Odometry

Introduction

The first step is to understand the mobile robot odometry and to review some helpful concepts. In a robotic environment the usage of internal sensors to estimate the robot pose is called odometry. The classical technique for a wheeled robot to calculate its position is to track its location through a series of measurements of the rotations of the robot's wheels. Odometry requires a method for accurately counting the rotation of the robot and, with this information, estimates the left and right wheel velocities. Therefore, relative position estimation is extremely dependent on the measurement of the robot's velocity. Once we have our left and right spinning velocities, we can compute the kinematics and obtain our robot pose integrating(x', y', θ').

Suppose your robot starts at the origin, pointed down the x-axis. Its state is (x, y, θ) = (0, 0, 0). If the robot travels (roughly) straight for three seconds at 1 m/s, a good guess of your state will be (3, 0, 0). That's the essence of odometry.

We'll assume that the vehicle is differentially driven: it has a motor on the side of the robot and another motor on the right side. If both motors rotate forward, the robot goes (roughly) straight. If the right motor turns faster than the left motor, the robot will move left (we will explain this point in details next chapters).

So, our goal is to measure how fast our left and right motors are turning. From this, we can measure our velocity and rate of turn, and then integrate these quantities to obtain our position.

Derivation

The speeds of our motors give us two quantities: the rate at which the vehicle is turning, and the rate at which the vehicle is moving forward. All we must do is to integrate these two quantities and we'll have our robots state (x, y, θ).

That sounds a bit scary, but the mathematics end up being very simple. If we had analytic expression for the angular rate and forward rate functions, their integral probably would be scary. But in a real system, our data comes from real sensors that sampled periodically. Every few milliseconds, we get a new

measurement of our motor velocities. We will numerically integrate our solution, which is of course, just a fancy way of saying that we'll divide time up into little chunks and just add up all little pieces.

Suppose our robot is at(x, y, θ). Depending on what kind of sensors we have, we might get measurements of how much the motors have rotated (in radians), or an estimate of how fast they're rotating (angular rate, radian/sec.) We'll consider the first case, but of course the second case can be handled by just multiplying the angular rate by the amount of time that has elapsed since our last iteration (the equations of this thesis will be adopt the second case but in this chapter the first case is chosen just to clarify the odometery concept).

Given the amount of rotation of the motor and the diameter of the wheel, we can compute the actual distance that the wheel has covered.

Suppose the left wheel has moved by a distance of d_{left} and the right wheel has moved d_{right}. For some small period of time (such that d_{left} and d_{right} are short), we can reasonably assume that the vehicle trajectory was an arc (see Fig. 1).

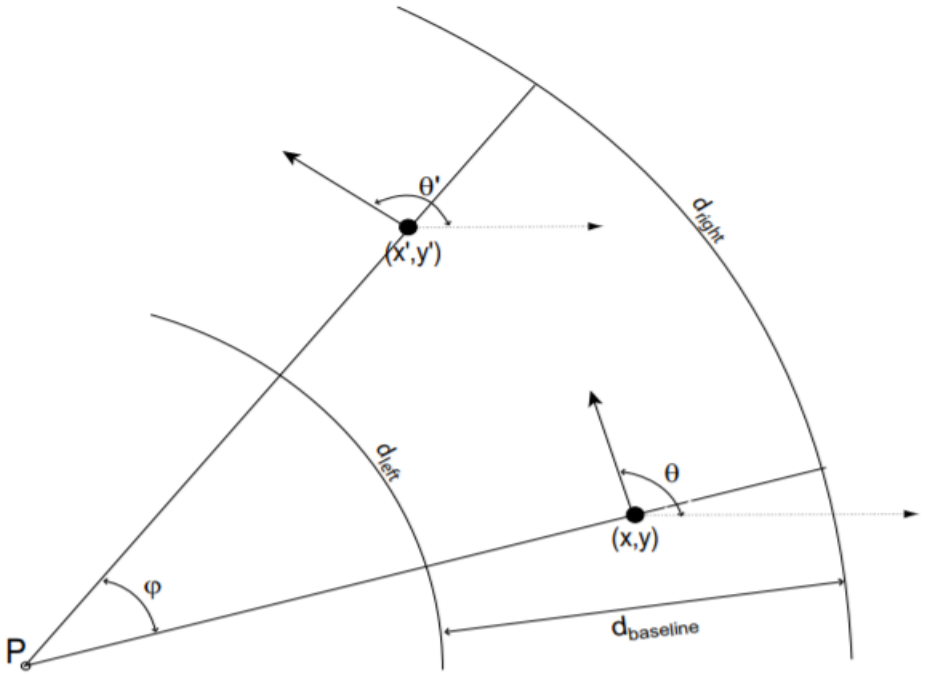

Figure 1: Odometery geometry. Over a small time period, the robot's motion can be approximated by an arc. The odometery problem is to solve for (x', y', θ') given (x, y, θ) and $d_{baseline}$. In the figure, the robot is moving counterclockwise.

The initial state (x, y, θ) defines the starting point, with θ representing the vehicle's heading. After our vehicle has moved by d_{left} and d_{right}, we want to compute the new position(x', y', θ').

The center of the robot (the spot immediately between the two wheels that defines the robot's location), travels along an arc as well. Remembering that arc length is equal to the radius times the inner angle, the length of this arc is:

$$d_{center} = \frac{d_{left} + d_{right}}{2} \quad (1)$$

Given basic geometry, we know that:

$$\varphi\, r_{left} = d_{left} \quad (2)$$

$$\varphi\, r_{right} = d_{right} \quad (3)$$

If $d_{baseline}$ is the distance between the left and right wheels, we cam write:

$$r_{left} + d_{baseline} = r_{right} \quad (4)$$

Subtracting (2) from (4), we see:

$$\varphi\, r_{right} - \varphi\, r_{left} = d_{right} - d_{left}$$

$$\varphi\, (r_{right} - r_{left}) = d_{right} - d_{left}$$

$$\varphi\, d_{baseline} = d_{right} - d_{left}$$

$$\varphi = \frac{d_{right} - d_{left}}{d_{baseline}} \quad (5)$$

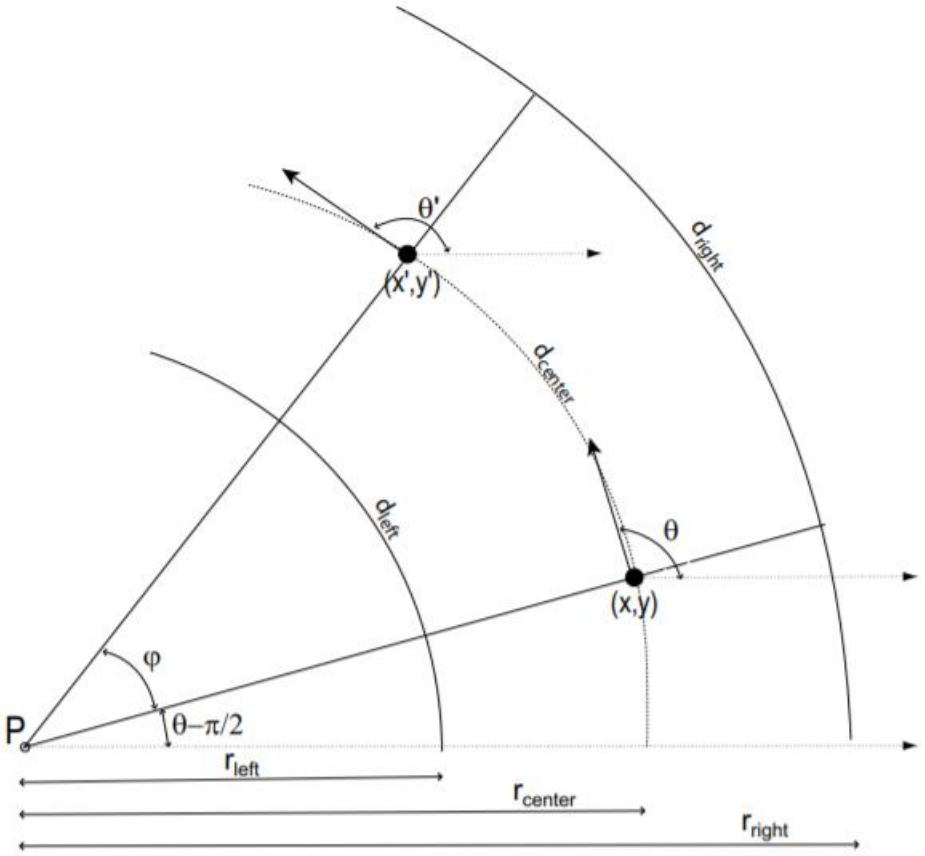

Figure 2: Detailed odometry geometry. We compute the center of the circle P by forcing d_{left} and d_{right} to be two arcs with the same inner angle φ. From this, we can compute the new robot position (x', y', θ').

At the risk of making the math a bit uglier than necessary, let's very carefully compute the new robot state. All our arcs have a common origin at point P.

Note that the angle of the robot's baseline with respect to x-axis is $\theta - \frac{\pi}{2}$.

So, we now compute the coordinates of P:

$$\begin{aligned} P_x &= x - r_{center} \cos\left(\theta - \frac{\pi}{2}\right) \\ &= x - r_{center} \sin(\theta) \\ P_y &= y - r_{center} \sin\left(\theta - \frac{\pi}{2}\right) \\ &= y - r_{center} \cos(\theta) \end{aligned}$$

Now we can compute x' *and* y'

$$\begin{aligned} x' &= P_x + r_{center}\cos\left(\varphi + \theta - \frac{\pi}{2}\right) \\ &= x - r_{center}\sin(\theta) + r_{center}\sin(\varphi + \theta) \\ &= x + r_{center}[-\sin(\theta) + \sin(\varphi)\cos(\theta) + \sin(\theta)\cos(\varphi)] \end{aligned} \quad (6)$$

And

$$\begin{aligned} y' &= P_y + r_{center}\sin\left(\varphi + \theta - \frac{\pi}{2}\right) \\ &= y - r_{center}\cos(\theta) + r_{center}\cos(\varphi + \theta) \\ &= x + r_{center}[\cos(\theta) - \cos(\varphi)\cos(\theta) + \sin(\theta)\sin(\varphi)] \end{aligned} \quad (7)$$

If φ is small (as is usually the case for small time steps), we can approximate $\sin(\varphi) = \varphi$ and $\cos(\varphi) = 1$. This now gives us:

$$\begin{aligned} x' &= x + r_{center}[-\sin(\theta) + \varphi\cos(\theta) + \sin(\theta)] \\ &= x + r_{center}\varphi\cos(\theta) \\ &= x + d_{center}\cos(\theta) \end{aligned} \quad (8)$$

And

$$\begin{aligned} y' &= y + r_{center}[\cos(\theta) - \cos(\theta) + \varphi\sin(\theta)] \\ &= y + r_{center}\varphi\sin(\theta) \\ &= y + d_{center}\sin(\theta) \end{aligned} \quad (9)$$

In summary, our odometry equations for (x', y', θ') reduce to:

$$d_{center} = \frac{d_{left} + d_{right}}{2} \quad (10)$$

$$\varphi = \frac{d_{right} - d_{left}}{d_{baseline}} \quad (11)$$

$$\theta' = \varphi + \theta \quad (12)$$

$$x' = x + d_{center}\cos(\theta) \quad (13)$$

$$y' = y + d_{center}\sin(\theta) \quad (14)$$

Dead reckoning Sensors

In order to build an odometry system, you must have a way of measuring the angular velocity of the motor. This can be accomplished by an encoder sensor which is placed on the motor shaft or wheel such that when the shaft rotates, the circuitry generates alternating 1's and 0's. One way of implementing a simple encoder is by attaching a disc with holes to the shaft, and using a break beam sensor to detect when a hole passes by. A mono phase encoder cannot determine the direction of motion. A serious failure mode occurs when the motor is essentially stationary with a hole halfway in front of the sensor: environmental noise can cause the encoder to trigger and untriggered, making it appear as though the shaft is rotating.

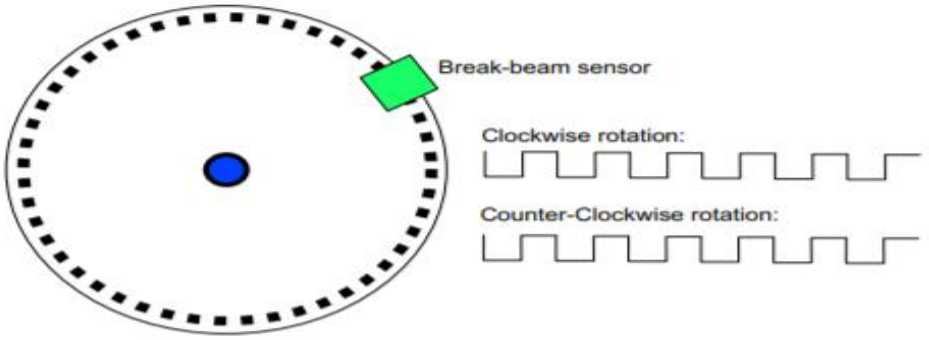

Figure 3: Simple Encoder. As a perforated disk rotates, a break beam sensor alternates between 1 and 0. The direction of motion is ambiguous.

Odometry error sources

Odometry can provide good dead reckoning over short distances, but error accumulates very rapidly. At every step, we inject error (noise) into not just x and y, but also θ. The error in θ is the killer, since every error in θ will be amplified by future iterations.

There are a number of basic noise sources

- ✓ Limited resolution during integration
- ✓ Unequal wheel diameter
- ✓ Variation in the contact point of the wheel
- ✓ Unequal floor contact and variable friction can lead to slipping

Odometry Errors

- Deterministic errors can be eliminated through proper calibration
- Non-deterministic errors have to be described by error models and will always lead to uncertain position estimate.

Kinematics

In mobile robotics, we need to understand the mechanical behavior of the robot both in order to design appropriate mobile robots for tasks and to understand how to create control software for an instance of mobile robot hardware.

A mobile robot's workspace is important because it defines the range of possible poses that the mobile robot can achieve in its environment. A mobile robot's controllability defines possible paths and trajectories in its workspace. Due to mass and force considerations robot dynamics places additional constraints on workspace and trajectory. The mobile robot is also limited by dynamics, for instance, a high center of gravity limits the practical turning radius of a fast, car-like robot because of the danger of rolling. A mobile robot is a self-contained automation that can wholly move with respect to its environment. There is no direct way to measure a mobile robot's position instantaneously. Instead, one must integrate the motion of the robot over time. Add to this the inaccuracies of motion estimation due to slippage and it is clear that measuring a mobile robot's position precisely is an extremely challenging task.

The process of understanding the motions of a robot begins with the process of describing the contribution each wheel provides for motion. Each wheel has a role in enabling the whole robot to move. By the same token, each wheel also imposes constraints on the robot's motion, for example refusing to skid laterally. In this section, we introduce notation that allows expression of robot motion in a global reference frame as well as the robot's local reference frame. Then, using this notation, we demonstrate the construction of simple forward kinematic models of motion, describing how the robot as a whole moves as a function of its geometry and individual wheel behavior. As we formally describe the kinematic constraints of individual wheels, and then combine these kinematic constraints to express the whole robot's kinematic constraints. With these tools, one can evaluate the paths and trajectories that define the robot's maneuverability.

Deriving a model for the whole robot's motion is a bottom-up process. Each individual wheel contributes to the robot's motion and, at the same time,

imposes constraints on robot motion. Wheels are tied together based on robot chassis geometry, and therefore their constraints combine to form constraints on the overall motion of the robot chassis. But the forces and constraints of each wheel must be expressed with respect to a clear and consistent reference frame. This is particularly important in mobile robotics because of its self-contained and mobile nature; a clear mapping between global and local frames of reference is required. We begin by defining these reference frames formally, then using the resulting formalism to annotate the kinematics of individual wheels and whole robots.

Coordinate Systems

In order to describe the position of the WMR in his environment, two different coordinate systems (frames) need to be defined

1. Inertial Coordinate System: This coordinate system is a global frame which is fixed in the environment or plane in which the WMR moves in. Moreover, this frame is considered as the reference frame and is denoted as {XI ,YI}.

2. Robot Coordinate System: This coordinate system is a local frame attached to the WMR, and thus, moving with it. This frame is denoted as {Xr ,Yr}.

The two defined frames are shown in Fig. 4. The origin of the robot frame is defined to be the mid-point A on the axis between the wheels.

The center of mass C of the robot is assumed to be on the axis of symmetry, at a distance d from the origin A.

As shown in Fig. 4, the robot position and orientation in the Inertial Frame can be defined as

$$q^I = \begin{bmatrix} x_a \\ y_a \\ \theta_a \end{bmatrix} \tag{15}$$

The important issue that needs to be explained at this stage is the mapping between these two frames. The position of any point on the robot can be defined in the robot frame and the inertial frame as follows.

Let $X^r = \begin{bmatrix} q^r \\ q^r \\ \theta^r \end{bmatrix}$, and $X^l = \begin{bmatrix} q^l \\ q^l \\ \theta^l \end{bmatrix}$ and be the coordinates of the given point in the robot frame and inertial frame, respectively. Then, the two coordinates are related by the following transformation:

$$X^l = R(\theta)X^r \tag{16}$$

Where R(θ) is the orthogonal rotation matrix

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

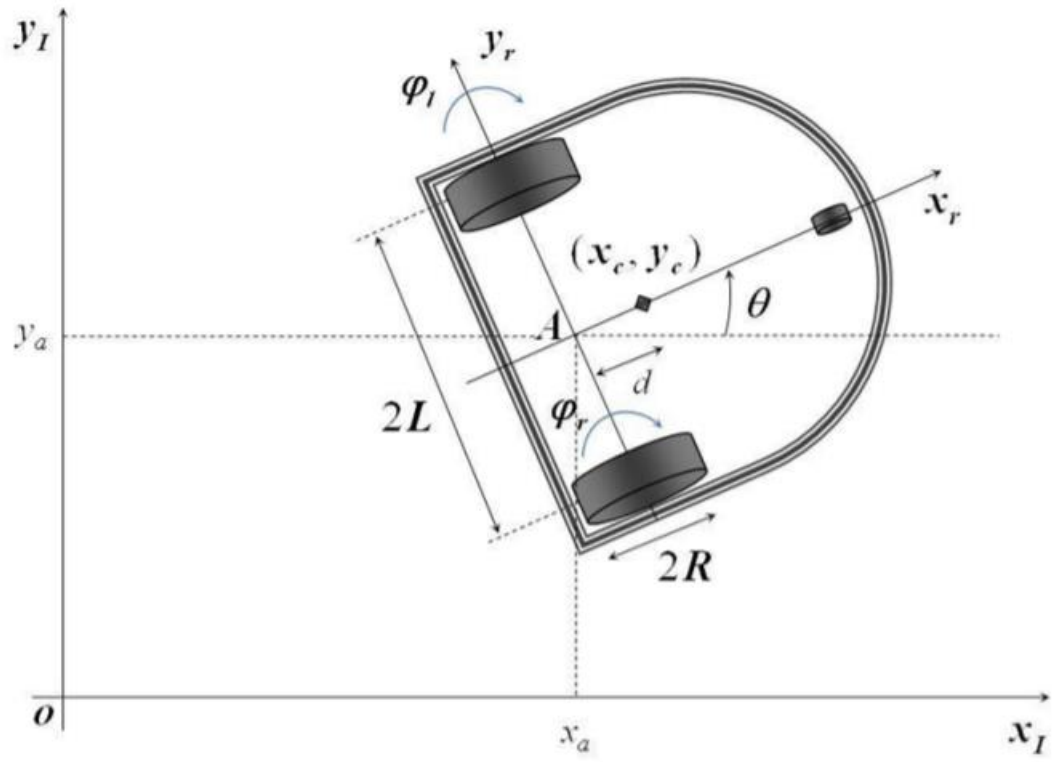

Figure 4: Differential Drive Mobile Robot (DDMR)

This transformation will enable also the handling of motion between frames

$$V^l = R(\theta)V^r \tag{18}$$

It will be seen in the next section that equation (18) is very important in deriving the DDMR kinematic and dynamic models as it describes the relationship between the velocities in the Inertial Frame and the Robot Frame.

Kinematic Constraints of the Differential-Drive Robot

The motion of a differential-drive mobile robot is characterized by two non-holonomic constraint equations, which are obtained by two main assumptions:

• No lateral slip motion: This constraint simply means that the robot can move only in a curved motion (forward and backward) but not sideward. In the robot frame, this condition means that the velocity of the center-point A is zero along the lateral axis:

$$\dot{y}_a^r = 0 \tag{19}$$

Using the orthogonal rotation matrix R(θ), the velocity in the inertial frame gives

$$-\dot{x}_a \sin\theta + -\dot{y}_a \cos\theta = 0 \tag{20}$$

• Pure rolling constraint: The pure rolling constraint represents the fact that each wheel maintains a one contact point P with the ground as shown in Figure 5. There is no slipping of the wheel in its longitudinal axis (x_r) and no skidding in its orthogonal axis (y_r). The velocities of the contact points in the robot frame are related to the wheel velocities by:

$$v_{pR} = R\dot{\varphi}_R \tag{21}$$

$$v_{pL} = R\dot{\varphi}_L \tag{22}$$

In the inertial frame, these velocities can be calculated as a function of the velocities of the robot center-point A:

$$\dot{x}_{PR} = \dot{x}_a + \mathrm{L}\theta\cos\theta \quad (23)$$

$$\dot{y}_{PR} = \dot{y}_a + \mathrm{L}\theta\sin\theta \quad (24)$$

$$\dot{x}_{PL} = \dot{x}_a + \mathrm{L}\theta\cos\theta \quad (25)$$

$$\dot{y}_{PL} = \dot{y}_a + \mathrm{L}\theta\sin\theta \quad (26)$$

Using the rotation matrix R(θ), the rolling constraint equations are formulated as follows:

$$\dot{x}_{PR}\cos\theta + \dot{y}_{PR}\sin\theta = R\dot{\varphi}_R \quad (27)$$

$$\dot{x}_{PL}\cos\theta + \dot{y}_{PL}\sin\theta = R\dot{\varphi}_L \quad (28)$$

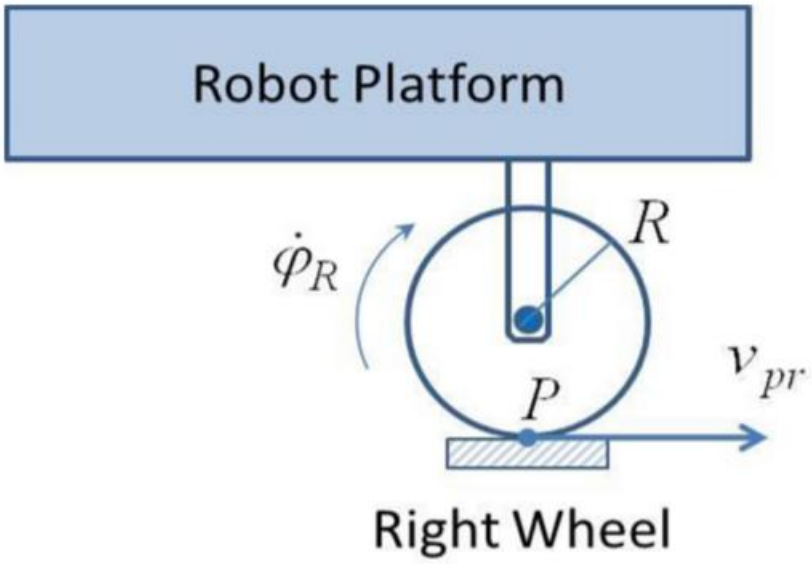

Figure 5: Pure Rolling Motion Constraint

Using the contact points velocities from equation (x, y) and substituting in (x, y), the three constraint equations can be written in the following matrix form:

$$\Lambda(q)\dot{q}=0$$

Where

$$\Lambda(q)=\begin{bmatrix}-\sin\theta & \cos\theta & 0 & 0 & 0\\ \cos\theta & \sin\theta & L & -R & 0\\ \cos\theta & \sin\theta & -L & 0 & -R\end{bmatrix}$$

and

$$\dot{q}=\begin{bmatrix}\dot{x}_a & \dot{y}_a & \dot{\theta} & \dot{\varphi}_R & \dot{\varphi}_L\end{bmatrix}^T$$

$$\begin{cases}v_R=R\dot{\varphi}_R\\ v_L=R\dot{\varphi}_L\end{cases} \quad (29)$$

The above constraints matrix Λ(q) will be used in the next section for the DDMR dynamic modeling.

Kinematic Model

Kinematic modeling is the study of the motion of mechanical systems without considering the forces that affect the motion. For the DDMR, the main purpose of kinematic modeling is to represent the robot velocities as a function of the driving wheels velocities along with the geometric parameters of the robot.

The linear velocity of each driving wheel in the Robot Frame is therefore, the linear velocity of the DDMR in the Robot Frame is the average of the linear velocities of the two wheels

$$v=\frac{v_R+v_L}{2}=R\frac{(\dot{\varphi}_R+\dot{\varphi}_L)}{2} \quad (30)$$

And the angular velocity of the DDMR is

$$\omega=\frac{v_R-v_L}{2L}=R\frac{(\dot{\varphi}_R-\dot{\varphi}_L)}{2} \quad (31)$$

The DDMRs velocities in the robot frame can now be represented in terms of the center-point A velocities in the robot frame as follows:

$$\begin{cases} \dot{x}_a^r = R\dfrac{(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ \dot{y}_a^r = 0 \\ \dot{\theta} = \omega = R\dfrac{(\dot{\varphi}_R + \dot{\varphi}_L)}{2L} \end{cases}$$

Thus

$$\begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \quad (32)$$

The DDMR velocities can be obtained also in the inertial frame as follows:

$$\dot{q}^I = \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\theta & \frac{R}{2}\cos\theta \\ \frac{R}{2}\sin\theta & \frac{R}{2}\sin\theta \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \quad (33)$$

Equation (33) represents the forward kinematic model of the DDMR. Another alternative form for the kinematic model can be obtained by representing the DDMR velocities in terms of the linear and angular velocities of DDMR in the Robot frame is the equation (34).

$$\dot{q}^I = \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (34)$$

Dynamic Modeling of the DDMR

Dynamics is the study of the motion of a mechanical system taking into consideration the different forces that affect its motion unlike kinematics where the forces are not taken into consideration. The dynamic model of the DDMR is essential for simulation analysis of the DDMR motion and for the design of various motion control algorithms.

A non-holonomic DDMR with n generalized coordinates (q1, q2, qn) and subject to m constraints can be described by the following equations of motion:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = B(q)\tau - \Lambda^T(q)^\lambda \qquad (35)$$

M (q) an nxn symmetric positive definite inertia matrix, V (q, $\dot{q}$) is the centripetal and coriolis matrix, F ($\dot{q}$) is the surface friction matrix, G (q) is the gravitational vector, τ_d is the vector of bounded unknown disturbances including unstructured un-modeled dynamics, B (q) is the input matrix, τ is the input vector, $\Lambda^T(q)$ is the matrix associated with the kinematic constraints, and λ is the Lagrange multipliers vector.

Lagrange dynamic approach

Lagrange dynamic approach is a very powerful method for formulating the equations of motion of mechanical systems. This method, which was introduced by Lagrange, is used to systematically derive the equations of motion by considering the kinetic and potential energies of the given system.

The Lagrange equation can be written in the following form:

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}_i}\right) + \frac{dL}{dq_i} = F - \Lambda^T(q)^\lambda \qquad (36)$$

Where L=T-V is the Lagrangian function, T is the kinetic energy of the system, V is the potential energy of the system, qi are the generalized coordinates, F is the generalized force vector, Λ is the constraints matrix, and λ is the vector of Lagrange multipliers associated with the constraints.

The first step in deriving the dynamic model using the Lagrange approach is to find the kinetic and potential energies that govern the motion of the DDMR. Furthermore, since the DDMR is moving in the {XI, YI}. Plane, the potential energy of the DDMR is considered to be zero.

For the DDMR, the generalized coordinates are selected as

$$q = [\, x_a \quad y_a \quad \theta \quad \varphi_R \quad \varphi_L \,]^T \tag{37}$$

The kinetic energies of the DDMR is the sum of the kinetic energy of the robot platform without wheels plus the kinetic energies of the wheels and actuators.

The kinetic energy of the robot platform is

$$T_c = \frac{1}{2} m_c V_c^2 + \frac{1}{2} I_c \dot{\theta}^2 \tag{38}$$

While the kinetic energy of the right and left wheel is

$$T_{wR} = \frac{1}{2} m_w V_{wR}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_R^2 \tag{39}$$

$$T_{wL} = \frac{1}{2} m_w V_{wL}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_L^2 \tag{40}$$

where, m_c is the mass of the DDMR without the driving wheels and actuators (DC motors), m_w is the mass of each driving wheel (with actuator), I_c is the moment of inertia of the DDMR about the vertical axis through the center of mass, I_w is the moment of inertia of each driving wheel with a motor about the wheel axis, and I_m is the moment of inertia of each driving wheel with a motor about the wheel diameter.

All velocities will be first expressed as a function of the generalized coordinates using the general velocity equation in the inertial frame.

$$V_i^2 = \dot{x}_i^2 + \dot{y}_i^2 \quad (41)$$

The Xi and Yi components of the center of mass and wheels can be obtained in terms of the generalized coordinates as follow

$$x_c = x_a + d\cos\theta$$

$$y_c = y_a + d\sin\theta \quad (42)$$

$$x_{wR} = x_a + L\cos\theta$$

$$y_{wR} = x_a + L\sin\theta \quad (43)$$

$$x_{wL} = x_a - L\cos\theta$$

$$y_{wL} = y_a + L\sin\theta \quad (44)$$

Using equations (38)-(40) along with equations (41- (44), the total kinetic energy of the DDMR is

$$T = \frac{1}{2}m(\dot{x}_a^2 + \dot{y}_a^2) - m_c d\dot{\theta}(\dot{y}_a cos\theta - \dot{x}_a sin\theta) + \frac{1}{2}I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) + \frac{1}{2}I\dot{\theta}^2 \quad (45$$

Where the following new parameters are introduced

$m = m_c + 2m_w$ is the total mass of the robot,

$I = I_c + m_c d^2 + 2m_w L^2 + 2I_m$ is the total equivalent inertia.

Using equation (36) along with the Lagrangian function, L=T the equations of motion of the DDMR are given by

$$m\ddot{x}_a - md\ddot{\theta}\sin\theta - md\dot{\theta}^2\cos\theta = C_1 \quad (46)$$

$$m\ddot{y}_a - md\ddot{\theta}\cos\theta - md\dot{\theta}^2\sin\theta = C_2 \quad (47)$$

$$I\ddot{\theta} - md\ddot{x}_a\sin\theta + md\ddot{y}_a\cos\theta = C_3 \quad (48)$$

$$I\ddot{\theta}_R = \tau_R + C_4 \quad (49)$$

$$I\ddot{\theta}_L = \tau_L + C_5 \quad (50)$$

where (C1 , C2 , C3 , C4 , C5), are coefficients related to the kinematic constraints, which can be written in terms of the Lagrange multipliers vector λ and the kinematic constraints matrix Λ introduced in previous section.

$$\Lambda^{T}(q) = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix} \tag{51}$$

Now, the obtained equations of motion (46)-(50) can be represented in the general form given by equation (35) as

$$M(q)q + V(q,\dot{q})\ddot{q} = B(q)\tau - \Lambda^{T}(q)^{\lambda} \tag{52}$$

Where

$$M(q) = \begin{bmatrix} m & 0 & -md\sin\theta & 0 & 0 \\ 0 & m & md\cos\theta & 0 & 0 \\ -md\sin\theta & md\cos\theta & I & 0 & 0 \\ 0 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix},$$

$$V(q,\dot{q}) = \begin{bmatrix} 0 & -md\dot{\theta}\cos\theta & 0 & 0 & 0 \\ 0 & -md\dot{\theta}\sin\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B(q) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ and } \Lambda^{T}(q)\lambda = \begin{bmatrix} -\sin\theta & \cos\theta & \cos\theta \\ \cos\theta & \sin\theta & \sin\theta \\ 0 & L & -L \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix} \times \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{bmatrix}$$

Next, the system described by equation (52) is transformed into an alternative form which is more convenient for the purpose of control and simulation. The main aim is to eliminate the constraint term $\Lambda^{T}(q)^{\lambda}$ in equation (52) since the Lagrange multipliers λi are unknown. This is done first by defining the reduced vector

$$\dot{\eta} = \begin{bmatrix} \dot{\vartheta}_R \\ \dot{\vartheta}_L \end{bmatrix} \tag{53}$$

Next, by expressing the generalized coordinates velocities using the forward kinematic model (33). Then we have

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_a \\ \dot{\theta} \\ \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} = \frac{1}{2} \begin{bmatrix} R\cos\theta & R\cos\theta \\ R\sin\theta & R\sin\theta \\ \frac{R}{L} & -\frac{R}{L} \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{54}$$

Wheels velocity VS Robot behavior

Before we derive the algorithm Extended Kalman Filter (EKF) in order to estimate the future pose of our robot, let's see what the different values of wheel velocities can do to our robot. In such way the summary of our kinematics with different torque profile

- Here the focus just in the velocities in the section of simulation we will see more details about torque and friction).
- The next figures are just summery for what we have discussed before.

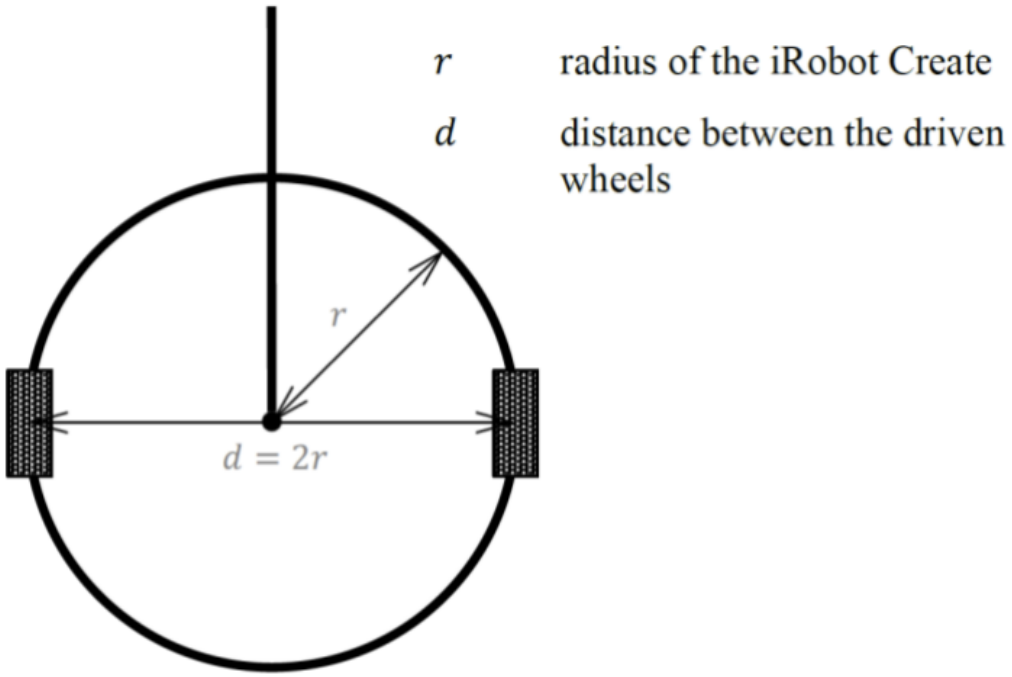

Figure 6: Robot Top view model

''… Driving control for differential drive is more complex than for single wheel drive, because it requires the coordination of the two driven wheels… If both motors run at the same speed, the robot drives straight forward or backward, if one motor is running faster than the other, the robot drives in a curve along the arc of a circle, and if both motors are run at the same speed in opposite directions, the robot turns on the spot.''

The above driving actions are illustrated in Fig.7 and Fig.8.

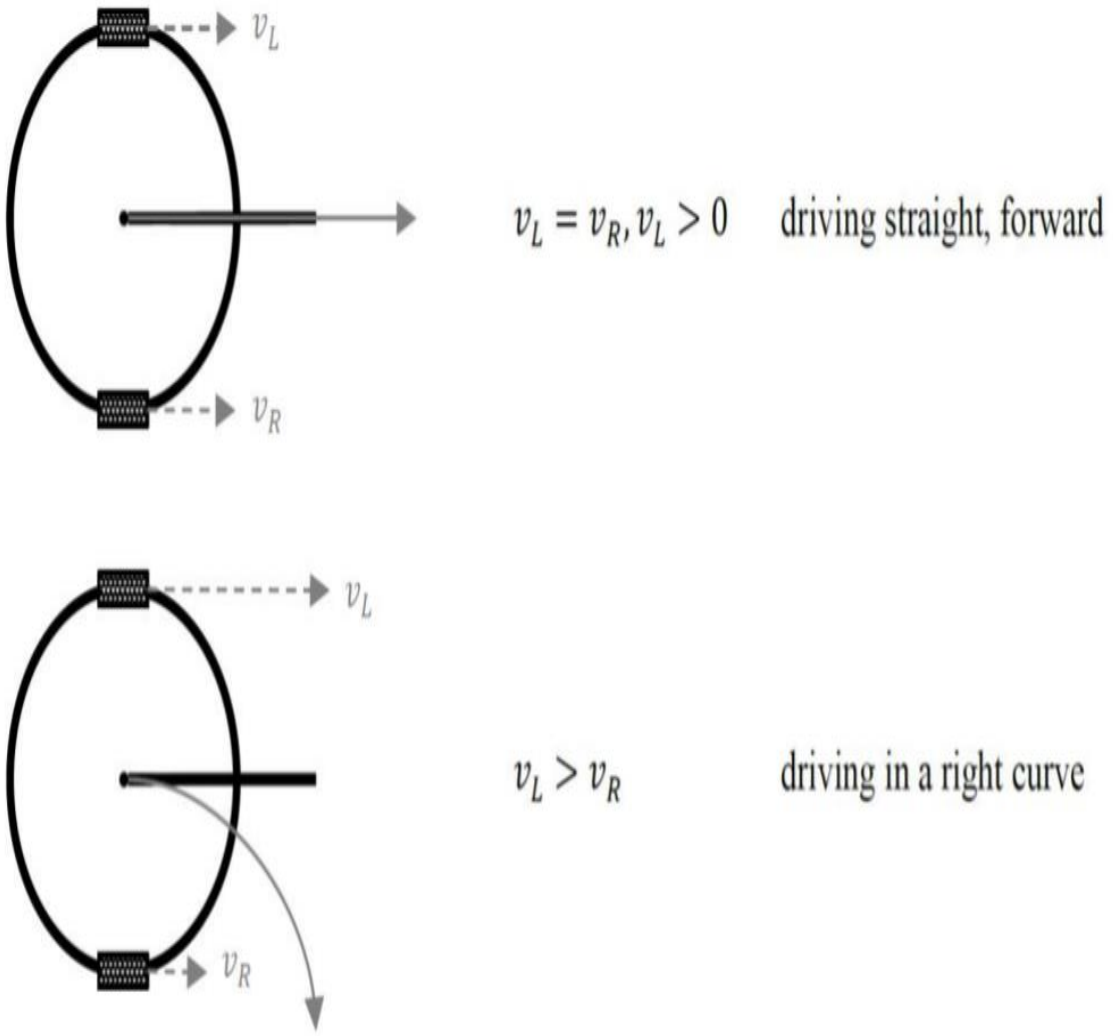

Figure 7: Driving forward and driving in a curve of differential drive

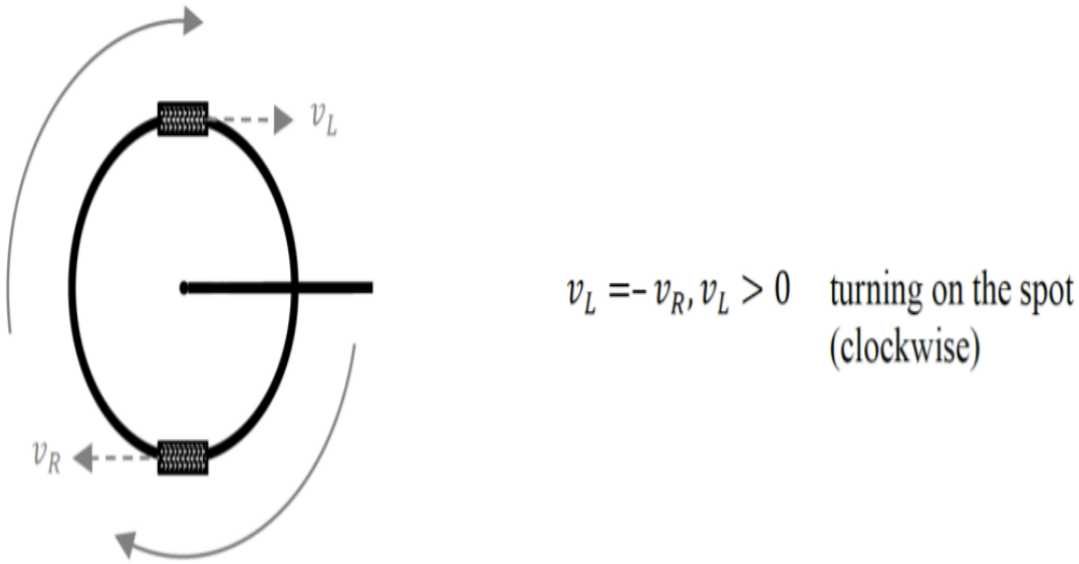

Figure 8: Rotation of differential drive

So, The kinematics of iRobot Create can be described as the relations among the driven wheel velocities $v_{R,L}$, linear velocity v , and angular velocity , which are illustrated in Fig.9.

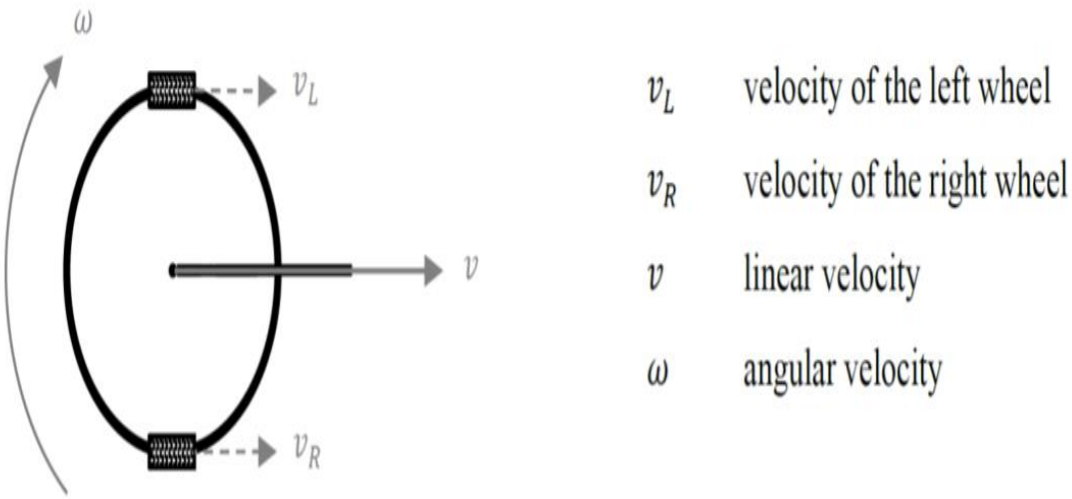

Figure 9: Linear Velocity, Angular velocity, and Driven Wheel Velocities

Dynamics of Wheels

By referring to the wheels dynamic equation (49) and (50) we can model the first part which is our dynamics

$$\ddot{\theta}_{\mathrm{R}} = \frac{\tau_{\mathrm{R}}}{I} + \frac{C_4}{I} \qquad (55)$$

$$\ddot{\theta}_{\mathrm{L}} = \frac{\tau_{\mathrm{L}}}{I} + \frac{C_5}{I} \qquad (56)$$

If we integrate the wheels angular acceleration $\ddot{\theta}_{\mathrm{R,L}}$ we get our control inputs $\dot{\vartheta}_{R,L}$ the right and left wheel angular velocities to be able compute the odometery by the aid of the kinematic equations seen before, see the matrix form (54).

Again $\mathrm{I} = \mathrm{I_c} + \mathrm{m_c d^2} + 2\mathrm{m_w L^2} + 2\mathrm{I_m}$ is the total equivalent inertia. $\tau_{\mathrm{R,L}}$ Right and Left wheel motor torque according to the profile chosen $C_{4,5}$ Our target in this section which are the torque that introduced to our system by the traction forces on wheels and they include the friction coefficient and the slip ratio depending on the road we (wet, dry, snow and ice etc. …)

One must consider the interaction of the wheels with the ground to develop a model for this interaction, which will lead to obtain the solution to the motion equations of the robot. The models are complex and require several properties of the tires and the ground in order to estimate the interaction between the tire and the ground. These properties are determined through extensive empirical experiments. However, most of the applications in robotics involve rigid wheels on a rigid ground. Therefore, the approach of using a friction model is simple and reasonable for this type of contact between rigid surfaces. In the following, we will describe the traction force model.

Consider a wheel that is rotating without slip as shown in Fig.12 .

The equations of motion of the wheel are given by

$$m_w a_w = F_{long} \qquad (57)$$

$$I\ddot{\theta} = \tau - F_{long}\mathrm{R} \qquad (58)$$

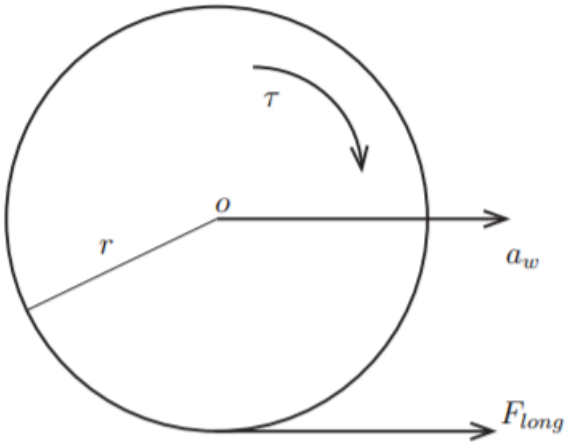

Figure 10 Wheel rotating without slip

N.B: r = R

Where a_w is the linear acceleration of the wheel.

Under pure rolling conditions the angular acceleration $\ddot{\theta}$ is related to linear acceleration as, see also Figure 5 and equation (29)

$$a_w = \mathrm{R}\ddot{\theta} \qquad (59)$$

If we consider these equations and express the longitudinal force in terms of applied torque, we obtain

$$F_{long} = \frac{\tau * m_w * R}{I + m_w R^2} \qquad (60)$$

Such a simple relation between longitudinal force and applied torque cannot be established when there is slip.

In the presence of slip, we consider the Coulomb friction model to establish the relationship between the normal force and the force exerted by the applied torque, due to irregularities of the bodies in contact. A static friction coefficient is utilized to determine the value of the maximum force that has to be applied to the wheel before the wheel starts to slip.

Considering μ_s to be the coefficient of static friction and N as the normal force on the wheel, we have

$$F_{long} = \mu_s N \qquad (61)$$

If the resultant force due to the applied torque is less than the static frictional force, it is completely transmitted to the ground resulting in pure rotation of the wheel. If the resultant force exceeds the static frictional force, a portion of this force that is above the static frictional force causes the wheel to slip (i.e., to rotate without any linear displacement), the remaining force is utilized for the forward motion of the wheel. Hence, slip occurs if the value of F_{long} given by equation (60) is greater than that of equation (61). The force that results in the linear motion is determined by the kinetic friction coefficient and is given by

$$F_{long} = \mu_k N \qquad (62)$$

Lateral slip of the wheel may occur in conjunction with longitudinal slip. In order to determine the actual relation between the amount of slip and the force applied, we decompose the total force due to the applied torque into lateral and longitudinal force components as shown in Fig.13. Decomposing the total force ($\mu_k N$) into longitudinal and lateral forces.

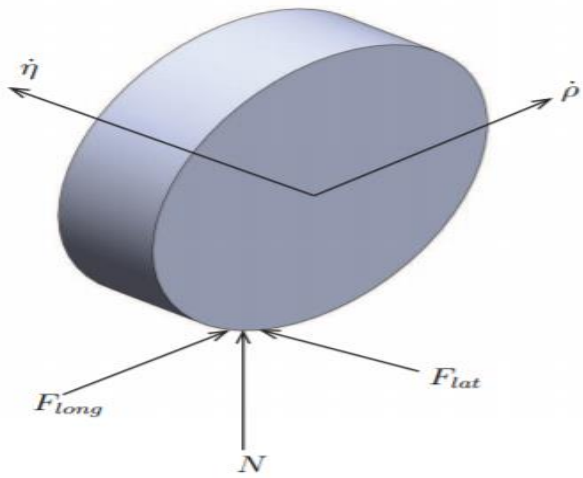

Figure 11: Longitudinal and lateral slip due to friction forces

We obtain

$$F_{longr} = \mu_k N \left(\frac{\dot{\rho}_r}{\sqrt{\dot{\rho}_r^2+\dot{\eta}^2}}\right) \quad (63)$$

$$F_{longl} = \mu_k N \left(\frac{\dot{\rho}_l}{\sqrt{\dot{\rho}_l^2+\dot{\eta}^2}}\right) \quad (64)$$

We considered the longitudinal forces where the subscript r and l are used to refer to the right and left wheel, respectively. And to get the torques we just multiply by the wheel radius so,

$$\tau_{longr} = RF_{longr} \quad (65)$$

$$\tau_{longl} = RF_{longl} \quad (66)$$

In the previous section kinematics and dynamics under pure rolling and without lateral slip are derived. However, in practice, there is slip in both lateral and longitudinal directions.

Lateral slip is present when the direction of the movement is different from wheel's plane of rotation, so the wheel must slip in order to reorient itself in the desired direction; see Fig.14.

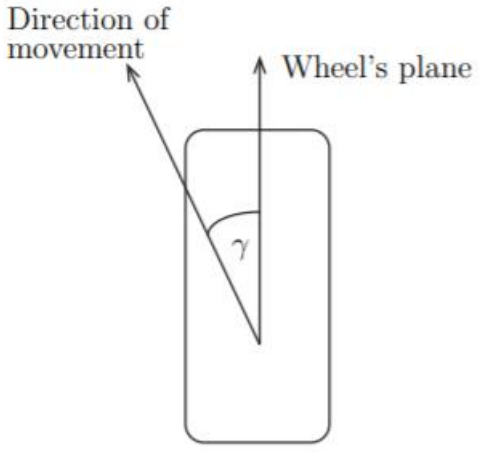

Figure 13 : Lateral slip due to difference in robot motion direction and plane of rotation of the wheel

Longitudinal slip occurs when the input torque is not completely transmitted to the ground. As a consequence, the linear speed of the where ($\dot{\rho}$) is not equal to the peripheral speed of the wheel, i.e. $\dot{\rho} < \dot{\theta}R$. A portion of the applied torque is transmitted to the ground resulting in wheel forward motion, the rest is consumed by pure wheel rotation causing wheel slip. Longitudinal slip is also possible when the wheel stops rotating but the linear velocity is not zero.

Lateral slip is a phenomenon that has minimal effect on achieving a motion objective for the robot at low speeds. It can be observed mainly when the robot is in motion along a curved path at high speeds, i.e., when continuous and rapid orientation changes are required. Longitudinal slip is more prevalent as it is a direct consequence of the application of wheel torques beyond what can be supported by the traction between the wheels and the ground.

It is directly influenced by the value of coefficient of friction along the motion path. In the following we provide equations of motion under slip

In the presence of slip, the non-holonomic constraints are given by

$$\dot{\rho}_r = \dot{x}_c \cos\theta + \dot{y}_c \sin\theta + \frac{1}{2}L\dot{\theta} \qquad (67)$$

$$\dot{\rho}_l = \dot{x}_c \cos\theta + \dot{y}_c \sin\theta - \frac{1}{2}L\dot{\theta} \qquad (68)$$

$$\dot{\eta} = \dot{x}_c \sin\theta - \dot{y}_c \cos\theta - d\dot{\theta} \qquad (69)$$

Where $\dot{\eta}$ represents the lateral velocity due to slip and$\dot{\rho}_r$, $\dot{\rho}_l$ are the linear velocities of right and left wheels, respectively.

Extended Kalman Filter (EKF)

In the mobile robotic systems, a precise estimate of the robot pose with the intention of the optimization in the path planning is essential for the correct performance, on the part of the robots, for tasks that are destined to it. Sensors data like odometry, compass, and the result of triangulation Cartesian estimative, are fused for better position estimative. It uses a mathematical and computational tool for nonlinear systems with time-discrete sampling for pose estimative calculation of mobile robots, with the utilization of extended Kalman filter (EKF). A mobile robot platform with differential drive and non-holonomic constraints is used as a base for state space, plants and measurements models that are used in the simulations and validation of the experiments.

Kalman Filtering

Kalman Filter (KF) is a well-known algorithm for estimation and prediction especially when data has a lot of noise. KF is used for linear transition functions whereas under non-linear transition, Extended Kalman Filter (EKF) is used. A brief summary of the basic discrete time linear Kalman Filter as follows. The Kalman Filter produces a state estimate of a discrete time linear difference equation of the form given in equation (70), where x_k is the current state of the system, A is the linear process dynamics matrix, B is the matrix that relates the control input u_{k-1}, and w_{k-1} is the Gaussian process noise. Equation (71) describes how the sensors produce a measurement z_k of the process given in equation (70).where H is the measurement matrix with a number of rows equal to the number of sensor inputs and v_k is the Gaussian measurement noise for the sensors

$$\mathrm{x_k} = \mathrm{Ax_{k-1}} + \mathrm{Bu_{k-1}} + \mathrm{w_{k-1}} \quad (70)$$

$$\mathrm{z_k} = \mathrm{Hx_k} + \mathrm{v_k} \quad (71)$$

The two fundamental assumptions in the linear Kalman Filter are that the estimated discrete time process is linear, and the measurement and process noise are Gaussian distributions with zero means.

The Kalman Filter algorithm is recursive, and runs in two stages. The first stage is called the 'predictor' in which a new state estimate is produced from the previous estimate. The second stage is called the 'corrector' in which the estimate produced by the predictor stage is adjusted based on the new sensor measurements produced by equation (70). The algorithm only uses data from the previous state estimate, and thus requires minimal data storage and computation to produce the new state update.

Extended Kalman Filtering

As we have noticed before both the kalman filter and the extended kalman filter has the same algorithm strategy, based on the predictor stage, innovation and correction stage. But the only difference is that the extended kalman filter is used in the non-linear system as we will see in our case where the state transition and observation models are non- linear functions of the state and input.

The dynamic model of the EKF is given by,

$$x_k = f(x_{k-1}, u_k) + w_k \tag{72}$$

$$z_k = h(x_k) + v_k \tag{73}$$

The function f can be used to compute the predicted state from the previous estimate and similarly the function h can be used to compute the predicted measurement from the predicted state. However, f and h cannot be applied to the covariance directly. Instead, a matrix of partial derivatives (the Jacobian) is computed.

- $u_k,\ w_k\ and\ v_k$ Have been defined before.
- At each time step the Jacobian is evaluated with current predicted states. These matrices can be used in our filter equations. This process essentially linearizes the non-linear function around the current estimate.

EKF algorithm:

Predict

Predicted State

$$\hat{x}_{k|k-1} = f(\hat{x}_{k|k-1}, u_{k-1}) \quad (74)$$

Predicted estimate covariance

$$C_{k|k-1} = F_{k-1} C_{k-1|k-1} F_{k-1}^{T} + W_{k-1} \quad (75)$$

Update

Innovation or measurement residual

$$\hat{y}_k = z_k - h(\hat{x}_{k|k-1}) \quad (76)$$

Innovation (or residual) covariance

$$S_k = H_k C_{k|k-1} H_k^{T} + V_k \quad (77)$$

Optimal Kalman Gain

$$K_k = C_{k|k-1} H_k^{T} S_k^{-1} \quad (78)$$

Updated State estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (79)$$

Updated estimate covariance

$$C_{k|k} = (I - K_k H_k) C_{k|k-1} \quad (80)$$

Where the state transition and observation matrices are defined to be the following based on the previous discussion of the odometery and the robot analysis.

In particular our system becomes:

The function f :

$$\boldsymbol{f}\begin{cases} x_{k+1} = x_k + \frac{R}{2}Ts\,(\dot{\vartheta}r_k + \dot{\vartheta}l_k)\cos\theta \\ y_{k+1} = y_k + \frac{R}{2}Ts\,(\dot{\vartheta}r_k + \dot{\vartheta}l_k)\sin\theta \\ \theta_{k+1} = \theta_k + \frac{R}{d}Ts\,(\dot{\vartheta}r_k - \vartheta l_k) \end{cases} \quad (81)$$

Where:

- $X_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}$ is the pose of the robot.
- $u_k = \begin{bmatrix} \dot{\vartheta}r_k \\ \dot{\vartheta}l_k \end{bmatrix}$ is the control input, wheels angular velocity.
- Ts is the sampling time.
- R is the robot wheel radius.
- K is the current state
- K-1 s the previous state
- K+1 is the future state

The function h:

The localization in structuralized environment is helped, in general, by external elements that are called of land markers. It is possible to use natural markers that already existing in the environment for the localization. Another possibility is to add intensionally to the environment artificial markers to guide the localization of the robot

$$\boldsymbol{h}\begin{cases} h_1 = \sqrt{(f_{(x)} - x)^2 + (f_{(y)} - y)^2} \\ h_2 = \arctan 2\frac{fy-y}{fx-x} - \theta \end{cases} \quad (82)$$

Where:

- (x, y, θ) current state
- $(f_{(x)}, f_{(y)})$ landmarks

Jacobians

$$F_{k-1} = \frac{df}{dX} \mid_{\hat{X}_{k-1|k-1,u_{k-1}}}$$

$$= \begin{bmatrix} 1 & 0 & -\frac{\mathrm{R}}{2}\mathrm{Ts}(\dot{\vartheta}\mathrm{r_k} + \dot{\vartheta}\mathrm{l_k})\sin\theta \\ 0 & 1 & \frac{\mathrm{R}}{2}\mathrm{Ts}(\dot{\vartheta}\mathrm{r_k} + \dot{\vartheta}\mathrm{l_k})\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \quad (83)$$

$$H_k = \frac{dh}{dX} \mid_{\hat{x}_{k|k-1}}$$

$$= \begin{bmatrix} \frac{dh_1}{dx} & \frac{dh_1}{dy} & \frac{dh_1}{d\theta} \\ \frac{dh_2}{dx} & \frac{dh_2}{dy} & \frac{dh_2}{d\theta} \end{bmatrix} \quad (84)$$

Modeling and simulation

Modeling is a way to create a virtual representation of a real-world system. We can simulate this virtual representation under a wide range of conditions to see how it behaves and to test the proposed robot analysis (Kinematics, dynamics, sensors and filters) and evaluate the response of the models due to different profiles.

We have discussed in the previous chapters the kinematics and dynamics of the differential mobile robot, as we have seen the odometry which comes out from an encoder sensor to get up the pose of the mobile robot, but due to the errors of this sensor we will advise this sensor by an EKF to estimate a perfect pose of our robot. So we will adopt the MATLAB/SIMULINK to model, simulate and analyze our mobile robot (Unicycle differential drive mobile robot).

So, this chapter introduces our robot system by using Simulink/Matlab in order to define the physical concepts explained before and test due to different scenarios.

Kinematic Model (in order to get the current pose)

By the having the linear and angular velocity of the robot and use them as inputs to the equation (34) that models the kinematic of the DDMR where It has been deduced from the geometry and constraints of the pure rolling wheels. So far to get the robot pose we must integrate it by adding the block integrator via Simulink. Fig.10 and Fig.11 show us how to build or kinematics by the aid of Simulink blocks.

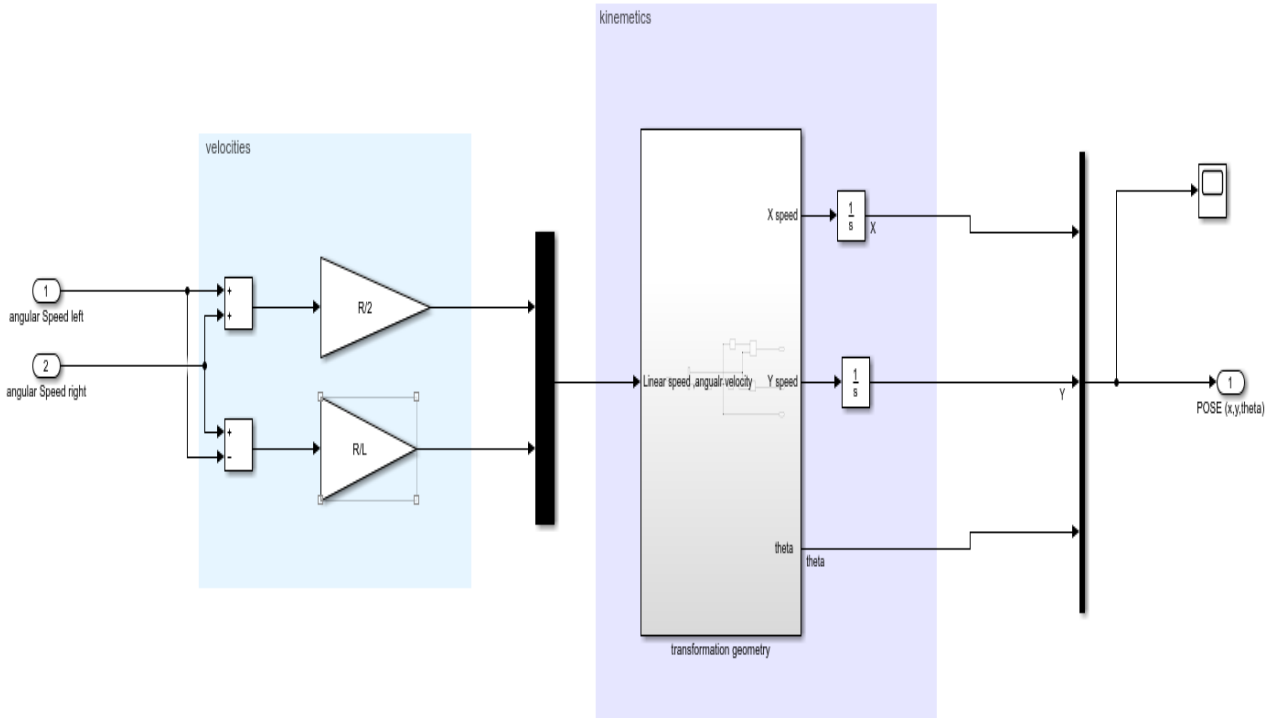

Figure 14: linear and angular velocity as inputs to the kinematics of the robot Equation ((30), (31) and (34))

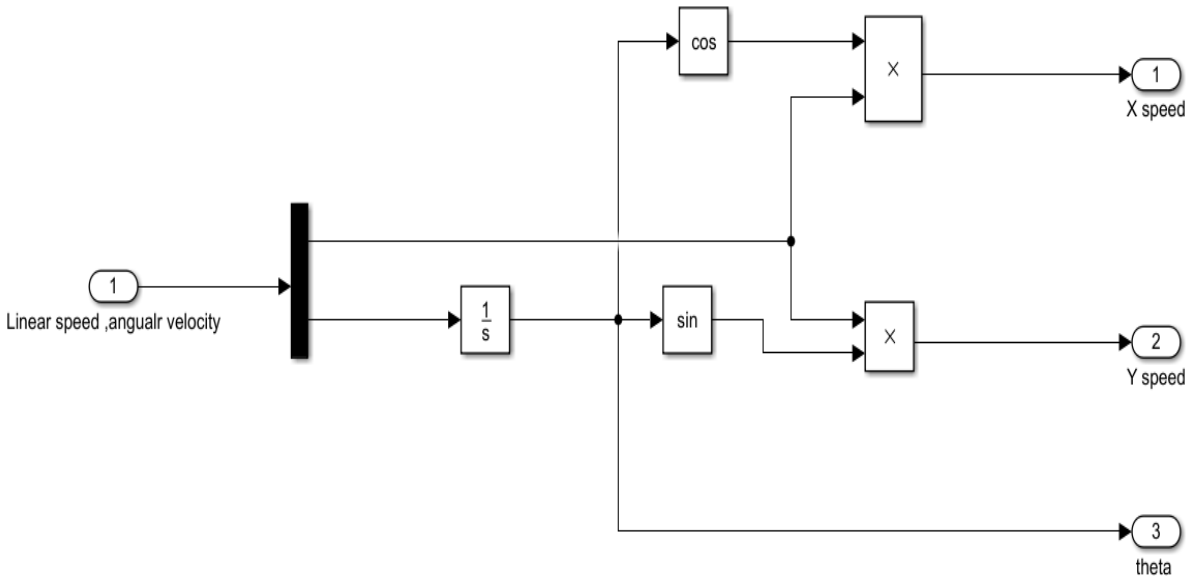

Figure 15: Robot's kinematics / equation (34)