Omni-wheel robot Dynamics, Kinematics, and Controllers

Benjamin M Dyer
MASc Student
Department of Mechanical Engineering
University of Guelph
Guelph, Ontario
Email: dyerb@uoguelph.ca

The purpose of this report is to give an overview of what has been investigated since the start of my masters with regard to the omni-wheel robot. This document discusses a number of dynamic and kinematic models developed for the omni-wheel robot. Using these models a number of controllers are developed and testing through experiment and simulation. In the appendices Kalman filtering is discussed along with a least squares method for finding unknown variables in the dynamics of the robot.

1 Omniwheel Platforms

Two omniwheel platforms are used over the past two semesters. The first is a four wheeled platform which was built prior to the work discussed in this report. It was primarily used for testing of controllers and to gain a better understanding of the dynamics of omniwheeled robots. The second robot is a three wheeled platform built specifically for the research project aiming to solve issues that would have been encountered trying to retrofit the four wheeled platform.

1.1 Four wheeled platform

1.1.1 Architecture

The robot is comprised of 5 main subsystems. Power management is handled by two NiMH batteries in series providing 14.4V nominal with a capacity of 3000mAh. The batteries are connected to a variable buck converter which is set to an output voltage of 12V. Any further reduction in voltage is carried out by the device which requires lower power. The second subsystem is the micro-controller. In this case a Arduino Uno is used due to its low cost and it's logic level being 5V, this is important since the rest of the modules expect a 5V logic, eliminating the need for logic conversion. The Arduino receives data from the XBee WiFi module, the third subsystem. Data is sent to the XBee at 9600 BAUD which is then received and used by the Arduino to determine PWM levels for each motor. The Arduino is connected to the fourth sub-system, the motor drivers, which are able to provide up to 2A of PWM power to each motor. Finally the fifth subsystem, the motors, use 12V power from the motor drivers to run and are able to provide a max torque of 23.15 N-m. Figure 1 shows the full architecture.

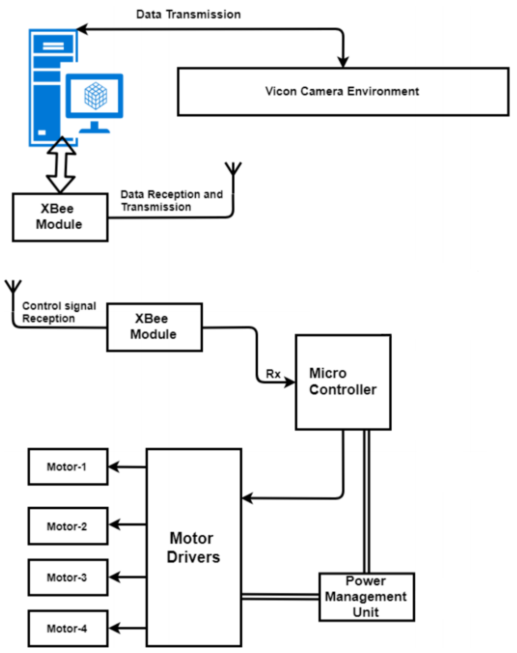

Fig. 1. Block Diagram of Robot and Vicon Architecture patel

1.1.2 Vicon System

Positional tracking of the robot is achieved using the VICON system. A set of six IR reflectors are placed on the robot as shown in figure 2 which are then picked up by the eight IR cameras of the VICON system. By viewing at least five of the six reflectors the system is able to determine the position and orientation of the robot. The positional information is then fed into a Matlab script in order to implement the controller. This has the advantage of using a faster processor for the controller, resulting in a faster feedback loop, typically at 50Hz. Once the controller has completed it's cycle,

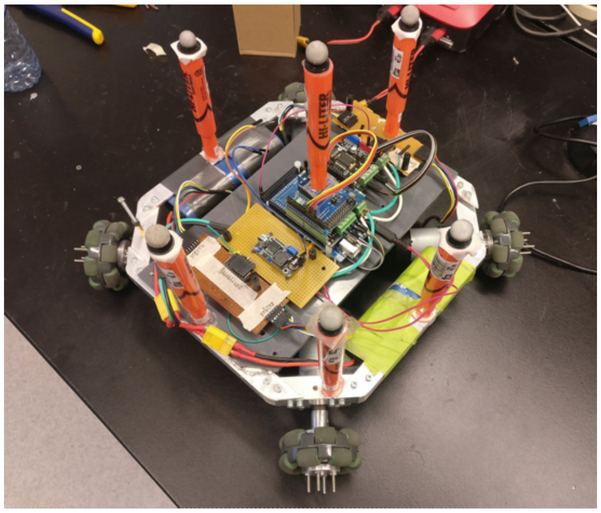

Fig. 2. Image of robot platform showing the VICON markers and all sub-systems

the PWM values are sent to the robot using another XBee WiFi module.

1.2 Three wheeled platform

1.2.1 Rational

The four wheeled platform has been extremely useful for development of controllers for omniwheel robots, however there are a few major issues that make it unsuited to the task of environmental sensing.

The first major issue is size. Accurate sensing of wind speeds, temperature, and relative humidity require a minimum of two large sensors, an ultrasonic anemometer and a relative humidity sensor. Both of the sensors must be mounted on a pole, preferably over one meter above the robot platform. Mounting the sensors at such a height requires a larger base in order to provide stability to the robot and avoid any change of the robot tipping due to a high center of gravity. In addition to stability, the a larger platform allows for more space to mount supporting hardware for the sensors, such as Campbell Scientific CR6 Data-loggers. Retrofitting the four wheeled robot would require a significant change to the structure to allow for the mounting sensors and supporting hardware.

A second major issue is the slight imbalance inherent in a four wheel mobile robot. Since the robot has four points of contact with the ground, without any suspension system one of the wheels will always have a lower contact force than the other three. This issue is amplified when not operating on a completely flat surface. The robot could be fixed by adding a suspension system, however this would also require significant modification of the structure and will still have issues in rough terrain.

A more minor issue is the robots modularity. The goal of developing a new platform is to allow it to be extremely modular so that once this project concludes, the platform can be used with any variation of sensors. This means that during the design process a lot of thought was put into making the robot easy to assemble and disassemble, and making parts that are be regularly accessed easy to change out.

1.2.2 Electrical design

The robot follows the same design architecture as the three wheeled platform with the exception of the Vicon system. The goal is to have the robot be fully autonomous so tracking of position will be accomplished using sensors and an accurate dynamic or kinematic model of the robot.

Despite the architecture being relatively unchanged, in order to provide adequate power delivery for high torque low speed motors and any sensor, custom power delivery boards and motor drivers were developed. These include a 12V buck converter, a 3.3V/5V buck converter, and a motor driver board. In addition, a Teensy4.0 is used as the micro-controller instead of the Arduino Mega2560 since it is significantly faster, has more pin functionalities, and consumes significantly less power.

1.2.2.1 12V buck converter

The 12V buck converter is designed to handle up to 10A of current allowing for a maximum power output of 120W. This will allow for high current motors to be used in addition to powering most environmental sensors. The board is based off the LTC3807 switching regulator. The board is able to convert an input voltage of 14-42V and convert to 12V at up to 95% efficiency. The use of a full ground plane and metal standoffs acts as a sufficient heat sink for up to 5A power draw. If the heat generation is too high a heat sink can be applied to the ground plane.

1.2.2.2 3.3V/5V buck converter

The 3.3/5V buck converter is based on the LT8650S, a duel output switching regulator. The converter receives between 6.8V and 40V input and outputs 3.3V at 4.5A and 5V at 4.5A simultaneously. Due to fabrication issues the board is still in development. In the mean time a board using linear regulators was designed instead.

1.2.2.3 Motor controller

The motor controller is comprised is designed to run 12V motors at up to 3.6A per motor using DVR8871D chips. The pwm signal is provided by a PCA9685 PWM driver, controllable over i^2c. A PWM driver is used so that any micro-controller can be used, without concern for using up 6 PWM pins. For feedback the encoder pins are fed through an MC14045B level shifter, converting the 5V encoder ticks to the 3.3V logic most micro-controllers use.

2 Dynamics and Kinematic Models

Multiple models of the robot have been developed. The first is a slip-space dynamic model which was derived for a four wheeled platform based on the slip-space model of a three wheeled platform. The second is an already established linear model which accounts for friction, restated here for

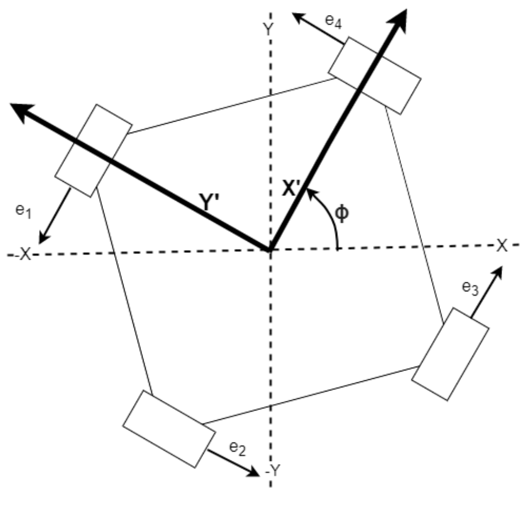

Fig. 3. Coordinate diagram of robot

completeness. Finally there is a kinematic model of a four wheeled platform which accounts for wheel slip.

2.1 Slip-space Dynamic Model

2.1.1 Definition of Coordinate

We start by defining three coordinate systems, a fixed system, a rotating frame, and a wheel based frame. Figure 3 shows all frames with respect to the robot. The transformations between frames is described as follows.

$$\begin{aligned} e_x' &= cos\phi e_x + sin\phi e_y \\ e_y' &= -sin\phi e_x + cos\phi e_y \\ e_z' &= e_z \\ e_1 &= e_y' \\ e_2 &= -e_x' \\ e_3 &= -e_y' \\ e_4 &= e_x' \end{aligned} \tag{1}$$

2.1.2 Euler-Lagrange Equation

The derivation of the dynamics is based on [**?**] which derived the dynamics for a 3 wheeled robot. Instead we now do the same derivation but for a 4 wheeled robot. Using the Euler-Lagrange method we can describe the dynamics of the system.

$$\frac{\delta}{\delta t}\frac{\delta L}{\delta \dot{q}_i} - \frac{\delta L}{\delta q_i} = F_i \tag{2}$$

where q are the generalized coordinates of the system, and F is the external force acting in the direction of a given coordinate. In the case of this system the coordinates are x, y, ϕ, and θ_i, the angular position of each wheel. Since there is no potential energy in the system, the Lagrangian is equal to the kinetic energy of the system, described as follows.

$$L = \frac{1}{2}\left[m(\dot{x}^2 + \dot{y}^2) + I_R\dot{\phi}^2 + \sum_{i=1}^{4} I_w \dot{\theta_i}^2\right] \tag{3}$$

where m is the total robot mass, I_R is the robots moment of inertia about the z-axis, and I_w is each wheels moment of inertia about their axes of rotation. Now we define the external forces on the robot.

$$\begin{aligned} F_q &= \underline{Q}(\phi) f_t \\ F_\theta &= \tau - r_w f_t \end{aligned} \tag{4}$$

where τ is the torque vector, r_w is the radius of the wheels, f_t is the traction force between each wheel and the ground, and $Q(\phi)$ is defined as follows.

$$\underline{Q}(\phi) = \begin{bmatrix} -\sin(\phi) & -\sin(\phi+\frac{\pi}{2}) & -\sin(\phi+\pi) & -\sin(\phi+\frac{3\pi}{2}) \\ cos(\phi) & cos(\phi+\frac{\pi}{2}) & cos(\phi+\pi) & cos(\phi+\frac{3\pi}{2}) \\ R & R & R & R \end{bmatrix} \tag{5}$$

Finally, substituting (4) and (3) into (2) we get the dynamics of the system.

$$M\ddot{q} = Q(\phi) f_t \tag{6}$$

$$I_w\ddot{\theta} = \tau - r_w f_t \tag{7}$$

where

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_R \end{bmatrix} \tag{8}$$

2.1.3 Slip Space Conversion

Now that we have the dynamics described we can convert from the (q,θ) space to the (q,s) space, where s is the slip coordinate of each wheel. The slip is defined in (9).

$$s_i = \frac{r_w\dot{\theta}_i - v_{w_i}}{v_{w_i}} \tag{9}$$

$$\tag{10}$$

where $v_{w_i} = v \cdot e_i + r\dot{\phi}$ is the velocity of a wheel in the worlds frame. This can be described in matrix form by using the transformations described in (1).

$$v_w = RT(\phi)\dot{q} \tag{11}$$

where

$$R = \begin{bmatrix} 0 & 1 & r \\ -1 & 0 & r \\ 0 & -1 & r \\ 1 & 0 & r \end{bmatrix}$$

$$T(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and by taking the derivative we can get the acceleration of the wheels.

$$\dot{v}_w = RT(\phi)\ddot{q} + R\frac{dT(\phi)}{d\phi}\dot{q} \tag{12}$$

Now in order to finish the transformation from standard to slip space, we look at the derivative of the slip variable. By rearranging the derivative and substituting it into (7) we can arrive at the dynamics of the system in our new coordinates.

$$\begin{aligned} \dot{s}_i &= \frac{r_w\ddot{\theta}_i - \dot{v}_{w_i}}{v_{w_i}} - \frac{(r_w\dot{\theta}_i - v_{w_i})\dot{v}_{w_i}}{v_{w_i}^2} \\ &= \frac{r_w\ddot{\theta}_i v_{w_i} - r_w\dot{\theta}_i\dot{v}_{w_i}}{v_{w_i}^2} \\ v_{w_i}\dot{s}_i &= \left(\frac{r_w}{I_w}\tau_i - \frac{r_w^2}{I_w}f_{t_i} - (s_i+1)\dot{v}_{w_i}\right) \end{aligned}$$

or substituting in (6) and rewriting in matrix form

$$V_w\dot{s} = \frac{r_w}{I_w}\left(\tau - M^*(s)\ddot{q} - N^*\dot{q}\right) \tag{13}$$

where

$$\begin{aligned} N^* &= \frac{I_w}{r_w}(S+I)R\frac{dT(\phi)}{d\phi} \\ M^* &= rQ^{-1}(\phi)M + \frac{I_w}{r_w}(S+I)RT(\phi) \\ S &= diag(s) \\ V_w &= diag(v_w) \end{aligned}$$

Equation (13) describes the dynamics of the system allowing for a controller to be developed for it.

2.2 Linearized Dynamic Model

The dynamics of an omni-wheel robot as described by Helder Oliveira et. al. are as follows [**?**]:

$$\dot{\mathbf{x}}_c = \mathbf{A}_c\mathbf{x} + \mathbf{B}_c\mathbf{u} + \mathbf{K}_c sign(\mathbf{x}) \tag{14}$$

$$\begin{aligned} \mathbf{A}_c &= \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{2K_t^2l^2}{r^2RM} - \frac{B_v}{M} & 0 & 0 \\ 0 & -\frac{2K_t^2l^2}{r^2RM} - \frac{B_{vn}}{M} & 0 \\ 0 & 0 & -\frac{4d^2K_t^2l^2}{r^2RJ} - \frac{B_v}{J} \end{bmatrix} \end{aligned}$$

$$\mathbf{B}_c = \frac{lK_t}{rR}\begin{bmatrix} 0 & -\frac{1}{M} & 0 & \frac{1}{M} \\ \frac{1}{M} & 0 & -\frac{1}{M} & 0 \\ \frac{d}{J} & \frac{d}{J} & \frac{d}{J} & \frac{d}{J} \end{bmatrix}$$

$$\mathbf{K}_c = \begin{bmatrix} -\frac{C_v}{M} & 0 & 0 \\ 0 & -\frac{C_{vn}}{M} & 0 \\ 0 & 0 & -\frac{C_w}{J} \end{bmatrix}$$

where K_t is the motor torque constant; l is the gearbox reduction; M is the mass of the robot; r is the radius of each wheel; R is the motor resistance; B_v, B_{vn}, and B_w are the viscous frictions; J is the moment of inertia of the robot; d is the distance between the center of the robot and each wheel; and C_v, C_{vn}, and C_w are the Coulomb frictions.

2.3 Kinematic Model

The kinematic model can be derived using the constraint on Swedish wheels as described in (15)

$$\begin{bmatrix} \sin(\alpha_i) \\ -\cos(\alpha+\beta+\gamma) \\ -l\cos(\beta+\gamma) \end{bmatrix}^T \mathbf{R}(\theta)\dot{\xi}_I - r\dot{\phi}\cos\gamma = v_s \tag{15}$$

where α is the angle of the wheel's center from the positive x axis in the robots frame of reference, β is the mounting angle of the wheel with respect to α, γ is the angle of the rollers with respect to the wheels plane, l is the distance from the robot's center of mass to the center of the wheel, $\mathbf{R}(\theta)$ is a rotation matrix described in (16), $\dot{\xi}_I$ is the state velocities in the global frame described in (17), r is the radius of the wheel, $\dot{\phi}$ is the angular velocity of the wheel, and v_s is the slip velocity in the direction of the wheel defined in (21). Figure 4 displays the coordinates in terms of the robot's frame of reference.

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

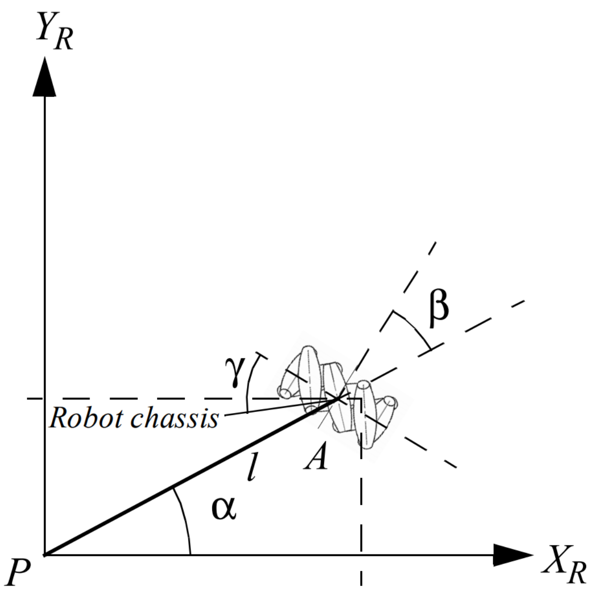

Fig. 4. Diagram of a Swedish wheel and it's corresponding coordinates [?]

$$\dot{\xi}_I = \begin{bmatrix}\dot{x} \; \dot{y} \; \dot{\theta}\end{bmatrix}^T \tag{17}$$

$$v_s = r\dot{\phi} - v_{wheel} \tag{18}$$

Using (15) and the geometry of the robot, the kinematics can be derived. The wheels being used are standard 90 degree Swedish wheels meaning $\gamma = 0$, and since the wheels are mounted radially $\beta = 0$. For each wheel i the constraint simplifies to (19)

$$\begin{bmatrix}\sin\alpha_i \\ -\cos\alpha_i \\ -l\end{bmatrix}^T \mathbf{R}(\theta)\,\dot{\xi}_I - r\dot{\phi}_i = v_{s_i} \tag{19}$$

where $\alpha_i = \frac{2i\pi - \pi}{4}$ for $i = 1..4$. Making the substitution the kinematics of the robot in a global frame are described by (20)

$$\mathbf{u} = \begin{bmatrix}\dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4\end{bmatrix} = \frac{1}{r}\left(\mathbf{B}(\theta)\,\dot{\xi}_I - \mathbf{v}_s\right) \tag{20}$$

$$\mathbf{B}(\theta) = \frac{\sqrt{2}}{2}\begin{bmatrix}\cos\theta + \sin\theta & -\cos\theta + \sin\theta & l \\ \cos\theta - \sin\theta & \cos\theta + \sin\theta & l \\ -\cos\theta - \sin\theta & \cos\theta - \sin\theta & l \\ -\cos\theta + \sin\theta & -\cos\theta - \sin\theta & l\end{bmatrix} \tag{21}$$

where $\mathbf{u}$ contains the inputs for the system. In order to develop a controller it is more useful to use the inverse kinematic model described by (22)

$$\dot{\xi}_I = \mathbf{B}^{-1}(\theta)\,(r \cdot \mathbf{u} + \mathbf{v}_s) \tag{22}$$

Finally, the inverse kinematic model can be converted into a discrete time form, making it more suitable for control on a discrete system. This takes the form of (23).

$$\xi_{I_{i+1}} = \xi_{I_i} + \mathbf{B}^{-1}(\theta)_i\,(r \cdot \mathbf{u}_{i+1} + \mathbf{v}_{s_i}) \cdot T \tag{23}$$

where i denotes the current time step, and T denotes the length of each time step. Given this form, controllers can be developed for the system

2.4 Motor Dynamics

It will be necessary later to design a controller for the motors in conjunction with a controller for the robot, so the motor dynamics are derived. The dynamics can be described based on Newton's second law and Kirchhoff's voltage law as seen in (24) and (25) respectively.

$$J\ddot{\theta} + b\dot{\theta} = Ki \tag{24}$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta} \tag{25}$$

where J is the moment of inertia of the motor, θ is the angular position of the motor shaft about the shaft axis, b is the motors viscous friction, K is the motor's torque constant, i is the current provided to the motor, L is the inductance of the motor, R is the resistance of the motor, and V is the voltage provided to the motor. From these equations the full dynamics can be written out in (26) using the observer shown in (27).

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{L}\end{bmatrix}V \tag{26}$$

$$\mathbf{z}_{motor} = \begin{bmatrix}1 \; 0\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} \tag{27}$$

3 Controllers

A number of controllers have been developed and tested to various extents using each of the models described above. These controllers are described and the results of testing shown in this section.

3.1 Direct Torque Controller

Using the slip-space dynamic model a direct torque controller can be derived using feedback linearization.

$$\tau = M^*(\ddot{q}_d - u) + N^*\dot{q} \tag{28}$$

where $\ddot{q}_d$ is the desired acceleration vector. By substituting equation 28 into equation **??** we get the error dynamics of $\ddot{e} = u$ where $e = q_d - q$. Now that the system has been linearized we can set u to be a PD controller of the following form.

$$u = -K_d\dot{e} - K_p e \tag{29}$$

where K_d and K_p are design matrices which can be adjusted in order to properly fit the system.

Due to the use of brushed DC motors, it is difficult to directly control the torque, as it would require a constant voltage variable current supply for each wheel. Instead pulse width modulation (PWM) is used to indirectly control the torque. The relationship between torque and PWM is roughly linear, with some non-linearity similar to a saturation curve at low PWM values. This relationship was determined experimentally using a shunt resistor to determine current and an oscilloscope to determine voltage. By sending PWM values to the robot, the PWM to power curve was determined, which then was converted to a PWM to torque curve, using the power to torque curve given in the motor data sheet. Conveniently, due to the load the robot places on the motors, the robot is unable to move at low PWM values, so only the linear relationship will be necessary.

The relationship between torque and PWM can be described as follows.

$$\text{PWM} = \begin{cases} \frac{255\tau}{\tau_{max}} & \forall|\gamma \in \tau| \leq \tau_{max} \\ \frac{255\tau}{absmax(\tau)} & \exists|\gamma \in \tau| > \tau_{max} \end{cases} \tag{30}$$

Where τ_{max} is the largest producible torque of the weakest wheel, and $absmax(\tau)$ is the largest absolute element of τ. Through this method, the PWM value will linearly track with the computed torque while the torque is less than the maximum producible torque. If the computed torque for any wheel is large than producible, the torque is capped and the relative torque is maintained.

3.1.1 Results

Due to difficulties working with the VICON system, experimental results were not obtained. The main issue was slow communication and an error in the code causing the PWM value to be sent to the wrong motors. This issue has since been fixed. Instead simulated results are presented.

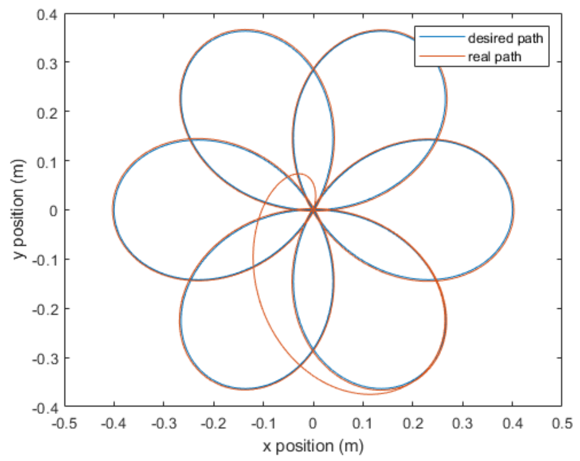

Fig. 5. Position plot with the robot tracking a six pedal rose trajectory

The system was simulated using a backward Euler solver, the code of which can be found in Appendix A. Design of the control matrices K_d and K_p was conducted through trial and error. Since we know that for stability the matrices must be negative definite, the matrices were designed to be diagonal dominant with fully negative diagonals. Through trial and error the control matrices were defined as follows.

$$K_d = \begin{bmatrix} -4.1 & 0 & 0 \\ 0 & -4.1 & 0 \\ 0 & 0 & -4.1 \end{bmatrix} \tag{31}$$

$$K_p = \begin{bmatrix} -5 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & -5 \end{bmatrix} \tag{32}$$

The simulation was run for a number of cases, including the robot following a linear path, moving to random points, following a circular path, and following the path of rose plots with between three and eight pedals. Since the robots main function it to maintain orientation while translating, for all but the random point testing the robots angle was set to zero. The robot was always assumed to start at the origin. Figure 5 and 6 show the position and error of the robot during a 6 pedal rose trajectory, respectively. It should be noted that the simulation took into account the maximum torque per wheel. The controller will try to converge much faster than possible, so the simulation capped the maximum torque according to equation 30.

Figure 7 and 8 shows the position and error of the robot when given random points to reach resulting in random linear trajectories. The slow time response is purely due to the physical limitations of the robot, as the torque is being limited to match how the robot would act in the real world.

3.2 Kinematic Feedback Linearization

Using the kinematic model of the robot, a different feedback linearization controller can be created. It is important

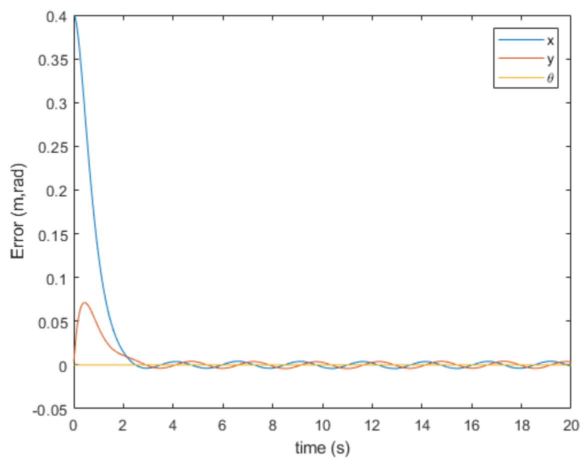

Fig. 6. Error plot with the robot tracking a six pedal rose trajectory

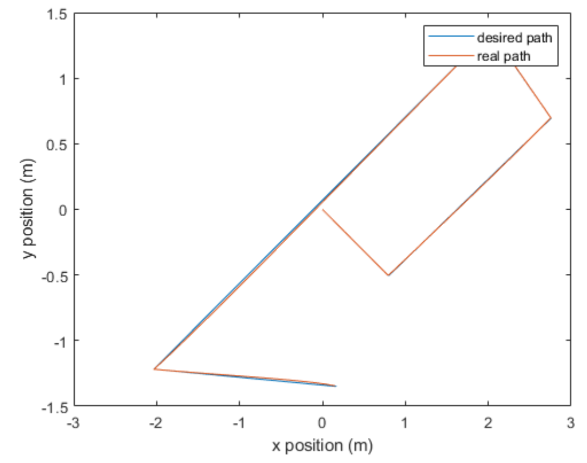

Fig. 7. Position plot with the robot tracking a set of random linear trajectories

to note that it is assumed the PWM linearly affects the angular velocity of each wheel and it acts on the wheel instantaneously. This assumption is unrealistic and will be dealt with in a future controller.

The feedback linearization controller assumes a no-slip condition. Doing so simplifies (23) which after converting to error dynamics resulting in (33).

$$\mathbf{e}_{i+1} = \mathbf{e}_i + r\mathbf{B}^{-1}(\theta)_i\,\mathbf{u}_{i+1} \cdot T \tag{33}$$

where $\mathbf{e}_i = \xi_{I_i} - \xi_d$ and ξ_d is the desired state for the system. Linearization is accomplished easily by defining $\mathbf{u}_{i+1}$ using a linear controller $\mathbf{v}$ as seen in (34) resulting in (35)

$$\mathbf{u}_{i+1} = \frac{1}{rT}\mathbf{B}_i(\theta)\,\mathbf{v}_{i+1} \tag{34}$$

$$\mathbf{e}_{i+1} = \mathbf{e}_i + \mathbf{v}_{i+1} \tag{35}$$

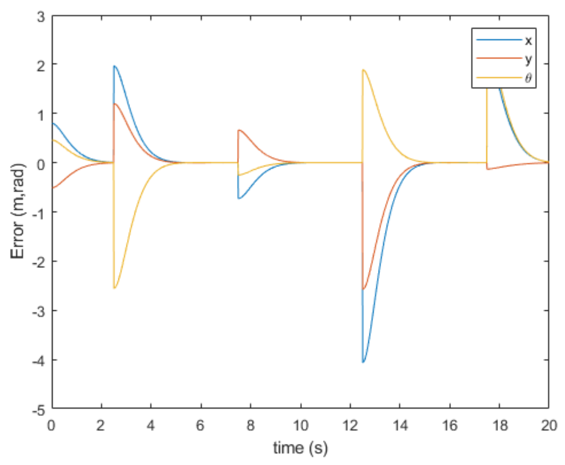

Fig. 8. Error plot with the robot tracking a set of random linear trajectories

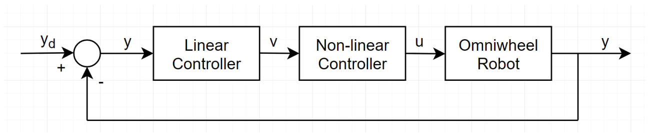

Fig. 9. Block diagram of controller using feedback linearization

From (35) it is obvious that the controller **v** must have an opposite sign to the error in order to have the error converge to 0. A P, PD, and PID controller were developed for v taking the form of (36), (37), and (38) respectively.

$$\mathbf{v}_{i+1} = -\mathbf{K}_p\mathbf{e}_i \tag{36}$$

$$\mathbf{v}_{i+1} = -\mathbf{K}_p\mathbf{e}_i - \mathbf{K}_d\dot{\mathbf{e}}_i \tag{37}$$

$$\mathbf{v}_{i+1} = -\mathbf{K}_p\mathbf{e}_i - \mathbf{K}_d\dot{\mathbf{e}}_i - \mathbf{K}_s\sum_{n=0}^{i}\mathbf{e}_n \cdot T \tag{38}$$

The architecture of the controller can be seen in figure 9. Each controller was implemented on the four wheel drive omniwheel robot using a VICON tracking system. The controller was implemented at 50Hz. The robot was placed at random between 1-3 meters of the origin and instructed stabilize, this was repeated five times for each controller. The average accuracy of each controller is listed in table 3.2. It should be noted that the PID controller would likely have better accuracy with a change of motors, the robots motors are unable to make small movements, resulting in an inability to finish stabilizing without inducing oscillations.

3.3 Integral Sliding Mode Control

The integral sliding mode controller does not assume a no-slip condition like the feedback linearized controller however it does still assume the system Since the system is first

Controller	Absolute Accuracy (cm)
P	10 ± 3
PD	4 ± 1
PID	3 ± 1

Table 1. Accuracy of each controller during stabilization maneuver

order it is logical to use an integral sliding mode controller where the sliding surface is defined in (39)

$$\mathbf{s} = \mathbf{e} + \lambda \int \mathbf{e} \cdot dt \tag{39}$$

where $\mathbf{s}$ is the sliding surface, $\lambda \in \mathbb{R}^{3x3}$ is a gain matrix, and $\mathbf{e} = \xi_I - \xi_d$. The first half of the controller can be found by taking the derivative of the sliding surface, show in (40), and rearranging for the control vector $\hat{\mathbf{u}}$ as seen in (41).

$$\dot{\mathbf{s}} = \dot{\mathbf{e}} + \lambda\mathbf{e} = \mathbf{B}(\theta)^{-1}(r \cdot \hat{\mathbf{u}} + \hat{\mathbf{v}}_s) - \xi_d + \lambda\mathbf{e} = 0 \tag{40}$$

$$\hat{\mathbf{u}} = \frac{1}{r}\left(\mathbf{B}(\theta)\left(-\lambda\mathbf{e} + \dot{\xi}_d\right) - \hat{\mathbf{v}}_s\right) \tag{41}$$

where $\hat{\mathbf{v}}_s$ is a vector containing the estimated slip of each wheel.

In order to maintain the system on the sliding surface the second half of the controller is defined as (42).

$$\mathbf{u}_c = -\mathbf{B}(\theta)\mathbf{k}\text{sign}(\mathbf{s}) = -\mathbf{B}(\theta)\mathbf{k}\text{sign}\left(\mathbf{e} + \lambda \int \mathbf{e} \cdot dt\right) \tag{42}$$

where $\mathbf{k} \in \mathbb{R}^{3x3}$ is a gain matrix. Combining, the full controller is shown in (43).

$$\begin{aligned} \mathbf{u} &= \hat{\mathbf{u}} + \mathbf{u}_c \\ &= \frac{1}{r}\left(\mathbf{B}(\theta)\left(-\lambda\mathbf{e} + \dot{\xi}_d\right) - \hat{\mathbf{v}}_s\right) - \mathbf{B}(\theta)\mathbf{k}\text{sign}\left(\mathbf{e} + \lambda \int \mathbf{e} \cdot dt\right) \end{aligned} \tag{43}$$

A Lyupanov function is defined as $V = \frac{1}{2}\mathbf{s}^2$ showing the system is controllable. A constraint on $\mathbf{k}$ can be found using the derivative of V as shown in (44).

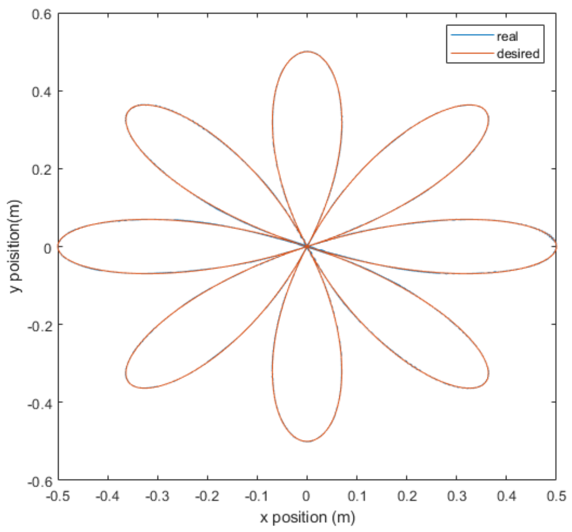

Fig. 10. Robot tracking an 8-pedal rose while rotating

Controller	RMSE (m)
x	0.0008
y	0.0007
theta	0.0056

Table 2. RMSE of each state during simulation using an integral sliding mode controller

$$\begin{aligned} \frac{dV}{dt} &\leq \eta|\mathbf{s}| \\ \mathbf{s}^T\dot{\mathbf{s}} &\leq \eta|\mathbf{s}| \\ \mathbf{s}^T\left[\mathbf{B}(\theta)^{-1}(\mathbf{v}_s - \hat{\mathbf{v}}_s) - \mathbf{k}\text{sign}(\mathbf{s})\right] &\leq \eta|\mathbf{s}| \\ \mathbf{k} &\geq \eta + \mathbf{B}(\theta)^{-1}(\mathbf{v}_s - \hat{\mathbf{v}}_s) \end{aligned} \tag{44}$$

where η is a small constant. From this derivation it can be seen that $\mathbf{s}^T\mathbf{k} = \mathbf{k}\mathbf{s}^T$ which can be achieved easily by making $\mathbf{k}$ a symmetric matrix.

3.3.1 Simulations

The controller was simulated and tuned in Matlab over a variety of trajectories. For all simulations the slip velocity was given up to a 10% error from the true slip. Figure 10 shows the robot tracking an 8 pedal rose plot while table 3.3.1 displays the RMSE of each state. The ISMC has good tracking in simulation although testing on the platform must be conducted to meaningfully compare the results to those of the feedback linearization controller.

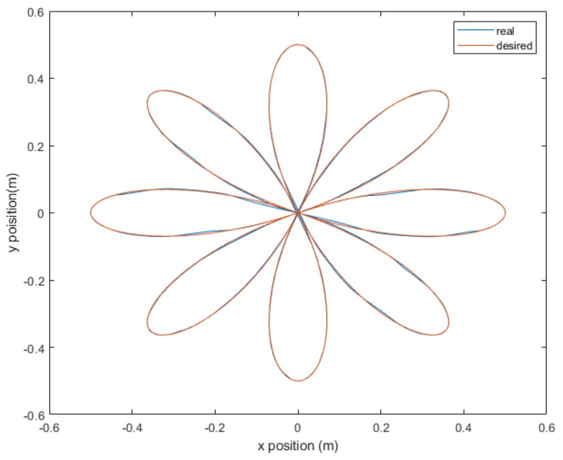

Fig. 11. Tracking of a rose plot trajectory using an ISMC with a motor controller on each wheel while rotating

Controller	RMSE (m)
x	0.017
y	0.016
theta	0.18

Table 3. RMSE of each state during simulation using an integral sliding mode controller

3.4 ISMC with Motor Control

A major issues with the ISMC is it's assumption that PWM directly affects the rpm of the wheels instantaneously. Testing shows this is not the case, so it becomes important to implement a controller for each motor. Using the motor dynamics a PID controller can be developed and tuned. Doing so allows the robot to accurately follow an arbitrary path such as a rose plot (figure 11) with a 10% error in estimation of wheel slip with an RMSE displayed in table 3.4. A block diagram of the controller can be seen in figure 3.4.

The tracking of this controller is worse than the ISMC controller due to it being a more accurate representation of how the motors will act. Feeding the ISMC controller directly into the motor dynamics resulted in a nearly uncontrollable system making this new controller a significant improvement which will be implemented on the robot.

4 Conclusion

So far a number of dynamic and kinematic models have been investigated by designing controllers around them. In the current state the best controller in theory is the integral sliding mode controller with a PID controller on each motor, however the best method tested on a robot so far is the feedback linearization controller based on the robot kinematics. However even the feedback linearization controller was only stabilizing and it is yet to be seen how it performs in trajectory tracking. In the future the feedback linearization and ISMC controllers must be implemented on the new three wheeled platform in order to accurately measure it's effectiveness. In addition predictive controls should be developed to increase accuracy.

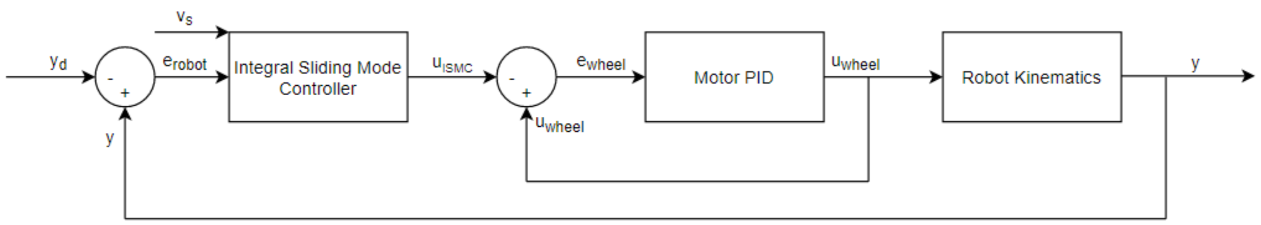

Fig. 12. Block diagram of ISMC controller with a motor PID sub-controller

A Future additions to this document

A number of additions need to be made to this document including but not limited to...

1. Adding more info to the board development, each board could be a section
2. Add schematics of all board (and pictures)
3. Add more pictures of robot
4. Rerun RMSE calculations over 1000 simulations
5. General editing
6. Add literature review
7. Add comprehensive abstract
8. Expand on ways to more accurately predict variables for LSM

B Kalman Filters

In order to accurately determine the position of the robot a set of filters can be used. A Kalman filter is applied to the output from the encoders to gain more accurate measurements of the motors angular velocities. This filtered data is then fed into the kinematic model of the omniwheel robot where an EKF is applied to adjust for any additional noise and give a more accurate position measurement. For comparison the EKF and KF are run separately as well.

B.1 Motor Kalman Filter

In order to reduce computational complexity it makes sense to have a single filter which applies to all four motors simultaneously. In order to do this we simply combine the motor dynamics from (26) and (27) and include noise terms in a standard state space equation described in (45).

$$\mathbf{x}_{k+1} = (\mathbf{A}_m + \mathbf{I})\,\mathbf{x}_k + (\mathbf{B}_m \mathbf{V}_k + \mathbf{w}_k)\,T \tag{45}$$
$$\mathbf{z}_{k+1} = \mathbf{C}_m \mathbf{x}_{k+1} + \mathbf{v}_k \tag{46}$$

$$\mathbf{x} = \left[\dot{\theta}_1\ i_1\ \dot{\theta}_2\ i_2\ \dot{\theta}_3\ i_3\ \dot{\theta}_4\ i_4\right]^T \tag{47}$$

$$\mathbf{A}_m = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{K}{L} & -\frac{R}{L} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{b}{J} & \frac{K}{J} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{K}{L} & -\frac{R}{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{b}{J} & \frac{K}{J} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{K}{L} & -\frac{R}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{b}{J} & \frac{K}{J} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \tag{48}$$

$$\mathbf{B}_m = diag\left(\left[0\ \tfrac{1}{L}\ 0\ \tfrac{1}{L}\ 0\ \tfrac{1}{L}\ 0\ \tfrac{1}{L}\right]\right) \tag{49}$$

$$\mathbf{V} = \left[V_1\ V_2\ V_3\ V_4\right]^T \tag{50}$$

$$\mathbf{C}_m = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0 \end{bmatrix} \tag{51}$$

where $\mathbf{w}_k$ is a vector for each motors input noise defined as $\mathbf{w}_k = \mathcal{N}(0, \mathbf{Q}_k)$ with $\mathbf{Q}_k$ being the input noise co-variance matrix, $\mathbf{v}$ is a vector composed of the measurement noise for each motor defined as $\mathbf{v}_k = \mathcal{N}(0, \mathbf{R}_k)$ with $\mathbf{R}_k$ being the measurement noise co-variance matrix, and T is the time step of the system.

Implementation of the Kalman filter was completed in a standard form, using a time update and a measurement update defined in (52) -(53) and (54)-(56) respectively.

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_m \hat{\mathbf{x}}_{k|k} \tag{52}$$
$$\mathbf{P}_{k+1|k} = \mathbf{A}_m \mathbf{P}_{k|k} \mathbf{A}_m^T + \mathbf{Q} \tag{53}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\left[\mathbf{z}_{k+1} - \mathbf{C}_m\left(\hat{\mathbf{x}}_{k+1|k}\right)\right] \tag{54}$$
$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}_m^T\left[\mathbf{C}_m \mathbf{P}_{k+1|k}\mathbf{C}_m^T + \mathbf{R}\right] \tag{55}$$
$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C}_m)\,\mathbf{P}_{k+1|k} \tag{56}$$

where $\hat{\mathbf{x}}_{k+1|k+1}$ is the estimation of the state vector, $\hat{\mathbf{x}}_{k+1|k}$ is the prediction of the state vector, $\mathbf{K}_{k+1}$ is the Kalman gain, $\mathbf{P}_{k+1|k}$ is the prediction of the error co-variance, and $\mathbf{P}_{k+1|k+1}$ is the updated error co-variance. With this the Kalman filter can be applied to the motors.

B.2 Robot Extended Kalman Filter

While the Kalman filter is sufficient for estimation of the motors, the non-linearity of the omni-wheel kinematics would cause a Kalman filter to be insufficient. As such an extended Kalman filter (EKF) can be implemented in it's place, using both the non-linear kinematics and a linearized version of them. First the discrete version of the system from (23) is redefined and non-linear functions are created shown in (57)-(60).

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_{k+1}\right) + \mathbf{w}_{k+1} \tag{57}$$

$$\mathbf{z}_{k+1} = h\left(\mathbf{x}_{k+1}\right) + \mathbf{v}_{k+1} \tag{58}$$

$$f\left(\mathbf{x}_k, \mathbf{u}_{k+1}\right) = \xi_{I_i} + \mathbf{B}^{-1}(\theta)_i \left(r \cdot \mathbf{u}_{i+1} + \mathbf{v}_{s_i}\right) \cdot T \tag{59}$$

$$h\left(\mathbf{x}_{k+1}\right) = \mathbf{x}_{k+1} \tag{60}$$

With the state-space kinematics redefined in terms of non-linear functions the EKF can be applied. The method first predicts the state and then updates the state estimates as seen in equations (61)-(62) and (63)-(67)

$$\hat{\mathbf{x}}_{k+1|k} = f\left(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_{k+1}\right) \tag{61}$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_{k+1}\mathbf{P}_{k|k}\mathbf{F}_{k+1}^T + \mathbf{Q}_{k+1} \tag{62}$$

$$\widetilde{\mathbf{y}}_{k+1} = \mathbf{z}_{k+1} - h\left(\hat{\mathbf{x}}_{k+1|k}\right) \tag{63}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1} \tag{64}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T\mathbf{S}^{-1}{}_{k+1} \tag{65}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\widetilde{\mathbf{y}}_{k+1} \tag{66}$$

$$\mathbf{P}_{k+1|k+1} = \left(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\right)\mathbf{P}_{k+1|k} \tag{67}$$

where $\hat{\mathbf{x}}_{k+1|k}$ is the predicted state estimate, $\mathbf{P}_{k+1|k}$ is the predicted co-variance estimate, $\widetilde{\mathbf{y}}_{k+1}$ is the innovation residual, $\mathbf{S}_{k+1}$ is the innovation co-variance, $\mathbf{K}_{k+1}$ is the near optimal Kalman gain, $\hat{\mathbf{x}}_{k+1|k+1}$ is the updated state estimate, $\mathbf{P}_{k+1|k+1}$ is the updated co-variance estimate, $\mathbf{F}_{k+1}$ is the Jacobian of f as defined in (68), and $\mathbf{H}_{k+1}$ is the Jacobian of h as defined in (69).

$$\mathbf{F}_{k+1} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k} \tag{68}$$

$$\mathbf{H}_{k+1} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k+1|k}} \tag{69}$$

It is important to make note of how the input noise co-variance is determined. While it would be easier to include the slip velocity of each wheel in the co-variance, this is not accurate. Slip is cannot be described using a normal distribution since it is highly dependant on the velocity of the robot as well as the environment the robot is traversing. For this reason the slip must be dealt with by the controller.

B.3 Controller

The control for the robot uses an integral sliding mode controller allowing for the slip term to be handled separately without the need to include it in the filtering process. Since slip of the robot is expected to be small, the controller always assumes the slip of each wheel is zero, and reactively adjusts for the error. The tuned design matrices are shown in (70) and (71).

$$\lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{70}$$

$$\mathbf{k} = \begin{bmatrix} 2.8 & 0 & 0 \\ 0 & 2.8 & 0 \\ 0 & 0 & 1.4 \end{bmatrix} \tag{71}$$

B.4 Simulations

Due to not having access to an area to test the robot, all filters were instead simulated using Matlab. For comparison there are four filter combinations used. The simulation was run using no filters, only Kalman filtering on the motors (KF), only the EKF on the robot (EKF), and both the Kalman filter and the EKF in tandem (EKF+KF).

The robot was instructed to follow both an arbitrary path defined in (81) and a parametric rose. The arbitrary path allows for long simulations with little overlap of the path, while the parametric rose paths allow for easily graphable plots. During the parametric rose path, the robot was set to follow a sinusoidal path in the θ coordinate. Since the effectiveness of the controller is not a concern, the robot is assumed to move under a no-slip condition for all simulations.

A number of physical constants must be defined to feed into the kinematic and dynamic models. These values are listed in table 6. The input and measurement noise co-variance of the motors are defined in (77) and (78). The input and measurement noise co-variance of the robot are defined in (79) and (80). In both cases the noise is assumed to be dependant only on it's respective state resulting in diagonal matrices.

Variable	Symbol	Value	Units
Wheel radius	r	3.275	cm
Robot Radius	l	19.5	cm
Motor Resistance	R	1	Ω
Motor Inductance	L	0.1	H
Motor Moment of Inertia	J	0.01	$N \cdot m^2$
Motor Viscous Friction	b	0.1	$N \cdot m \cdot s$
Motor Constant	K	0.01	$N \cdot m/A$
Time step	T	0.02	s

Table 4. List of physical constants needed to describe system kinematics and dynamics

$$\mathbf{Q}_{motor} = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix} \tag{72}$$

$$\mathbf{R}_{motor} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{73}$$

$$\mathbf{Q}_{robot} = \begin{bmatrix} 4e-5 & 0 & 0 \\ 0 & 4e-5 & 0 \\ 0 & 0 & 4e-5 \end{bmatrix} \tag{74}$$

$$\mathbf{R}_{robot} = \begin{bmatrix} 5e-2 & 0 & 0 \\ 0 & 5e-2 & 0 \\ 0 & 0 & 5e-2 \end{bmatrix} \tag{75}$$

B.5 Results

The robot was instructed to follow an arbitrary path defined in (81) over the course of 600 seconds. This test was then repeated 100 times using each combination of filters and the RMSEs of each test was averaged, the results of which are shown in table 7.

$$\xi_d = 0.5 \begin{bmatrix} \sin(t \cdot 2\pi/12) + \cos(t \cdot 2\pi/14) \\ \sin(t \cdot 2\pi/5) + \cos(t \cdot 2\pi/4) \\ \sin(t \cdot 2\pi/16) + \cos(t \cdot 2\pi/8) \end{bmatrix} \tag{76}$$

The results show there is only a slight increase in performance between using Kalman filters on the motors to feed into the robots EKF vs using the EKF by itself. When implemented on a robot, the extra filtering step should only be used if there is excess computational power that would never be needed elsewhere. Using only a Kalman filter on the wheels and no EKF on the robot kinematics showed markedly worse performance with over 10cm RMSE, although it is still better than no filtering at all. While using only the KFs is significantly worse than using an EKF, the KF requires much less computational power and could still be useful for low powered operations where positional accuracy is less important.

RMSE	No Filter	KF	EKF	KF and EKF
x (m)	0.1314	0.1234	0.0071	0.0067
y (m)	0.1069	0.1263	0.0070	0.0067
θ (rad)	0.1076	0.0927	0.0078	0.0074

Table 5. State RMSEs during no filtering, kalman filtering on each motor, an EKF on the robot kinematics, and both filters combined. The test was run using the path described in (81) over 600 seconds. The test was repeated 1000 times and the RMSE values averaged.

Results from running the robot on a 4-petal rose trajectory can be seen in figures 13 and 15. Figures 14 and 16 show zoomed versions of a small section of the previous figures in order to more clearly display the accuracy of each controller. It is obvious that the EKF and KF+EKF cases have very similar results while the KF and no filter cases lag behind, confirming the results of Table 7. It can also be noted that in all case the filters have a tendency to under estimate the position of the robot with 74% of estimates being closer to the origin than the true robot position.

It should be noted that the reason for the true path oscillating about a perfect rose plot is due to the controller and the motors physical constants. The constants for the motor were chosen to make the controller easily tune-able, however this resulted in motors with slow response times. Implementation of PID controllers on each motor would allow for more accurate values to be used, and therefore a better controller to be developed. In the future, data must be collected on the actual system to determine all parameters more accurately. Since said implementation would require significantly more computational power, it would make sense to avoid Kalman filters on each wheel to reduce computation time.

C Simulations

Due to not having access to an area to test the robot, all filters were instead simulated using Matlab. For comparison there are four filter combinations used. The simulation was run using no filters, only Kalman filtering on the motors (KF), only the EKF on the robot (EKF), and both the Kalman filter and the EKF in tandem (EKF+KF).

The robot was instructed to follow both an arbitrary path defined in (81) and a parametric rose. The arbitrary path allows for long simulations with little overlap of the path, while the parametric rose paths allow for easily graphable

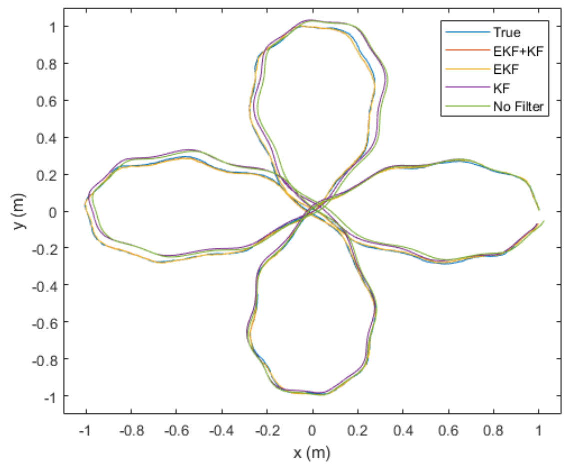

Fig. 13.   Filter results for the robot's position during a 4 petal rose path over 30 seconds using all filter combinations

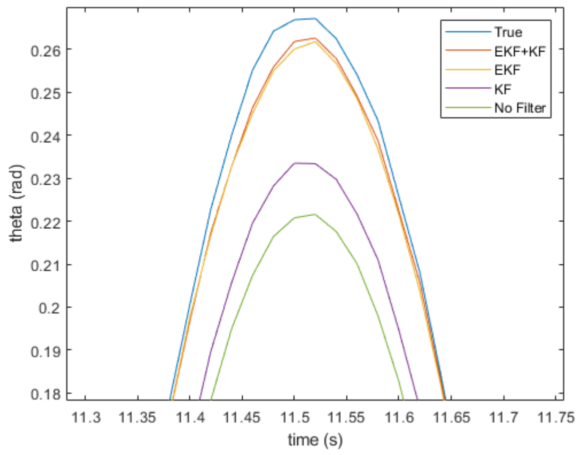

Fig. 16.   Zoomed in filter results for the robot's angular position during a 4 petal rose path over 30 seconds using all filter combinations

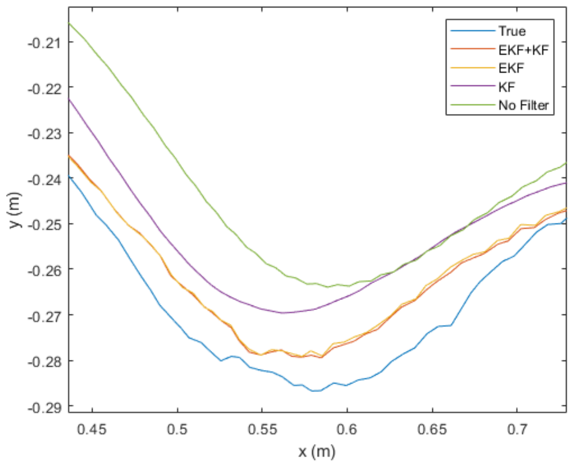

Fig. 14.   Zoomed in filter results for the robot's position during a 4 petal rose path over 30 seconds using all filter combinations

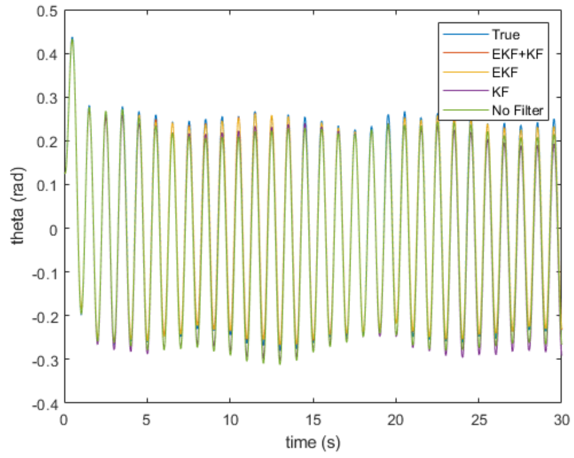

Fig. 15.   Filter results for the robot's angular position during a 4 petal rose path over 30 seconds using all filter combinations

plots. During the parametric rose path, the robot was set to follow a sinusoidal path in the θ coordinate. Since the effectiveness of the controller is not a concern, the robot is assumed to move under a no-slip condition for all simulations.

A number of physical constants must be defined to feed into the kinematic and dynamic models. These values are listed in table 6. The input and measurement noise co-variance of the motors are defined in (77) and (78). The input and measurement noise co-variance of the robot are defined in (79) and (80). In both cases the noise is assumed to be dependant only on it's respective state resulting in diagonal matrices.

$$\mathbf{Q}_{motor} = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix} \tag{77}$$

$$\mathbf{R}_{motor} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{78}$$

$$\mathbf{Q}_{robot} = \begin{bmatrix} 4e-5 & 0 & 0 \\ 0 & 4e-5 & 0 \\ 0 & 0 & 4e-5 \end{bmatrix} \tag{79}$$

$$\mathbf{R}_{robot} = \begin{bmatrix} 5e-2 & 0 & 0 \\ 0 & 5e-2 & 0 \\ 0 & 0 & 5e-2 \end{bmatrix} \tag{80}$$

Variable	Symbol	Value	Units
Wheel radius	r	3.275	cm
Robot Radius	l	19.5	cm
Motor Resistance	R	1	Ω
Motor Inductance	L	0.1	H
Motor Moment of Inertia	J	0.01	$N \cdot m^2$
Motor Viscous Friction	b	0.1	$N \cdot m \cdot s$
Motor Constant	K	0.01	$N \cdot m/A$
Time step	T	0.02	s

Table 6. List of physical constants needed to describe system kinematics and dynamics

RMSE	No Filter	KF	EKF	KF and EKF
x (m)	0.1314	0.1234	0.0071	0.0067
y (m)	0.1069	0.1263	0.0070	0.0067
θ (rad)	0.1076	0.0927	0.0078	0.0074

Table 7. State RMSEs during no filtering, kalman filtering on each motor, an EKF on the robot kinematics, and both filters combined. The test was run using the path described in (81) over 600 seconds. The test was repeated 1000 times and the RMSE values averaged.

C.1 Results

The robot was instructed to follow an arbitrary path defined in (81) over the course of 600 seconds. This test was then repeated 100 times using each combination of filters and the RMSEs of each test was averaged, the results of which are shown in table 7.

$$\xi_d = 0.5 \begin{bmatrix} \sin(t \cdot 2\pi/12) + \cos(t \cdot 2\pi/14) \\ \sin(t \cdot 2\pi/5) + \cos(t \cdot 2\pi/4) \\ \sin(t \cdot 2\pi/16) + \cos(t \cdot 2\pi/8) \end{bmatrix} \tag{81}$$

The results show there is only a slight increase in performance between using Kalman filters on the motors to feed into the robots EKF vs using the EKF by itself. When implemented on a robot, the extra filtering step should only be used if there is excess computational power that would never be needed elsewhere. Using only a Kalman filter on the wheels and no EKF on the robot kinematics showed markedly worse performance with over 10cm RMSE, although it is still better than no filtering at all. While using only the KFs is significantly worse than using an EKF, the KF requires much less computational power and could still be useful for low powered operations where positional accuracy is less important.

Results from running the robot on a 4-petal rose trajectory can be seen in figures 13 and 15. Figures 14 and 16 show zoomed versions of a small section of the previous figures in order to more clearly display the accuracy of each controller. It is obvious that the EKF and KF+EKF cases have very similar results while the KF and no filter cases lag behind, confirming the results of Table 7. It can also be noted that in all case the filters have a tendency to under estimate the position of the robot with 74% of estimates being closer to the origin than the true robot position.

It should be noted that the reason for the true path oscillating about a perfect rose plot is due to the controller and the motors physical constants. The constants for the motor were chosen to make the controller easily tune-able, however this resulted in motors with slow response times. Implementation of PID controllers on each motor would allow for more accurate values to be used, and therefore a better controller to be developed. In the future, data must be collected on the actual system to determine all parameters more accurately. Since said implementation would require significantly more computational power, it would make sense to avoid Kalman filters on each wheel to reduce computation time.

D Model identification

Using experimental data, it may be possible to identify robot parameters that can be used in a dynamic model. For this investigation, the linearized dynamic model is used. First the system is written in it's discrete time form.

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{D}_d \tag{82}$$

In order to simplify the problem, substitutions can be made to the $\mathbf{A}_d$ $\mathbf{B}_d$ and $\mathbf{D}_d$ matrices as follows:

$$\mathbf{A} = \begin{bmatrix} a_{11}^d & 0 & 0 \\ 0 & a_{22}^d & 0 \\ 0 & 0 & a_{33}^d \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & \beta_1^d & 0 & -\beta_1^d \\ \beta_2^d & 0 & -\beta_2^d & 0 \\ \beta_3^d & \beta_3^d & \beta_3^d & \beta_3^d \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} f_1^d \\ f_2^d \\ f_3^d \end{bmatrix}$$

With nine unknowns, a matrix can be designed by augmenting the $\mathbf{A}_d\mathbf{x}$, $\mathbf{B}_d\mathbf{u}$, and $\mathbf{D}_d$ matrices multiplied by a vector of the unknowns.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k & 0 & 0 & u_{2_k} - u_{4_k} & 0 & 0 & 1 & 0 & 0 \\ 0 & y_k & 0 & 0 & u_{1_k} - u_{3_k} & 0 & 0 & 1 & 0 \\ 0 & 0 & \theta_k & 0 & 0 & \sum_i u_{i_k} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a^d_{11} \\ a^d_{22} \\ a^d_{33} \\ \beta^d_1 \\ \beta^d_2 \\ \beta^d_3 \\ f^d_1 \\ f^d_2 \\ f^d_3 \end{bmatrix} \tag{83}$$

In this form as many time steps as needed can be added to the matrices. If we denote the matrices as $\mathbf{O} = \mathbf{EP}$ we quickly notice that the variable matrix can be found in the closed form as $\mathbf{P} = \mathbf{E}^{\dagger}\mathbf{O} = \left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{O}$. Of course this would assume the data is without noise or error, so instead a least squares method can be used to determine the parameters numerically.

Once the variables have been found the unknown variables from (14) can be calculated as shown in (84)-(88).

$$a_{ij} = \frac{\ln a^d_{ij}}{T} \tag{84}$$

$$\beta_1 = -\frac{\beta^d_1 M a_{11}}{e^{a_{11}T}} = \frac{\beta^d_2 M a_{22}}{e^{a_{22}T}} \tag{85}$$

$$\beta_2 = \frac{\beta^d_3 a_{33}}{\beta_1\left(e^{a_{33}T} - 1\right)} \tag{86}$$

$$J = \frac{d}{\beta_2} \tag{87}$$

$$f_i = \frac{f^d_1 a_{ij}}{e^{a_{ij}T} - 1} \tag{88}$$

D.1 Experiments

Multiple trials were run in order to collect enough data to make useful measurements. In order to make matrices sufficiently ranked, the robot must be moving in the x, y, and θ coordinate. The paths the robot traveled were circles in both the clockwise and anti-clockwise directions with a maximum PWM per wheel of 160. The tests were run for 40 seconds allowing the robot to complete four circles in each direction. To collect data for the θ coordinate the robot was spun in place in both the clockwise and anti-clockwise directions with a PWM of 150 on each wheel for 40 seconds.

The purpose of mirroring all trials with the same trial in the opposite direction was to reduce any bias caused by on direction. During testing it was noticed that if the robot was only spun in the clockwise direction, the final variables would dictate that the robot had an extreme drift in that direction.

D.2 Results

After running through the least squares method, the variables were found and are listed in table 8.

Table 8. Variables determined through least squares optimization

Symbol	Value
a^d_{11}	1.0046
a^d_{22}	1.0009
a^d_{33}	0.90876
β^d_1	$-2.5307 \cdot 10^{-5}$
β^d_2	$-1.4863 \cdot 10^{-5}$
β^d_3	$-1.9563 \cdot 10^{-4}$
f^d_1	$-2.2487 \cdot 10^{-3}$
f^d_2	$-1.9972 \cdot 10^{-4}$
f^d_3	$7.7180 \cdot 10^{-2}$

Table 9. Variables found describing the continuous state equations

Variable	Value	Units
a_{11}	0.22963	s^{-1}
a_{22}	0.04926	s^{-1}
a_{33}	-4.7835	s^{-1}
β_1	$1.9668 \cdot 10^{-5}$	$V \cdot s^{-1} \cdot m^{-1} \cdot \omega^{-1}$
β_2	318.7	m/Kg
d	0.195	m
J	0.156	$kg \cdot m^2$
f_1	-0.112	$m \cdot s$
f_2	$9.988 \cdot 10^{-4}$	$m \cdot s$
f_3	-3.723	$m \cdot s$

With the variables in discrete time found we can move on to finding the unknown variables in continuous time using (84)-(88). The results are listed in table 9. It should be noted that any variable associated with β_1 and β_2 must be multiplied by 255 in order to account for the controller using PWM values and not percentage values.

D.3 Validation

Validation of the results begins with observation of the found variables. Starting with the a^d_{ij} values, it should be expected the values are 1 since if there is no input to the system, the robot should not move. We notice this is the case for a^d_{11} and a^d_{22} which both agree with under 1% error. a^d_{33} on the other hand has a value of approximately 0.9 indicating that the robot has a tendency to spin during translation. This rotation is obvious when running the robot and is caused by not all the wheels having the same contact force with the ground.

We can also look at the discrete time friction terms. Due to a slight imbalance of the robot wheels 1 and 3 have lower contact with the ground than wheels 2 and 4 (this is also the reason for the rotational drift). Since the contact with the ground is different we expect that the frictional forces on wheels 1 and 3 to be lower than wheels 2 and 4. This is shown clearly by $\beta_1^d \approx 2\beta_2^d$ and $f_1^d \approx 10 f_2^d$ since β_1^d always corresponds to wheels 2 and 4 and during testing wheels 2 and 4 controlled the x direction during translation, which means f_1^d also corresponds to wheels 2 and 4.

Next we can compare the calculated J to an expected value. The moment of inertia calculated assuming the robot is a hollow square is 0.14, and the moment of inertia as found by the experiments is 0.156, which is approximately equal to the simplified moment of inertia.

D.4 Variable Finding Problems

There are a number of issues with the method used for variable finding. The first is the values in the $\mathbf{A}_d$ and $\mathbf{B}_d$ matrices should be constant, however this is not the case as plotting the variables as a function of velocity give a line. In order to get some reasonable values, all variables in $\mathbf{A}_d$ and $\mathbf{B}_d$ must be approximated and set as constant allowing for the friction terms to be found as a function of velocity. Doing so still results in wildly inaccurate results however, so more work must be done to more accurately measure the constant variables before solving for the friction terms.