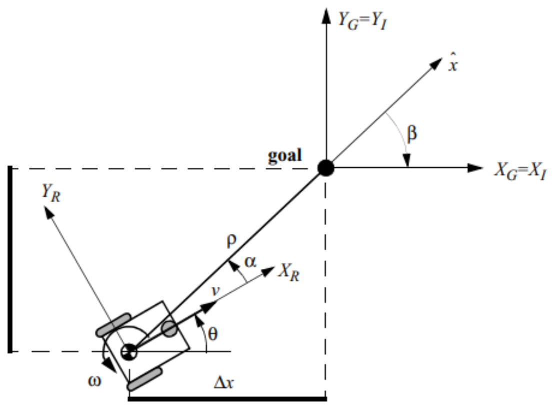

Fig. 2. Differential Drive Robot with respect to the global reference frame [4]

Another way to represent the kinematics of a differential drive robot with respect to the global reference frame is given by the following mathematical model and Fig. 2:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos(\theta) & 0\\ sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{8}$$

where $\dot{x}$ and $\dot{y}$ are the linear velocities in the direction of X_G and Y_G of the global reference frame. Let α be the angle between the x-axis of the robot's reference frame defined by X_R and Y_R and the vector $\hat{x}$ that connects the origin of the robot's reference frame to the goal position located in the world reference frame. β is the angle between $\hat{x}$ and the vector X_G of the global reference frame. ρ is the euclidean distance from the origin of the robot's reference frame to the goal position.

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\alpha = -\theta + atan2(\Delta y, \Delta x)$$
$$\beta = -\theta - \alpha$$

These equations represent the velocity of the robot in the x, y, and θ direction. This information can be used to describe the system in polar form. If the robot is facing towards the goal position, the system can be described in polar coordinates by [4]:

$$\begin{bmatrix}\dot{p}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-cos(\alpha) & 0\\ \frac{sin\alpha}{\rho} & -1\\ -\frac{sin\alpha}{\rho} & 0\end{bmatrix}$$

If the robot is facing away towards the goal position, then the system can be described in polar coordinates by [4]:

$$\begin{bmatrix}\dot{p}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}cos(\alpha) & 0\\ -\frac{sin\alpha}{\rho} & 1\\ \frac{sin\alpha}{\rho} & 0\end{bmatrix}$$

The polar kinematics above can then be used to generate a mathematical model for a proportional closed-loop dynamic control system [4]:

$$\begin{bmatrix}\dot{p}\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}-k_p\rho\cos(\alpha)\\ k_p - k_\alpha\alpha - k_\beta\beta\\ -k_p sin(\alpha)\end{bmatrix}$$

The k_p, k_α, and k_β are tunable parameters to fix understeering and oversteering of the robot [4]. These constants are similar to the K_P, K_I, and K_D terms used in PID control to decrease over-shooting, settling time, steady-state error, etc. Each row in the matrix represents it own proportional controller. The control system will look at the current position of the robot and the desired position, calculate the error, and adjust v and ω using the control law [4]:

$$v = k_p\rho \text{ and } \omega = k_\alpha\alpha + k_\beta\beta$$

The kinematic controller will generate the angular and linear velocities of the robot and use (1) and (2) to calculate the appropriate velocities for the left and right wheels.

Another option for controlling the robot's trajectory is to use an open-loop controller. The robot can either move straight or turn at a given time. However, the robot cannot spin or move forward concurrently [4]. The robot will first rotate in place by setting the velocities of both wheels opposite to one another until it faces the goal position. Once the robot is orientated towards the goal, it can simply drive forward until it reaches the goal destination by setting both wheel velocities to the same value. Finally, the robot can rotate in place until it reaches the desired pose [2].

The downside of the open-loop approach is the resulting trajectories are not smooth, and the robot cannot update its trajectory if the robot experiences wheel slippage (i.e, no traction) since an open-loop controller does not look at feedback [4]. The trajectory must also be pre-computed, which can be difficult since the limitations of the robot's velocity and acceleration can affect the validity of the calculations [4].

This section assumes no obstacles are within the global reference frame. However, a differential drive robot needs to know information about its environment to generate a path between the initial and goal position. To find out where the robot is permitted to travel, a kinematic controller needs input derived from a map using a path-planning algorithm. The following section will review techniques used to acquire a map of the environment that a path planning algorithm can utilize to create an optimal path.

IV. SLAM and Path Planning

A. The SLAM Problem

Odometry and kinematic calculations assume no obstacles are located within the global reference frame when navigating a desired path. In reality, the robot can be located in a highly dynamic industrial environment with many obstacles that move frequently. For example, a robot located in a warehouse environment needs to be aware of moving goods and forklifts.

As a result, the robot needs a way to generate a global path that avoids these obstacles so the robot can reach its goal position in the environment without interference from any obstructions. SLAM (Simultaneous Localization and Mapping) is a technique to acquire a map of an unknown environment while simultaneously localizing its position within the map [5]. SLAM is essential for autonomous navigation in a highly dynamic environment.

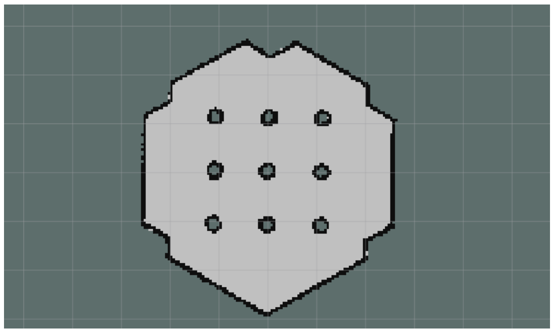

Fig. 3. Map generated using ROS gmapping package

A problem with SLAM is that the robot needs a way to position (localize) itself within the environment. To solve this problem, the robot must observe its environment to obtain features or landmarks from the unknown environment to use as reference geometry so the robot can create the map and keep track of its relative positioning simultaneously [7]. A few popular feature-based probabilistic SLAM algorithms are used in mobile robotics to solve the SLAM problem: Extended Kalman Filter (EKF-SLAM) and Rao-Blackwellized Particle Filter SLAM. However, a sensor is required to extract these features from the environment to use these algorithms.

Various types of sensors can be used with SLAM algorithms to gather information about the environment. A few classes of sensors for SLAM are 2D/3D LiDAR, visual (camera), and sonar. LiDAR is most common due to its reliability and robustness for indoor environments [6]. LiDAR works by emitting a collimated beam using a rotating transmitter which reflects off an object and goes to the receiver of the LiDAR sensor. The sensor finds the distance by using the time it took to receive the reflection of the beam [4]. This information is consumed by a SLAM algorithm (i.e., EKF-SLAM and RBPF-SLAM) to generate a map and localize the robot. The following sections will review how this information is useful for SLAM-based map generation.

B. EKF-SLAM Approach

EKF-SLAM is a mapping and localization probabilistic algorithm that uses an Extended Kalman filter, a modified version of the Kalman Filter for non-linear systems, to estimate the state or positioning of the robot. Kalman filters are commonly used for sensor or data fusion for robotic applications. Given several sources of information with a level of uncertainty, a Kalman filter improves the data by reducing the uncertainty (or error) and predicting the correct values [7]. The extended Kalman filter differs from the standard Kalman Filter since it uses Taylor Series Expansion to linearize sensor data to work with the Kalman algorithm [7]. Otherwise, the base algorithm is ineffective in filtering the non-linear sensor data. Using EKF with SLAM involves utilizing the Kalman filter to merge the distance assessment obtained from every feature into a corresponding object's position within the map [4]. There are five steps that take place in order to achieve EKF-based localization at time t [4]:

1) Position Prediction - Predict the position of the robot for time $t + 1$ based on its old location and the wheel odometry (from encoders)
2) Observation - Obtain measurements at $t+1$ from sensors (i.e., LiDAR) and extract features (i.e., line or door)
3) Measurement Prediction - Use the predicted robot position and the map information to generated a set of predicted features
4) Matching - Match the feature predictions in the set of predicted features with the set of observed features
5) Apply EKF - Use the EKF filter to compute the best estimate of the robot's position in the map based on the predicted position at time $t \to t + 1$ and the matching (validated) features

The output from this algorithm is the most likely position of the robot within the environment. This algorithm will continuously repeat during the SLAM process.

A limitation of this model is that it requires a lot of computational power and needs values from the sensors to follow a Gaussian Probability Distribution that minimizes the squared distance from the sensors [7]. The computational requirements make this algorithm not ideal for large maps with many features since the number of landmarks increases the size of the system, making it more computationally expensive. A second approach that reduces the size of the system is a Particle Filter along with a Extended Kalman Filter. This approach is called the Rao-Blackwellized Particle Filter.

C. Particle Filter Approach (Rao-Blackwellized)

An alternative to the EKF-SLAM is the Rao-Blackwellized Particle Filter SLAM (RBPF-SLAM) [2]. The issue with the EKF-SLAM approach is that it requires a probabilistic model that obeys the Gauss distribution. In reality, most robotic systems are non-linear, and the probabilistic model follows a non-Gaussian distribution [8]. With RBPF-SLAM, we do not need a system to follow a Gaussian Mode, and the sensor data does not need to be linear. RBPF-SLAM is based on the Monte Carlo Particle Filter but uses fewer estimations to determine the location of the robot, and thus is a popular choice [2]. The RBPF-SLAM algorithm splits the SLAM problem into two parts: Path Estimation and Feature Estimation [8]. The RBPF-SLAM algorithms use a set of particles, where each particle represents a possible configuration of the robot (Path Estimation) and map (Feature). Each particle has a weight associated with it that represents the likelihood of that particle containing the actual state of the robot and map [8]. As the robot moves throughout the environment, these weights are updated to reflect the potential states of the map and the position of the robot. Fig 4. shows a robot and its potential states as the robot moves. Notice how the estimations converge as the algorithm becomes more confident with the environment.

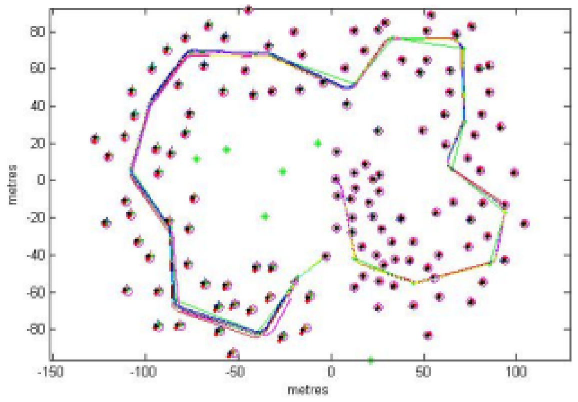

Fig. 4. RBPF-SLAM Particle Estimations [8]

D. Path Planning using A* with SLAM Map

For the robot to move autonomously using the environment data provided by SLAM, an algorithm is needed to find the shortest path from the current position of the robot to the goal. Path planning is used for autonomous exploration of the robot's surroundings by finding a sequence of waypoints in the environment that will lead the robot from an initial position to a goal position with the least amount of work [5]. A* is a graph search algorithm that will find the optimal path from the start to the goal [2]. The search looks at all of the possible paths for the robot and returns the shortest path to the goal node [5]. This algorithm is similar to Dijkstra's Shortest Path algorithm but only returns the required nodes for a single shortest path rather than a tree containing the shortest distance to every node. The path generated by the A* algorithm should minimized the total cost $f(n)$ given by:

$$f(n) = g(n) + h(n) \text{ [5]}$$

where n is the last node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is the cost from node n to the goal node. Only nodes that are not identified as an obstacle will be used in the A* algorithm for finding the optimal path. This algorithm will return a tree of nodes containing the optimal path from the start to the end goal [5]. Using inverse kinematic controller, the robot can navigate to each of the nodes in the tree, with each node representing (x, y, θ), until it reaches the final position (x', y', θ').

V. Conclusion

In conclusion, this paper reviewed the basic requirements for designing a two-wheel differential drive robot, including kinematics, dynamic control system, map acquisition using SLAM, motor control with encoders, and path planning using the A* search algorithm. Forward and inverse kinematics use mathematical models to control the overall linear and angular velocity of the robot to move the robot into a goal position using motor controllers and an open-loop or closed-loop kinematic controller. Simultaneous localization and mapping (SLAM) gather information about the environment using sensors with a landmark probabilistic algorithm such as EKF-SLAM or RBPF-SLAM to localize the robot. All of these systems work together to create a fully autonomous differential drive robot.

References

[1] R. Singh, G. Singh and V. Kumar, "Control of closed-loop differential drive mobile robot using forward and reverse Kinematics," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2020, pp. 430-433, doi: 10.1109/ICSSIT48917.2020.9214176.

[2] G. Dudek, Computational principles of mobile robotics. New York: Cambridge University Press, 2011.

[3] R. L. S. Sousa, M. D. do Nascimento Forte, F. G. Nogueira and B. C. Torrico, "Trajectory tracking control of a nonhol onomic mobile robot with differential drive," 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires, Argentina, 2016, pp. 1-6, doi: 10.1109/ARGENCON.2016.7585356.

[4] R. Y. Siegwart and I. R. Nourbakhsh, Introduction to autonomous mobile robots. Cambridge, Mass: MIT, 2004.

[5] L. d. S. Pinto, L. E. S. A. Filho, L. Mariga, C. L. N. Júnior and W. C. Cunha, "EKF-SLAM with Autonomous Exploration using a Low Cost Robot," 2021 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, 2021, pp. 1-7, doi: 10.1109/SysCon48628.2021.9447073.

[6] H. Zhu, P. Zhao, S. Liu and X. Ke, "A SLAM Based Navigation System for the Indoor Embedded Control Mobile Robot," 2020 4th International Conference on Robotics and Automation Sciences (ICRAS), Wuhan, China, 2020, pp. 22-27, doi: 10.1109/ICRAS49812.2020.9134926.

[7] F. Hidalgo and T. Bräunl, "Review of underwater SLAM techniques," 2015 6th International Conference on Automation, Robotics and Applications (ICARA), Queenstown, New Zealand, 2015, pp. 306-311, doi: 10.1109/ICARA.2015.7081165.

[8] F. Zhang, S. Li, S. Yuan, E. Sun and L. Zhao, "Algorithms analysis of mobile robot SLAM based on Kalman and particle filter," 2017 9th International Conference on Modelling, Identification and Control (ICMIC), Kunming, China, 2017, pp. 1050-1055, doi: 10.1109/ICMIC.2017.8321612.