Design and Control of a Highly Articulated Agricultural Robot

Rohan Deshpande

CMU-RI-TR-23-39

June 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
George Kantor, *chair*
Zachary Manchester
Samuel Triest

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Copyright © 2023 Rohan Deshpande. All rights reserved.

To all the robots out there toiling in thankless obscurity.

Abstract

Agricultural robots operate in environments with myriad challenges, such as nonflat terrain, nontraversable regions, strict tolerances on deviation from intended trajectories, and long travel distances. Many robots operate in agriculture today, but a large number of them are bespoke and intended for a single application. Repurposing an agricultural robot intended for one application for another typically takes significant effort, even in the case of simple changes such as performing the same task for a different crop.

This thesis presents the design of a robotic platform which is highly maneuverable and easily configurable to accommodate a wide variety of tasks with minimal reconfiguration effort. It is four-wheel steered and four-wheel driven, allowing a number of different operational modes which enable it to follow complex trajectories with tighter turns than many other agricultural robots.

These operational modes are described and evaluated, and different controllers are benchmarked on the robot over various trajectory profiles typical in agricultural settings. A pure pursuit controller is presented as a baseline geometric control algorithm, then a model predictive controller is implemented and tested on the robot. Finally, a mode-switching planner is proposed which reasons about the different operational modes available to the robot and intelligently switches between them to follow more complicated trajectories.

Acknowledgments

I'm very thankful to my advisor, Professor George Kantor, for his guidance and insight over the last couple years. He has a way of distilling the core essence of a research problem from the murky fog of confusion that so regularly permeates the practice of academic research. His pointed questions in our meetings regularly helped guide my thinking and set me on the right path when I was off going down rabbit holes.

I'd also like to thank my other committee members, Professor Zachary Manchester and Samuel Triest. I left each conversation with them a more knowledgeable person than I entered it. Professor Manchester's Advanced Robot Dynamics and Simulation, and Optimal Control and Reinforcement Learning classes were some of the best taught courses I've ever taken, and working with Sam on a project for the dynamics class was immensely valuable in aiding my research.

Special thanks go out to Abhi Silwal and Jim Picard, without whom I wouldn't have even had a robot to play with. Jim is a fount of wisdom on all things mechanical and electrical, and I've learned a great deal from him about how to build robots. If Abhi's uncanny ability to diagnose problems in the field borders on the supernatural, his capacity for getting robots working quickly is downright inhuman. He showed me, on our eleven day field test in Florida, what dedication to field robotics looks like, and I'll consider my time in the Field Robotics Center to have been worth it if I manage to hold on to half of what I learned from him.

I'd be remiss if I didn't mention that the RI community is singularly welcoming, helpful, and passionate. There truly is a genuine sense of connection among the students that transcends academic interests. Everyone cheers each other on, and is invested in everyone's success. I've had more interesting conversations, whether about robotics or entirely disjoint from anything academic, than I can count, and I consider myself fortunate to have made some lifelong friends here.

I'd be nothing without the support of my family. To my parents, grandparents, and sister who have instilled a love of learning in me and encouraged me to pursue my passions, you have my deepest gratitude.

And of course, to my fiancée Ankita who has kept me going through grad school these last couple years and given me everything to look forward to. I love you.

Funding

This work was supported by a NSF, USDA/NIFA grant under the program titled National Robotics Initiative 2.0: Ubiquitous Collaborative Robots.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Agricultural environments pose many challenges for autonomous robotic operation, such as irregular terrain, obstacles, nontraversable regions, and manipulation or perception occlusions. Overcoming these obstacles is as much an exercise in careful engineering, system, and robot design as it is sophisticated control methods and intelligent software.

In the case of the agricultural robot pictured in Figure 1.1, and the subject of this thesis, the intended application is as an herbicide sprayer which autonomously drives through rows of crops and selectively sprays herbicide only on the weeds. By selectively spraying only where needed instead of indiscriminately over entire crops, it will have the effect of using less material, lowering cost, decreasing risk of damage to crops, reducing the development of herbicide resistance, and decreasing environmental pollution. Furthermore, by doing so autonomously, it will reduce workers' exposure to aerosolized harsh, and potentially carcinogenic, chemicals. Studies have shown that precision spraying may reduce herbicide use to as little as 0.01% of blanket spraying [24].

Another common shortcoming in agricultural robotics is the prevalence of bespoke, single application robots such as harvesters built for specific crops or pruning robots built for specific phenotypes. Many robots are difficult to reconfigure for different tasks and work environments, even for relatively simple perturbations such as crop rows

Figure 1.1: The robot

planted at different widths. To combat this, we would like to design an environment-agnostic, configurable robot that can be easily be adapted and deployed to different settings and tasks. This robot must be easily controllable to a high degree of accuracy and not require too much effort to reconfigure for a new task.

1.2 Related Work

There is a rich literature of agricultural robotics research, from robots built to harvest specific crops like SWEEPER for green peppers [1] or Williams et. al.'s work on kiwifruit [27], to high throughput plant phenotyping like the Robotanist [12], to irrigation or precision spraying tasks as done by Underwood et. al. in [24].

Weed-control robots have been developed by both commercial entities such as Verdant Robotics and by research institutions and labs such as Bawden et. al.'s work in [2]. Blue River technologies also has a laser-based weeding system in the works so as to completely eliminate chemical herbicide application altogether [21]. Most commercial robotic weeding products unfortunately can cost in the millions of dollars, placing them outside the reach of smaller farms. One of the design considerations for the robot of this thesis, then, will be to ensure the cost is not too high.

There has been a recent trend away from task-specific agricultural robot design

in favor of more modular and reconfigurable approaches, but the field is still nascent. Grimstad and From have proposed the Thorvald II system which aims to be a modular robotic platform that can be assembled as desired to complete a wide variety of tasks [5]. It does, however, require disassembly and reassembly of the robot in order to reconfigure it. The MARS project by Xu and Li has similar aims, but strives for even lower cost than the Thorvald system [29]. For our application, we hope to design a robot which can be reconfigured on the fly through software rather than requiring manual assembly.

1.3 Scope and Structure

The work detailed in this thesis documents the design and operation of a wheeled mobile robot platform capable of carrying out the goals above. It will discuss the system design and construction of the robot, as well as methods to control it.

The thesis is organized into four main chapters. Chapter 2 discusses preliminaries and background for understanding the rest of the thesis. Chapter 3 discusses the design and engineering behind the robotic platform which is the focus of this thesis, and Chapter 4 details the methods by which the robot is controlled and presents experimental results. Finally, Chapter 6 summarizes the work and proposes future research directions for the robot.

The specific contributions of this thesis are to present the design of a robust, reconfigurable, highly maneuverable ground robot platform, establish the kinematics of the robot, present multiple control modes, and demonstrate trajectory following in agricultural settings with multiple controllers.

Chapter 2

Background

2.1 Conventions and Naming

Table 2.1 enumerates the symbols and physical meaning of many of the common variables used throughout this thesis. Typically, the subscript p refers to a *path* parameter and the subscript c refers to a *commanded* value. Subscripts $\{fl, fr, bl, br\}$ refer to front left, front right, back left, and back right wheels. Bolded lowercase letters such as $\mathbf{r}$ are vector quantities.

Curvature κ and lateral distance d are positive according to a right-handed coordinate system. A positive κ indicates positive angular velocity and can be intuitively thought of as the path curving left as an observer moves along it. Positive d is also when the observer is to the left of the path as they move along it.

Wheel base refers to the distance between the front and rear axles, and wheel track is the width between the centerlines of the left and right tires.

2.2 Frenet Frame

A Frenet frame is defined at all points along a path by the tangent to the path, the normal to the path at that point, and a binormal, which is the cross product of the former two. The tangent and normal are shown in Figure 2.1. This frame moves along with the robot, and can be thought of as an (s, d) space, where s is the arc

Category	Symbol/Variable	Description
State Variables and Derivatives	s, $\dot{s}$	curvilinear abscissa or arc length
	d, $\dot{d}$	lateral distance from path, and derivatives
	x, $\dot{x}$	x position/velocity of robot
	y, $\dot{y}$	y position/velocity of robot
	v, $\dot{v}$	magnitude of velocity/acceleration
	ψ, $\dot{\psi}$	heading of robot
	$\mathbf{r}^C$, $\dot{\mathbf{r}}^C$	Cartesian state vector
	$\mathbf{r}^F$, $\dot{\mathbf{r}}^F$	Frenet state vector
Control Variables	κ_c	commanded curvature
	ϕ_c	commanded steering angle
	$\dot{\theta}_c$	commanded wheel speed
	v_c	commanded velocity
	$\dot{\psi}_c$	commanded angular velocity
Trajectory Parameters	x_p	x position of point on path
	y_p	y position of point on path
	κ_p	curvature of path/trajectory
	ψ_p	heading of path/trajectory
Physical Parameters	l	wheel base
	w	wheel track
	r_w	wheel radius
	θ_b	angle of the differential bar
	θ_r	angle of the rocker joints

Table 2.1: Descriptions of commonly used symbols and variables in this thesis

length along the curve, and d is the lateral distance from the curve, as shown in Figure 2.2.

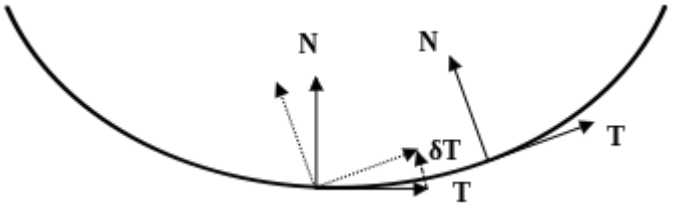

Figure 2.1: Frenet-Serret frame

A Frenet frame is useful because it allows a very natural formulation and coordinate system in applications where there is a nominal centerline or reference to follow,

such as autonomous driving or agriculture. Especially when working with row crops, there are clearly defined centerlines that the robot should track when navigating. As discussed in Section 4.2.2, deviation from the centerline can be directly optimized for in a controller.

The Frenet axes, $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{B}$, where $\mathbf{B}$ is the binormal, are defined as follows:

$$\mathbf{T} = \frac{d\mathbf{r}}{ds} \tag{2.1}$$

$$\mathbf{N} = \frac{\frac{d\mathbf{T}}{ds}}{||\frac{d\mathbf{T}}{ds}||} \tag{2.2}$$

$$\mathbf{B} = \mathbf{T} \times \mathbf{N} \tag{2.3}$$

Curvature

The quantity $||\frac{d\mathbf{T}}{ds}||$ is typically called κ, or curvature. Curvature is the reciprocal of the radius of a circle fitted to the path at that point, and is an important characteristic for certain steering methods (4.1.1, 4.1.2) or control schemes (4.2.1). Hence, we must have a method to calculate the curvature of a path.

Most often we do not have an analytical equation of a trajectory, so we must resort to sample-based methods to calculate path parameters. One such method is Menger curvature[10], which calculates the curvature based on three successive points $\mathbf{w}$, $\mathbf{u}$, $\mathbf{v}$:

$$\kappa_p = \frac{4A}{||\mathbf{u}-\mathbf{v}||||\mathbf{u}-\mathbf{w}||||\mathbf{v}-\mathbf{w}||} \tag{2.4}$$

where A is the area of the triangle formed by the three points. Substituting in the equation for the pointwise area of a triangle, we get:

$$\kappa_p = \frac{2|u_x(v_y-w_y)+v_x(w_y-u_y)+w_x(u_y-v_y)|}{||\mathbf{u}-\mathbf{v}||||\mathbf{u}-\mathbf{w}||||\mathbf{v}-\mathbf{w}||} \tag{2.5}$$

If, however, we have an analytical form available for the trajectory in terms of s, there exists an explicit formula for directly calculating the curvature from the

trajectory [7]. Let

$$\dot{x} = \frac{dx}{ds}$$
$$\dot{y} = \frac{dy}{ds}$$

Then,

$$\kappa_p = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{2.6}$$

Coordinate Transform

Localization is performed in a Cartesian frame and trajectories are specified in a Cartesian frame, but we wish to penalize the Frenet state when doing trajectory tracking, so we must be able to convert between Frenet and Cartesian coordinates. Figure 2.2 shows the relationships between important state variables. The dotted line is the reference trajectory to follow. $\mathbf{r} = (x, y)$ marks the position of the robot in Cartesian space, v indicates the robot's velocity vector, and ψ is the robot's heading. ψ_p is the path heading at the closest point on the path to the robot.

To convert from Cartesian to Frenet, we must find the closest point on the trajectory, marked $\mathbf{r_p} = (x_p, y_p)$ in Figure 2.2. Then the Frenet state is

$$s = \text{arc length}(x_p, y_p) \tag{2.7}$$
$$d = ||\mathbf{r} - \mathbf{r}_p|| \tag{2.8}$$

where d is assigned positive or negative according to the convention in Section 2.1. The arc length can be computed with numerical integration or by interpolation with an appropriate choice of underlying data structure for the path.

To transform from Frenet to Cartesian coordinates, we find the Cartesian point along the path that corresponds to s, as well as the path heading ψ_p at that point,

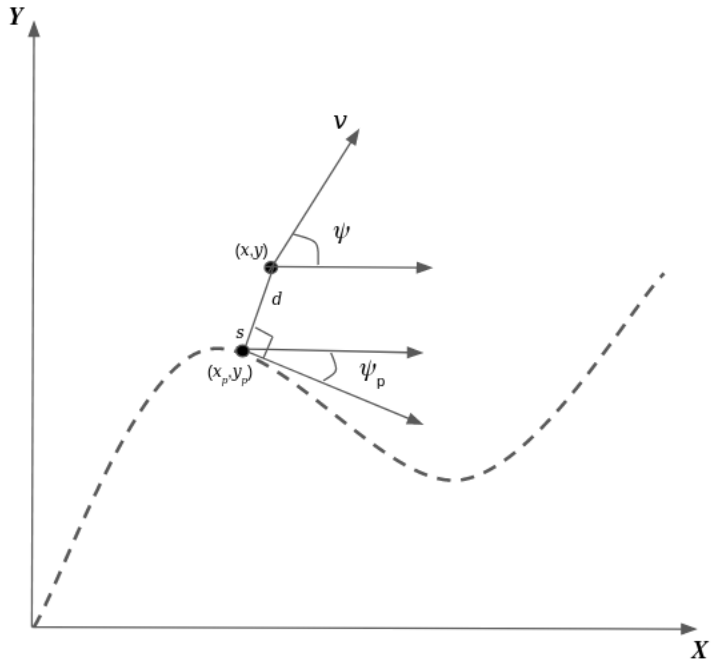

Figure 2.2: Diagram for coordinate transformations, represented as a point mass for clarity

and use the following equations:

$$x = x_p - d \sin \psi_p \tag{2.9}$$

$$y = y_p + d \cos \psi_p \tag{2.10}$$

2.3 Splines

Given a set of discrete, potentially sparse waypoints, we wish to make a smooth trajectory with an analytical representation so that we may apply Equation 2.6 to calculate curvature. In order to enforce that the curvature not have any discontinuities along the trajectory, we require that the analytical form of the trajectory be C^2-continuous [25].

Cubic spline interpolation satisfies these requirements by creating a set of piecewise cubic polynomials which pass through the waypoints and enforce continuity up to the second derivative [9]. Because they are polynomial functions, they have analytical

forms, which allow explicit calculations for path properties κ_p and ψ_p.

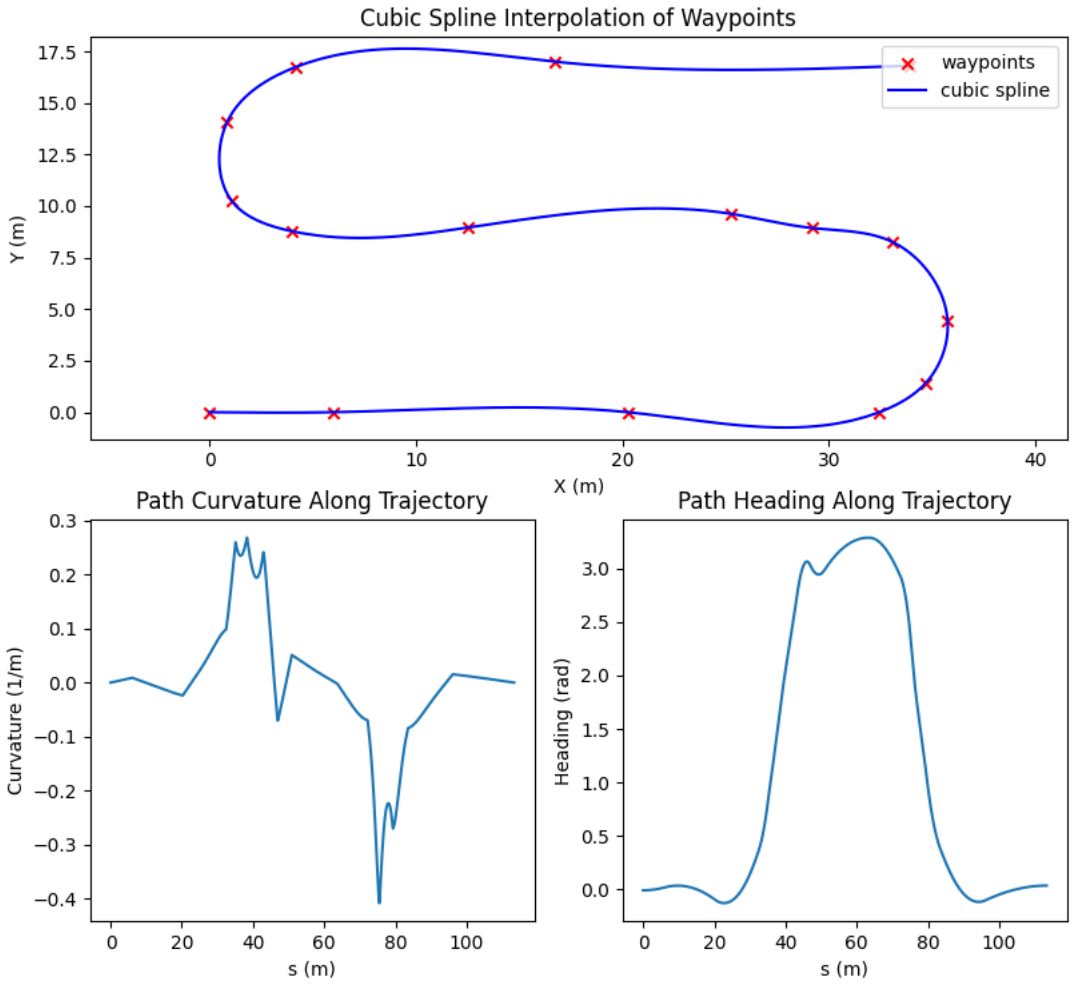

Figure 2.3: Cubic spline interpolation to generate a smooth trajectory from discrete waypoints

Figure 2.3 shows the cubic spline fit which exactly passes through the sparse waypoints and results in a continuous curvature and heading along the entire trajectory. Note that while the curvature is not smooth, it is continuous. While curvature could have been made to be smooth by choosing a higher order spline at a higher computational cost, it is unnecessary for control.

Chapter 3

System Design

3.1 Mechanical

The Buggspray robot platform is a four-wheel steered (4WS), four-wheel driven (4WD) robot with a passive rocker mechanism and the ability to change its wheel track. It is designed with a high ground clearance so that, when fitted with a spraying system, it may drive over crops without coming into contact with them and causing damage.

3.1.1 Rocker Mechanism

The purpose of the rocker mechanism is to function as a chassis averaging suspension, in which the angle of the chassis is kept at an average of the rocker angles (Figure 3.1), similar to the rocker-bogie suspension used in Mars rovers like Sojourner, Spirit, Opportunity, Curiosity, and Perseverance [28]. It also functions to more evenly distribute the weight on each wheel [8], which aids in reducing the stress on any individual drive or steering motor.

Though many of the agricultural environments this robot operates in do not feature highly irregular and bumpy terrain, the rocker mechanism is a useful feature when eventually deploying the robot in other off-road scenarios where the terrain may be much more unstructured.

Figure 3.1: The effect of the rocker mechanism in averaging the chassis angle

3.1.2 Differential Bar

The rocker joint's application as a chassis averager is achieved through the use of a differential bar which links the two rockers through a pivot. The caveat is that when the rocker is active, the wheel contact points are no longer a rectangle, so the geometric assumptions in Section 4.1 are no longer valid.

In order to be able to calculate and command the required steering angles and drive velocities to have the robot follow a certain trajectory, we must know the wheelbase and the wheel-ground contact points. We can calculate these by following the kinematic chain of transforms from the body frame to each wheel, but we must know the rocker angles. However, the robot has no way of directly measuring the rocker angles, and so we must derive a relationship from the differential bar angle, which is instrumented with an absolute position encoder, and the rocker angles.

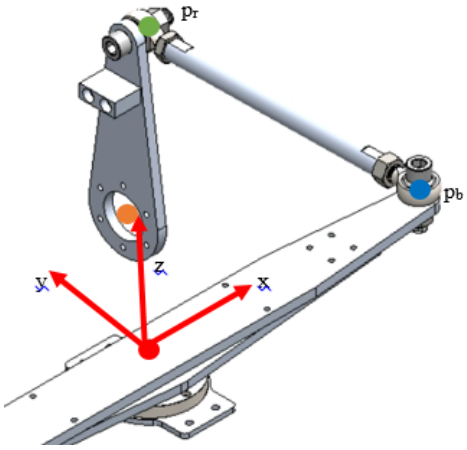

Figure 3.2: Differential Bar - Rocker Linkage

We can do this by making use of the geometric properties of the differential bar and rocker linkage as shown in Figure 3.2. By placing a frame at the differential bar pivot, we can express $\mathbf{p}_b$, the connection point on the differential bar, and $\mathbf{p}_r$, the connection point on the rocker linkage in terms of the differential bar and rocker angles, θ_b and θ_r respectively:

$$\mathbf{p}_b = \begin{bmatrix} l_b \cos\theta_b & l_b \sin\theta_b & 0 \end{bmatrix}^T \tag{3.1}$$

$$\mathbf{p}_r = \begin{bmatrix} l_b & l_r + h_r \sin\theta_r & -h_r + h_r \sin\theta_r \end{bmatrix}^T \tag{3.2}$$

where l_b is the length of the differential bar from pivot to attachment point, h_r is the height of the rocker linkage from the rocker pivot to the attachment point, and l_r is the length of the rigid rod connecting $\mathbf{p}_b$ and $\mathbf{p}_r$. By expressing the constraint $||\mathbf{p}_b - \mathbf{p}_r|| = l_r^2$ and solving for θ_r we arrive at the desired expression:

$$\begin{aligned} \theta_r = \sin^{-1}\left(\frac{l_r l_b \sin\theta_b - l_b^2(1-\cos\theta_b) - h_r^2}{h_r\sqrt{h_r^2 + (l_r - l_b\sin\theta_b)^2}}\right) \\ - \tan^{-1}\left(\frac{h_r}{l_b\sin\theta_b - l_r}\right) \end{aligned} \tag{3.3}$$

The relationship can be seen in Figure 3.3, where θ_r is plotted as a function of θ_b for two sets of parameters. The blue and orange lines are with the default link lengths as on the physical robot, which look almost linear. But the green and red lines show the

nonlinear nature of the angle relationship by redoing the calculations with different link lengths.

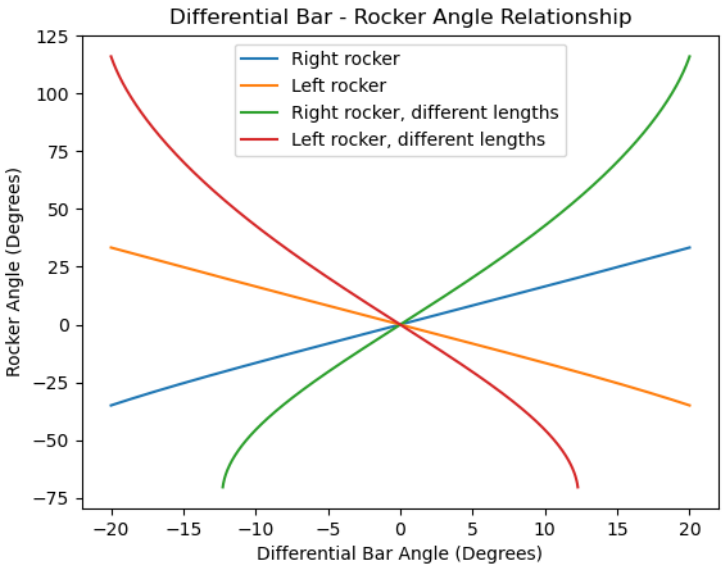

Figure 3.3: Differential Bar - Rocker Angle Relationship

3.1.3 Width Control

Different crops are planted at different widths, so the robot must be able to easily change its wheel track in order to be able to serve its function for a variety of crops. This is achieved through the use of two two-way threaded rods which extend or retract the rocker attachment points through heavy-duty linear bearings. The mechanism and its action can be seen in Figures 3.4 and 3.5.

When the wheels are on and the robot is operational, however, width control requires synchronization between the drive wheels and the threaded rods. If the widening mechanism is actuated without the wheels being oriented properly, there is a high potential for damage to the chassis due to the large off-axis torques that would be experienced at the rocker joints. This can be expressed mathematically as follows. Imagine a single wheel rolling without slip on a surface as shown in Figure 3.6. The contact point is (x, y), the steering angle is ϕ, and the angle of rotation is θ.

This can be represented in configuration space as a 4-element vector $q = \begin{bmatrix} x & y & \phi & \theta \end{bmatrix}^T$. Since the wheel is rolling without slip, the translational velocity of its center, $\dot{x}$ and $\dot{y}$

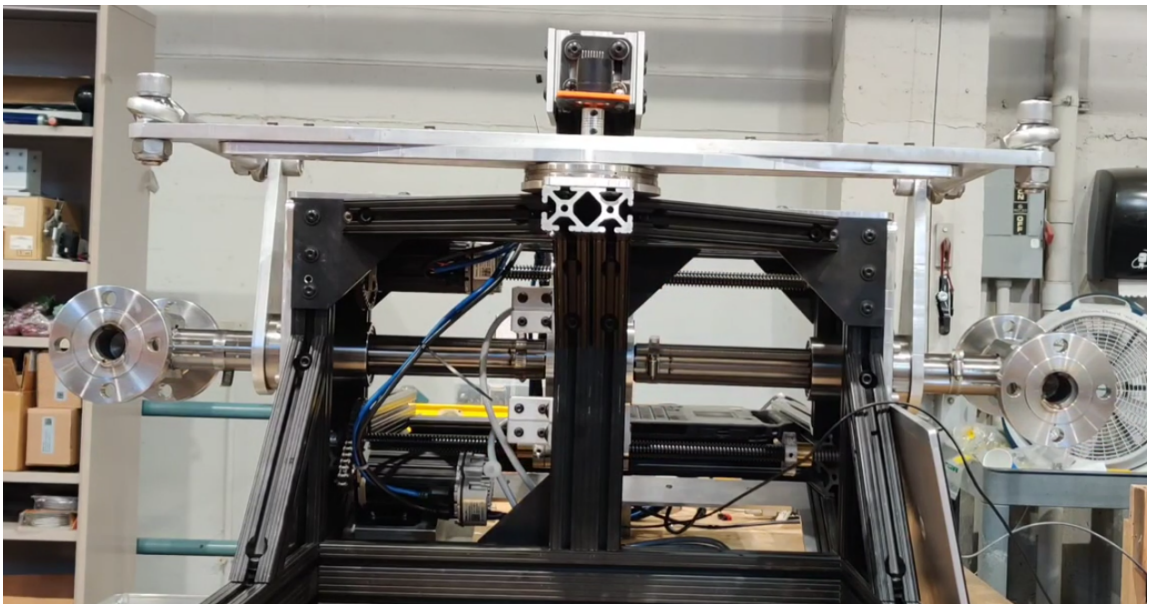

Figure 3.4: Width fully retracted

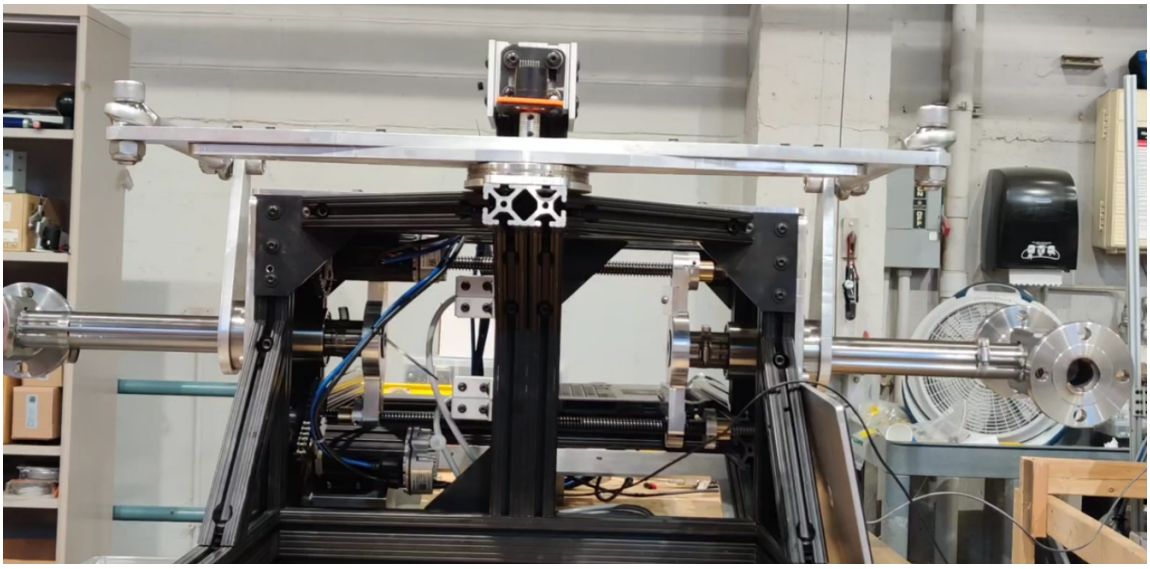

Figure 3.5: Width fully extended

is linked to its angular velocity and radius, in the direction of the steering angle. In other words,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = r\dot{\theta} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{3.4}$$

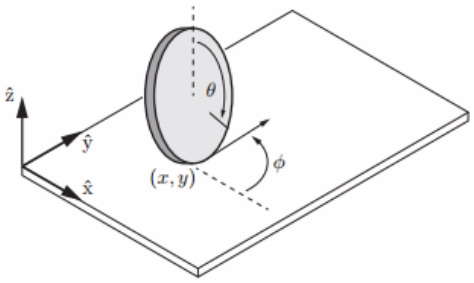

Figure 3.6: Wheel rolling without slip

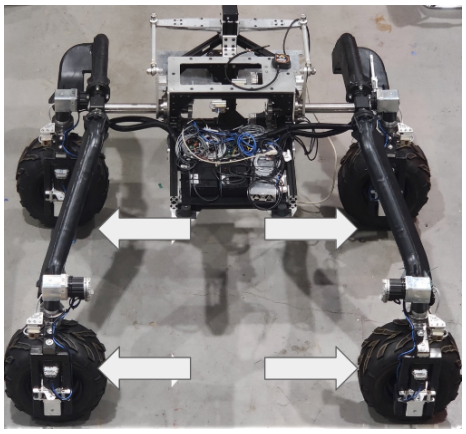

Figure 3.7: Wheel configuration for changing width

This can be rewritten as

$$\begin{bmatrix} 1 & 0 & 0 & -r\cos\phi \\ 0 & 1 & 0 & -r\sin\phi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\theta} \end{bmatrix} = 0 \tag{3.5}$$

Constraints of this form, $A(q)\dot{q} = 0$, are known as Pfaffian constraints, and depending on if they are integrable or nonintegrable, encode constraints on position/configuration or velocity, respectively [11]. In this case, we have a constraint on the velocity, since the wheel cannot slide sideways parallel to its axis of rotation.

What this means is that in order to widen or narrow the frame, each of the wheels must be oriented such that their rolling directions are parallel to the direction of changing width. This orientation is shown in Figure 3.7. We begin by setting the right side wheels to a steering angle of $\phi = -\pi/2$ and the left side wheels to a steering angle of $\phi = \pi/2$.

We must then ensure that not only the linear movement of the wheel centers be equal to the change in width, the rates must also be. If the screw mechanism which widens the axis has a linear pitch of p, a rotation of the screw by θ_s will correspond to a change in width of $\Delta w = p\theta_s$. The corresponding equation for the wheels is $\Delta w = r_w\theta_w$, where r_w is the radius of the wheels. Thus, for width control,

our control scheme is as follows. 1) Set the steering angles to $\phi_{FL} = \phi_{BL} = \pi/2$ and $\phi_{FR} = \phi_{BR} = -\pi/2$. 2) For a desired change in width calculate the screw and wheel angles and velocities with the following:

$$\Delta w = p\theta_s = r_w\theta_w \tag{3.6}$$

$$\dot{w} = p\dot{\theta}_s = r_w\dot{\theta}_w \tag{3.7}$$

3.2 Electrical

The robot is powered by a bank of eight 7.5Ah Lithium Ion batteries with a nominal voltage of 54V (Figure 3.10). This translates to an energy capacity of approximately 3200Wh. A major drawback of these batteries is their weight, at 110 pounds, which leads to increased strain on the motors. Although they add a significant amount of weight, they are capable of sustaining the robot for 10 hours of continuous field operation at a nominal 300W power draw. When deploying the robot in the field, this amount of runtime opens up the possibility of full-day operation.

We also attempted to to power the system with a much smaller, lighter 12 pound battery from the farm-ng Amiga robot platform (Figure 3.11). But at a total energy capacity of 660Wh, each battery could only support 2 hours of continuous operation before needing to be swapped out. These batteries also operate at a lower voltage of only 44V, which means the motors have a lower maximum speed, and can only produce peak torque up to a lower speed (Figures 3.8 and 3.9) [22][23].

3.3 Actuation

The robot is actuated with 10 Clearpath Teknic SCHP motors. Each wheel is steered with a CPM-SCHP-3436P-ELSA motor in position control mode and driven with a CPM-SCHP-3446P-ELSA motor in velocity control mode. One wheel module, with its associated limit switch for homing, can be seen in Figure 3.12. The steering motors are connected to the steering column through 32:1 reduction gearboxes for a peak output torque of 236.8 Nm, and the drive motors are connected to the wheels through 40:1 reduction gearboxes for a peak output torque of 392 Nm. The gearboxes are

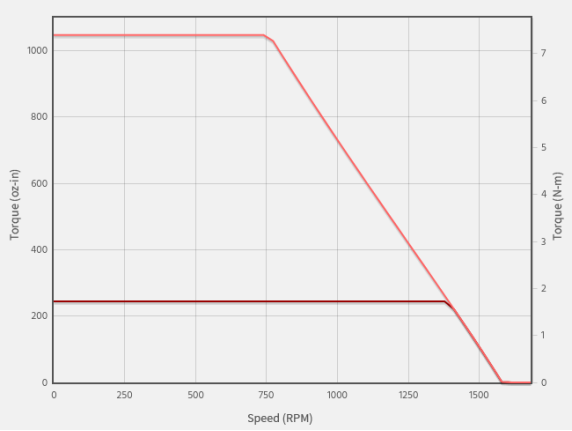

Figure 3.8: Torque-RPM Curve at 48V

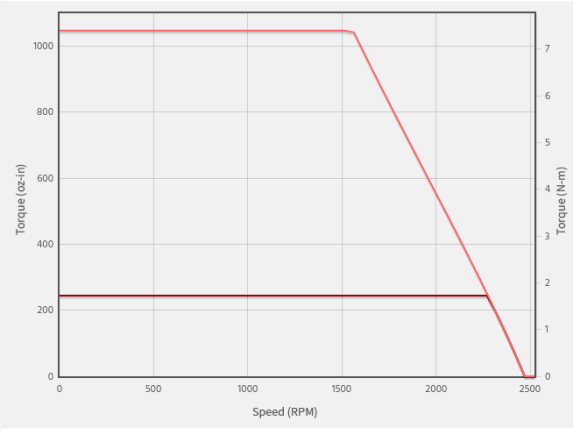

Figure 3.9: Torque-RPM Curve at 75V

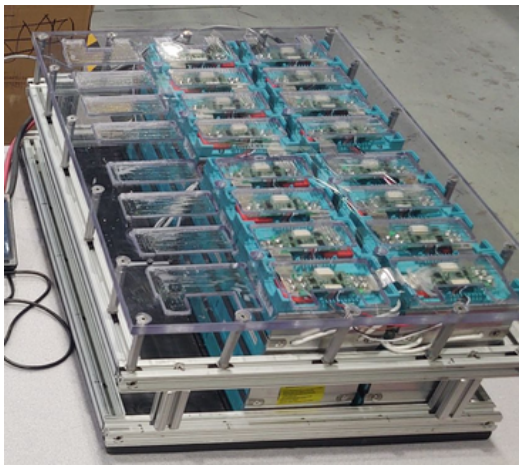

Figure 3.10: Large battery bank

Figure 3.11: Batteries from the farm-ng Amiga platform

necessary to be able to drive the robot without damaging the motors, but they come with the drawback that they introduce an appreciable amount of backlash. With the large bank of batteries from Figure 3.10 operating at 54V the maximum output wheel speed is 31.25 RPM, which translates to a maximum linear velocity of 0.83 m/s. In practice, because the torque at the maximum speed drops to near 0, the maximum speed is limited in software to the speed at which the motor can still deliver peak torque. This corresponds to a top speed of 0.6 m/s.

Two CPM-SCHP-2346P-ELSA motors are used for the width control mechanism shown in Figure 3.4 and 3.5 to drive two two-way threaded rods through a 3:1 gear

reduction.

Figure 3.12: One wheel module

3.4 Sensing

The robot used the following sensors:

1. Built-in encoders for all motors
2. Absolute position encoder for the differential bar
3. Vectornav VN-100 9-axis IMU
4. 2x Swiftnav Piksi RTK GNSS receivers and 1 Swiftnav Duro RTK GNSS Base Station in a Moving Baseline RTK setup [13]

Localization was initially performed with an EKF fusing position and orientation

data from the RTK GPS and IMU, but due to issues with hard/soft iron calibration and magnetic declination adjustment causing errors in the orientation estimates, the IMU was retired in favor of a dual RTK GPS receiver setup to estimate orientation.

3.5 Software

The software was developed in Python and C++ for ROS Noetic [14]. The software system architecture is shown in Figure 3.13.

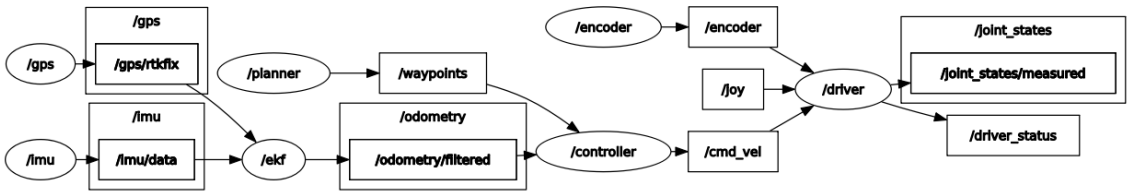

Figure 3.13: ROS system architecture

An EKF node provides a filtered odometry to the controller, which uses the localization estimate in conjunction with the waypoints from a planner to calculate a control message to send to the low level motor driver.

The motor driver was written as a multithreaded C++ application which controls each motor in its own separate thread and has a supervisory thread which handles synchronization between the other ten threads to ensures all motors move in parallel. This low level driver is responsible for taking in a twist message and turning it into the positions and velocities to command to each motor depending on which steering mode the robot is operating in. These steering modes are discussed in Section 4.1.

Chapter 4

Controls

4.1 Steering Modes

Because the robot has four independently steerable wheels, it is able to operate in a number of different steering modes, as detailed below.

4.1.1 Ackermann

Ackermann steering is the canonical approach to driving wheeled mobile robots with steerable front wheels [15]. A diagram is shown in Figure 4.1. As can be seen in this diagram, and in the maneuver performed by the robot in Figure 4.2, Ackermann steering orients the front two wheels towards a common center of rotation in line with the rear axle. This center of rotation, located a distance of r away from the midpoint of the rear axle, corresponds to an instantaneous curvature κ_c. We can calculate the required steering angles to drive the robot at this curvature as follows:

$$\phi_{fi} = \tan^{-1}\left(\frac{l}{\frac{1}{\kappa_c} - \frac{w}{2}}\right) \tag{4.1}$$

$$\phi_{fo} = \tan^{-1}\left(\frac{l}{\frac{1}{\kappa_c} + \frac{w}{2}}\right) \tag{4.2}$$

Then, since each wheel will travel a different distance as the robot moves we

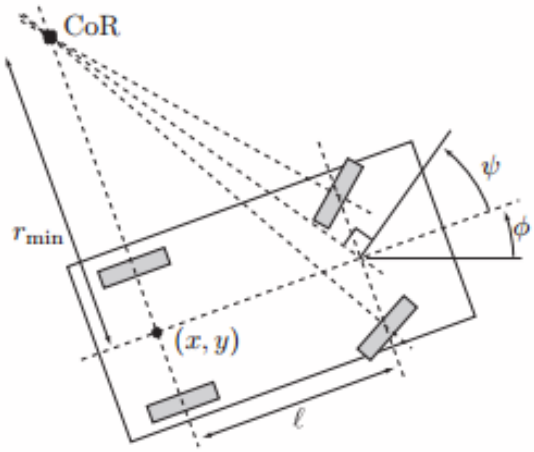

Figure 4.1: Ackermann steering diagram

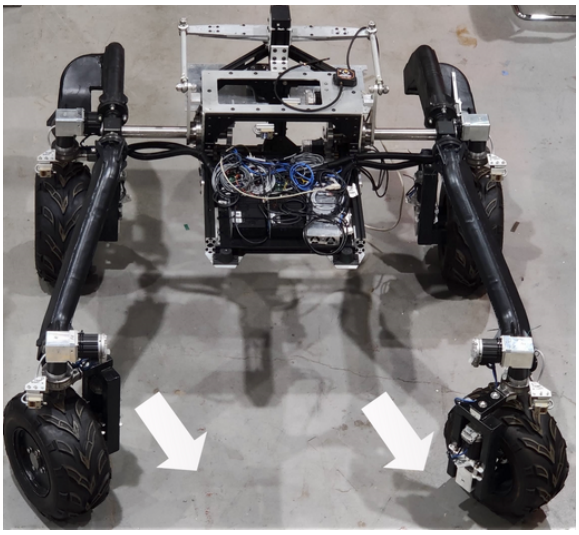

Figure 4.2: Ackermann steering on the robot

must calculate wheel speed independently for each wheel. The robot has an angular velocity about the center of rotation ω_c. Then the front inner and outer wheels must have linear velocities $v_{fi} = \omega_c/\kappa_{fi}$ and $v_{fo} = \omega_c/\kappa_{fo}$. We also know that $\omega_c = v_c\kappa_c$. Again from rotational/linear relations we get $v_{fi} = \dot{\theta}_{fi}r_w$ and $v_{fo} = \dot{\theta}_{fo}r_w$, where r_w is the wheel radius. Similar relations exist for the back wheels. Putting all of these together, we arrive at the required wheel speeds:

$$\dot{\theta}_{fi} = \frac{\kappa_c v_c}{\kappa_{fi} r_w}, \qquad \kappa_{fi} = \frac{1}{\sqrt{l^2 + (\frac{1}{\kappa_c} - \frac{w}{2})^2}} \tag{4.3}$$

$$\dot{\theta}_{fo} = \frac{\kappa_c v_c}{\kappa_{fo} r_w}, \qquad \kappa_{fi} = \frac{1}{\sqrt{l^2 + (\frac{1}{\kappa_c} + \frac{w}{2})^2}} \tag{4.4}$$

$$\dot{\theta}_{bi} = \frac{\kappa_c v_c}{\kappa_{bi} r_w}, \qquad \kappa_{bi} = 1/(1/\kappa_c - w/2) \tag{4.5}$$

$$\dot{\theta}_{bo} = \frac{\kappa_c v_c}{\kappa_{bo} r_w}, \qquad \kappa_{bo} = 1/(1/\kappa_c + w/2) \tag{4.6}$$

4.1.2 Dual Ackermann

With four wheel steering as an option, we can choose to perform dual Ackermann steering, in which all four wheels are oriented towards a common center of rotation

located on a line midway between the front and rear axles. This has the effect of simplifying the steering angle and wheel speed calculations, since by symmetry there are now only two steering angles and wheel speeds to calculate. This is because the front and back inner wheels and the front and back outer wheels will travel on the same circles as the robot drives. The required steering angles and wheel speeds for a commanded κ_c and v_c are given by Equations 4.7 - 4.10

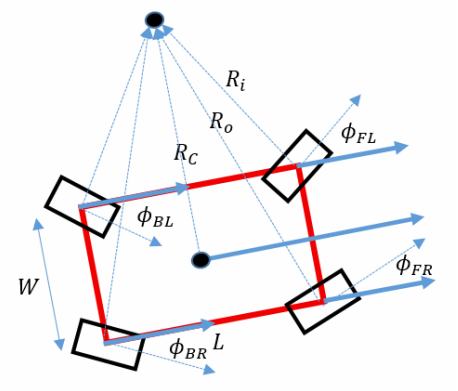

Figure 4.3: Dual Ackermann steering diagram

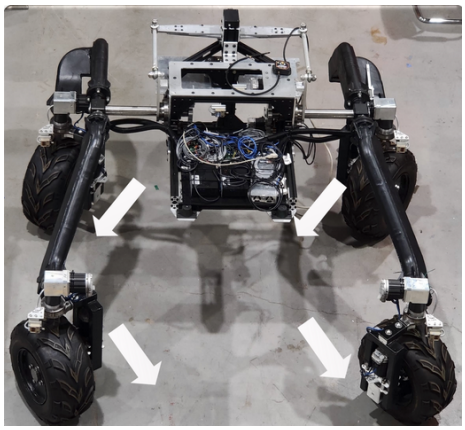

Figure 4.4: Dual Ackermann steering on the robot

$$\phi_{fi} = \phi_{bi} = \tan^{-1}\left(\frac{l}{\frac{2}{\kappa_c} - w}\right) \tag{4.7}$$

$$\phi_{fo} = \phi_{bo} = \tan^{-1}\left(\frac{l}{\frac{2}{\kappa_c} + w}\right) \tag{4.8}$$

$$\dot{\theta}_{fi} = \dot{\theta}_{bi} = \frac{\kappa_c v_c}{\kappa_i r_w}, \qquad \kappa_i = \frac{1}{\sqrt{l^2 + (\frac{1}{\kappa_c} - \frac{w}{2})^2}} \tag{4.9}$$

$$\dot{\theta}_{fo} = \dot{\theta}_{bo} = \frac{\kappa_c v_c}{\kappa_o r_w}, \qquad \kappa_i = \frac{1}{\sqrt{l^2 + (\frac{1}{\kappa_c} + \frac{w}{2})^2}} \tag{4.10}$$

A characteristic of Dual Ackermann steering is that the instantaneous velocity vector of the robot is parallel to the heading, so the side-slip angle is zero. Another

benefit is that it is capable of achieving a smaller turning radius than Ackermann steer. Figure 4.5 shows the minimum achievable radius circles on the physical robot, and Table 4.1 lists the measurements. From this data we can see that, for the joint limits and configuration of the physical robot, Dual Ackermann steering is capable of 34% tighter turns.

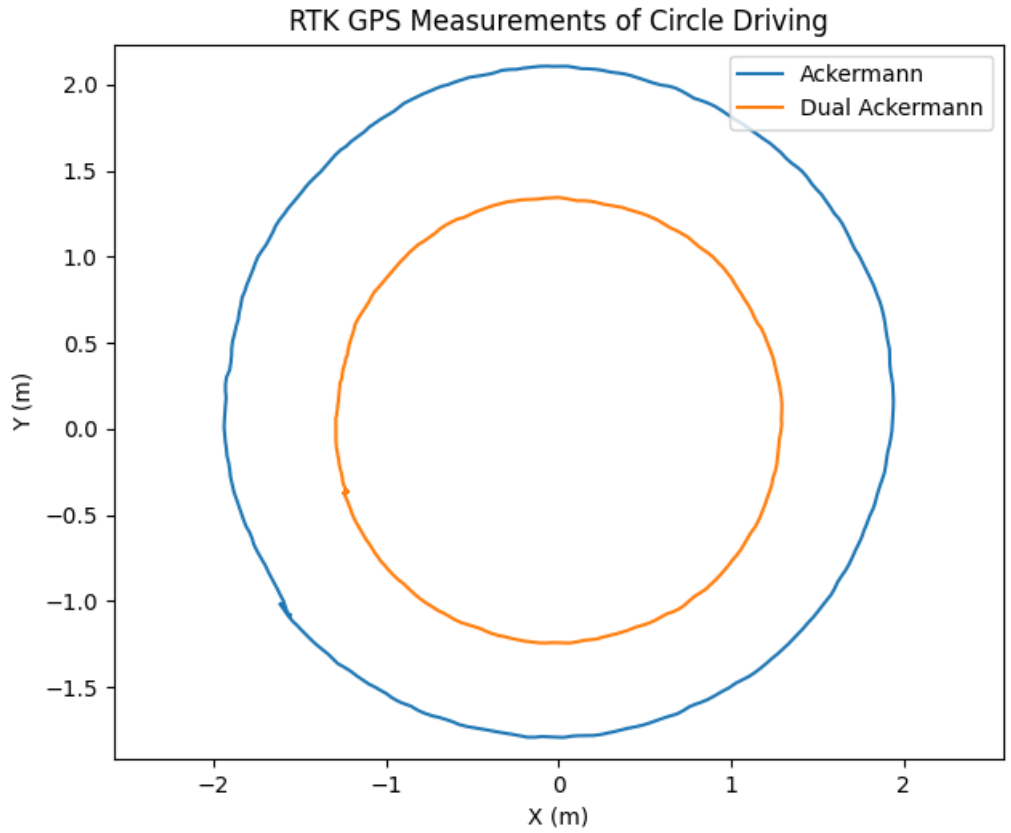

Figure 4.5: Minimum turning radius for Ackermann vs Dual Ackermann steering

Mode	Minimum Turning Radius (m)	Maximum Curvature (1/m)
Ackermann	1.94	0.515
Dual Ackermann	1.29	0.775
Crab	∞	0
Point Turn	0	∞

Table 4.1: Comparison of achievable turning radii by steering mode

For the purposes of this robot, then, there is no advantage to using Ackermann

steering over Dual Ackermann steering. Dual Ackermann can achieve tighter turns, requires fewer calculations, and has zero side-slip angle, simplifying the kinematics. Hence, both controllers discussed in Section 4.2 use Dual Ackermann steering.

4.1.3 Crab

In Crab mode, all wheels are steered and driven the same direction and amount (Figure 4.7). This has the effect of translating the robot while keeping its heading fixed. Calculating wheel steering angles and speeds for this mode is very simple, as for a commanded control ϕ_c and v_c, the steering angles are just

$$\phi_{fl} = \phi_{fr} = \phi_{bl} = \phi_{br} = \phi_c \tag{4.11}$$

and the speeds are

$$\dot{\theta}_{fl} = \dot{\theta}_{fr} = \dot{\theta}_{bl} = \dot{\theta}_{br} = \frac{v_c}{r_w} \tag{4.12}$$

The effect of this can be seen in Figure 4.6. The plots overlay the measured heading and position of the robot as it drives through a row change in Crab mode and Dual Ackermann mode. When driving in Dual Ackermann mode the heading of the robot is always tangent to the path, whereas in Crab mode it is fixed.

4.1.4 Point Turn

By orienting all wheels towards the geometric center of the robot as in Figure 4.8 and commanding all wheels to the same speed, the robot is also capable of turning in place. This has the function of changing the heading of the robot without changing its (x, y) position.

Because the wheels have steering limits and the robot is capable of changing its width, this mode is only feasible in certain configurations, namely that

$$\tan^{-1}\left(\frac{l}{w}\right) \leq \phi_{max} \tag{4.13}$$

If the robot width is too small or too large relative to its length, this condition is not met.

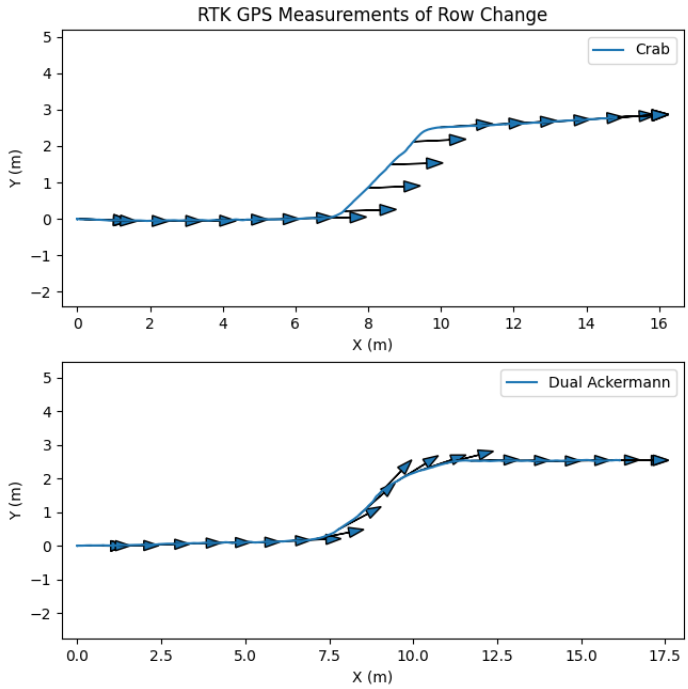

Figure 4.6: Heading comparison in a row change trajectory between Crab and Dual Ackermann steering

For a point turn, the only control input is the desired yaw rate $\dot{\psi}_c$. From this the steering angles and speeds are

$$\phi_{fl} = \phi_{br} = -\tan^{-1}\left(\frac{l}{w}\right) \tag{4.14}$$

$$\phi_{fr} = \phi_{bl} = \tan^{-1}\left(\frac{l}{w}\right) \tag{4.15}$$

and

$$\dot{\theta}_{fr} = \dot{\theta}_{br} = \frac{\dot{\psi}_c\sqrt{l^2+w^2}}{2} \tag{4.16}$$

$$\dot{\theta}_{fl} = \dot{\theta}_{bl} = -\frac{\dot{\psi}_c\sqrt{l^2+w^2}}{2} \tag{4.17}$$

where the negatives and steering/speed pairs are dictated by the joint limits of the robot.

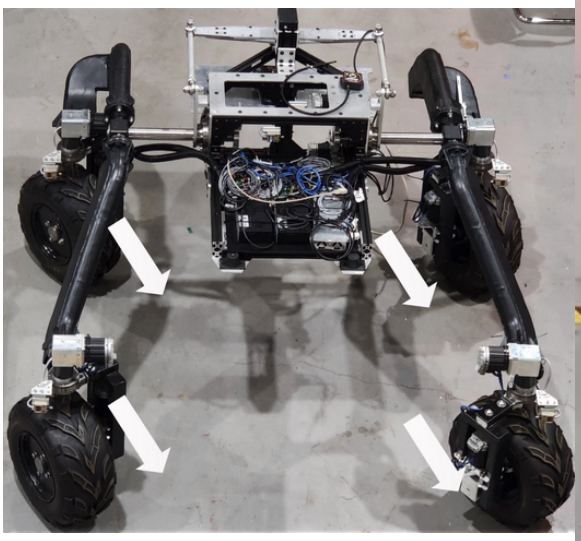

Figure 4.7: Crab steering

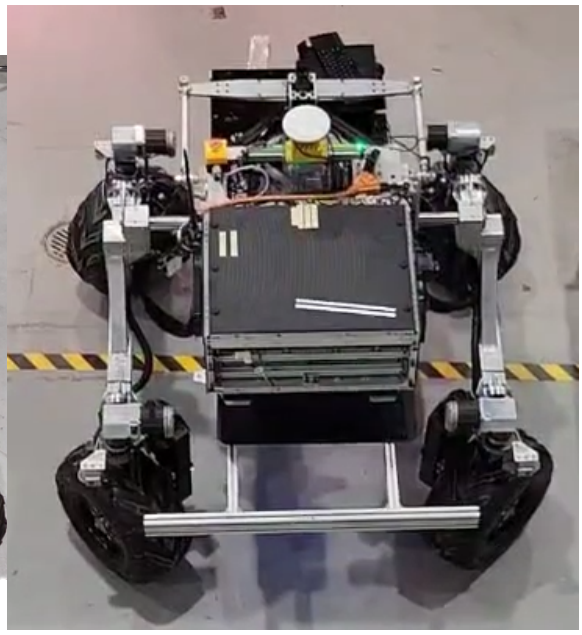

Figure 4.8: Point turn configuration

4.2 Controllers

As stated in section 4.1.2, the following controllers are formulated for the Dual Ackermann steering mode.

4.2.1 Pure Pursuit

Pure pursuit is a commonly used geometric path tracking algorithm which functions by calculating the curvature required to drive the robot to a goal point on the path some distance ahead of the current position [4]. Figure 4.9 [19] shows the geometric intuition behind the algorithm.

From the robot's position, look ahead on the path by a tunable parameter, the lookahead distance l_d. Then, by drawing an arc that connects the robot's current position to the goal point (g_x, g_y), the radius of curvature can be calculated from the heading difference α between the lookahead vector and the robot heading:

$$2R = \frac{l_d}{\sin \alpha} \tag{4.18}$$

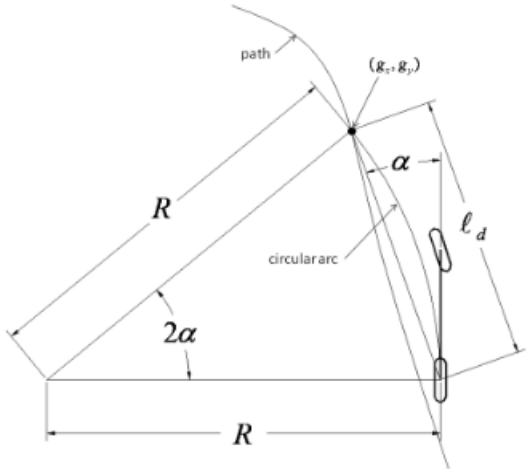

Figure 4.9: Geometric Derivation of Pure Pursuit

Since we have parameterized the Dual Ackermann controls by curvature, the pure pursuit control law can be rewritten accordingly as

$$\kappa_c = \frac{2 \sin \alpha}{l_d} \tag{4.19}$$

and the wheel angles can be calculated from Equations 4.7 and 4.8. An equivalent formulation is

$$\kappa_c = \frac{2}{l_d^2} d \tag{4.20}$$

which shows that a Pure Pursuit controller can be thought of as a simple proportional controller in Frenet coordinates.

One of the main benefits of the Pure Pursuit algorithm is its simplicity and robustness. There is only a single tunable parameter, the lookahead distance l_d, which makes this algorithm very expeditious to implement and get working on a real robot.

4.2.2 Model Predictive Control

Another controller used on the robot to perform trajectory tracking was Nonlinear Model Predictive Control, implemented online as a receding horizon formulation in a Frenet frame.

MPC is a family of control methods characterized by the use of a model to optimize

control inputs to a process with respect to an objective function in the presence of constraints [3]. In nonlinear receding horizon MPC, at each time step the future state for a horizon of length N is predicted using the model $\mathbf{r}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$, where f is the nonlinear dynamics of the system. The controls over this horizon are calculated by optimizing a cost function, which is typically a quadratic function of the error between the predicted and reference outputs. The control input is also included in the cost function so as to find a minimum cost solution. The first control input from the computed control horizon is sent to the robot, and the rest are thrown away, at which point this process repeats. Mathematically, this is

$$\begin{aligned} \min_{\mathbf{r}_{1:N}, \mathbf{u}_{1:N}} \quad & \sum_{i=1}^{N-1} ||\mathbf{r}_k^d - \mathbf{r}_k||_{Q_k} + ||\mathbf{u_k}||_{R_k} \\ \text{s.t.} \quad & \mathbf{r}_{k+1} = f(\mathbf{r}_k, \mathbf{u}_k) \\ & \mathbf{r}_k \in \mathcal{X} \\ & \mathbf{u}_k \in \mathcal{U} \end{aligned} \tag{4.21}$$

where $\mathbf{r_k^d}$, $\mathbf{r_k}$, $\mathbf{u_k}$, Q_k, and R_k are the desired state, robot state, control inputs, state cost matrix, and control cost matrix at time step k, respectively. (X and $\mathcal{U}$ are the set of feasible states and controls. $||\mathbf{r}_k^d - \mathbf{r}_k||_{Q_k}$ and $||\mathbf{u_k}||_{R_k}$ are shorthand for $\frac{1}{2}(\mathbf{r}_k^d - \mathbf{r}_k)^T Q_k (\mathbf{r}_k^d - \mathbf{r}_k)$ and $\frac{1}{2}\mathbf{u}_k^T R_k \mathbf{u}_k$. Q_k and R_k are typically chosen to be diagonal matrices $Q_k \in \mathbb{R}^{\dim(\mathbf{r})\text{x}\dim(\mathbf{r})}$, $R_k \in \mathbb{R}^{\dim(\mathbf{u})\text{x}\dim(\mathbf{u})}$, where each element of the diagonal determines the relative weighting of the corresponding state or control in the cost function.

We use ACADO [6] to solve the quadratic programming problem online. By using its code generation tool we are able to export efficient, optimized MPC controls that we can run at kHz rates on the robot, allowing real time online operation.

With some modifications from [26], we can write the system dynamics model in the Frenet Frame as follows:

Let the state in Frenet coordinates be

$$\mathbf{r}^F = \begin{bmatrix} s \\ d \\ v \\ \psi \end{bmatrix} \tag{4.22}$$

Then

$$\dot{\mathbf{r}}^F = \begin{bmatrix} \dot{s} \\ \dot{d} \\ \dot{v} \\ \dot{\psi} \end{bmatrix} = f(\mathbf{r}^F, \mathbf{u}) = \begin{bmatrix} \frac{v\cos(\psi-\psi_p)}{1-d\kappa_p} \\ v\sin(\psi-\psi_p) \\ a_c \\ v\kappa_c \end{bmatrix} \tag{4.23}$$

In this formulation, the controls are curvature and acceleration, which is integrated to give a command velocity.

$$u = \begin{bmatrix} \kappa_c \\ a_c \end{bmatrix} \tag{4.24}$$

Then the complete MPC formulation is as follows

$$\begin{aligned} \min_{\mathbf{r}_{1:N}, \mathbf{u}_{1:N}} \quad & \sum_{i=1}^{N-1} ||\mathbf{r}_k^d - \mathbf{r}_k||_{Q_k} + ||\mathbf{u_k}||_{R_k} \\ \text{s.t.} \quad & \mathbf{r}_{k+1} = f(\mathbf{r}_k, \mathbf{u}_k) \\ & d_{min} \leq d_k \leq d_{max} \\ & v_{min} \leq v_k \leq v_{max} \\ & -\kappa_{max} \leq \kappa_c \leq \kappa_{max} \\ & a_{min} \leq a_k \leq a_{max} \end{aligned} \tag{4.25}$$

with

$$Q_k = \begin{bmatrix} q_s & 0 & 0 & 0 \\ 0 & q_d & 0 & 0 \\ 0 & 0 & q_v & 0 \\ 0 & 0 & 0 & q_\psi \end{bmatrix} \tag{4.26}$$

$$R_k = \begin{bmatrix} r_\kappa & 0 \\ 0 & r_a \end{bmatrix} \tag{4.27}$$

4.3 Experiments

We test the performance of each controller on three representative trajectories. First the row change (Figure 4.10), which simulates leaving the edge of one crop patch and entering another at an offset from the previous row. Next the oval (Figure 4.11), which simulates exiting a crop row, entering another one in the same patch, exiting, and re-entering the first crop row. The third is the double row change (Figure 4.12), which simulates coverage of a field of row crops.

The performance of the controllers is evaluated with cross track error, as detailed in [20]. Cross-track error is a measure of trajectory similarity that is standard in agricultural applications because of the vital importance of perpendicular deviation from crop centerlines [16]. Parallel deviation is not as important, as this is mostly a reflection of travel speed and, unlike perpendicular deviation, poses little risk to crops.

Figures 4.10, 4.11, 4.12 show the Pure Pursuit and MPC controllers trajectory following performance on these three trajectories.

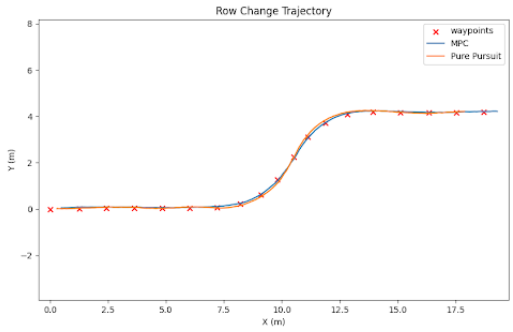

Figure 4.10: Performance of Controllers on Row Change Trajectory

From the plots we can see Pure Pursuit's characteristic to undershoot curves. Even after tuning the lookahead distance, there is a potential that a tuning which works well for one trajectory will not work well for another trajectory. Because of

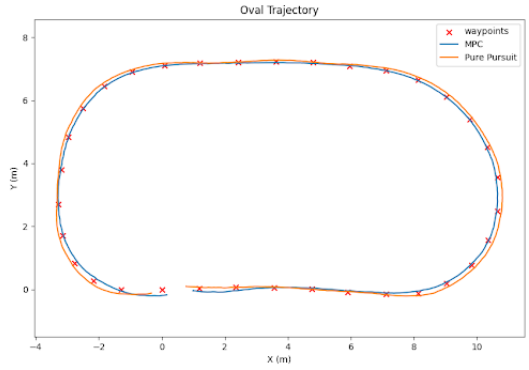

Figure 4.11: Performance of Controllers on Oval Trajectory

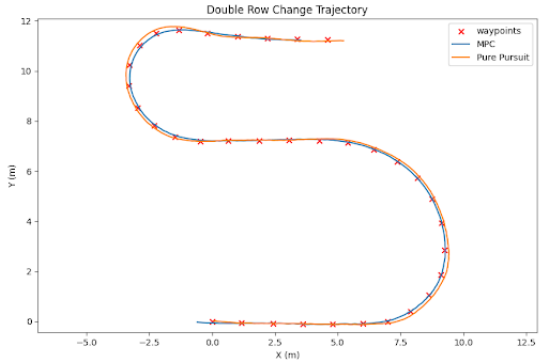

Figure 4.12: Performance of Controllers on Double Row Change Trajectory

this, a tradeoff must be made where a "good-enough" value is chosen to give good performance on all trajectories rather than excellent performance on only a few.

Both controllers perform reasonably well on the oval course, but as can be seen in Table 4.2, Pure Pursuit has almost twice the cross-track error. This may be improved with more tuning, but Pure Pursuit's main advantage is in its simplicity, not its perfect accuracy. As a geometric control method, it can track (x, y) waypoints, but it cannot reason about the robot's heading. With MPC, on the other hand, we can explicitly choose which state variables to favor. Since perpendicular deviation is much more important in agricultural applications than parallel deviation from the trajectory, the controller weights are chosen so that $q_d > q_s$.

Another benefit of MPC is that as a model-based control method, the fidelity of the model can be increased over time as better system identification or more sophisticated models are developed. It is also relatively straightforward to handle

	Pure Pursuit	MPC
Row Change	4.89	3.76
Oval	8.20	4.43
Double Row Change	7.30	3.29

Table 4.2: Average Cross-Track Error Performance (cm)

obstacles by including them as constraints in the formulation.

Chapter 5

Mode Switching Controller

5.1 Mode Switching

In Section 4.2.2 a Model Predictive Controller was developed for the Dual-Ackermann steering mode (4.1.2). This is sufficient for trajectories where there are no sections of higher curvature than the robot's maximum achievable curvature (Table 4.1), or when the orientation of the robot along the trajectory does not matter. If, however, we do care about the orientation of the robot at a particular location we can take advantage of the multiple steering modes. The different steering modes can be parameterized with different controls, and each has different kinematic characteristics. Table 5.1 shows the parameterization of each mode, and Table 5.2 shows the kinematics of each mode that can be used as MPC's forward model. As before, we will consider Dual Ackermann to supersede Ackermann, so the controls will only be presented for the former.

Mode	Control Parameterization	Description
Dual Ackermann	κ_c, v_c	Curvature and Velocity
Crab	ϕ_c, v_c	Steering Angle and Velocity
Point Turn	$\dot{\psi}_c$	Yaw Rate

Table 5.1: Controls per steering mode

Each column in Table 5.2 can be considered as a separate forward model f, so that we can formulate a mode switching controller as follows:

	Dual Ackermann	Crab	Point Turn
$\dot{s}$	$\frac{v\cos(\psi-\psi_p)}{1-d\kappa_p}$	$\frac{v\cos(\psi+\phi_c-\psi_p)}{1-d\kappa_p}$	0
$\dot{d}$	$v\sin(\psi-\psi_p)$	$v\sin(\psi+\phi_c-\psi_p)$	0
$\dot{v}$	a_c	a_c	0
$\dot{\psi}$	$v\kappa_c$	0	$\dot{\psi}_c$

Table 5.2: Kinematics per steering mode

$$\begin{aligned} \underset{f}{\text{argmin}} \min_{\mathbf{r}_{1:N},\mathbf{u}_{1:N}} \quad & \sum_{i=1}^{N-1} ||\mathbf{r}_k^d - \mathbf{r}_k||_{Q_k,f} + ||\mathbf{u_k}||_{R_k,f} \\ \text{s.t.} \quad & \mathbf{r}_{k+1} = f(\mathbf{r}_k, \mathbf{u}_k) \\ & \mathbf{r}_k \in \mathcal{X} \\ & \mathbf{u}_k \in \mathcal{U}_f \end{aligned} \tag{5.1}$$

The key difference between this and the original MPC formulation in Section 4.2.2 is the argmin wrapping the entire optimization problem, so we are now optimizing over different forward models f as well. Each model is parameterized by different controls as described in Table 5.1, so the control constraints are now $\mathbf{u}_k \in \mathcal{U}_f$, not just $\mathbf{u}_k \in \mathcal{U}$. Each cost matrix is also mode-dependent, as different modes can achieve different types of motion.

Performing this argmin is still computationally tractable because each individual model can be computed at kHz rates.

5.2 Experiments

We show the performance of the controller in simulation on a trajectory where the "best" sequence of states is to intersperse two Point Turns of 90 degrees each between straight driving segments, simulating the robot exiting one row and attempting to enter the next row over.

Figure 5.1 shows the performance of the controller operating solely in Dual Ackermann mode. Significant devations from the intended trajectory can be seen in red and yellow, because the robot is physically unable to steer at the required curvature. Figure 5.2 shows the performance with the mode-switching controller

enabled, where the controller understands that switching to Point Turn mode for the sharp turns enables better tracking.

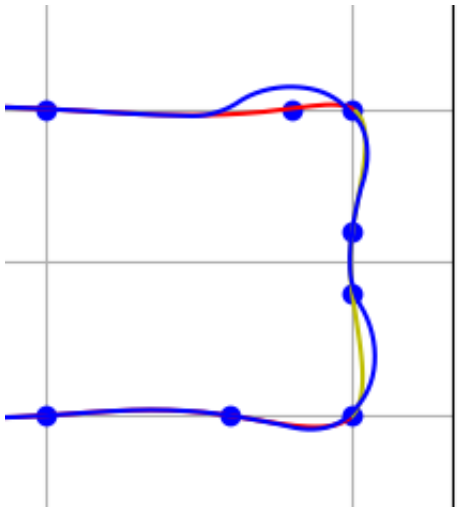

Figure 5.1: Performance of Dual Ackermann MPC, with no mode switching. Blue is the robot trajectory. Red and yellow are the intended trajectory

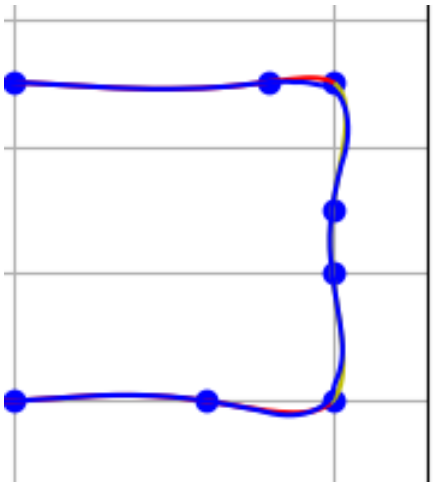

Figure 5.2: Performance of mode switching controller. Blue is the robot trajectory. Red and yellow are the intended trajectory

5.3 Limitations

While the mode switching controller does enable more complex trajectories and better tracking in some scenarios, it suffers from some drawbacks.

First, it requires careful tuning to ensure the costs of each objective function are similar in magnitude to one another, and this complexity is significantly greater than that of tuning a single mode. Second, each rollout considered in the encompassing argmin is limited to a single mode. This means the controller is not able to directly compare multi-mode horizons, such as one which operates in Dual Ackermann mode for some number of time steps, then switches to and operates in Point Turn for some other number of time steps. It is only able to compare a horizon of purely Dual Ackermann versus a horizon of purely Crab Mode versus a horizon of purely Point Turn.

Chapter 6

Conclusion

6.1 Conclusion

This thesis has presented the design and operation of a field robot nominally intended for agricultural applications, but broadly applicable as a general ground robot platform where increased maneuverability is an operational necessity. Despite the joint limits on the four-wheel steering configuration, the robot is capable of diverse steering behaviors which enable multiple driving modes such as Ackermann, Dual Ackermann, Crab Steer, and Point Turn. Multiple controllers were also developed and implemented on the robot to demonstrate its ability to accurately follow trajectories representative of agricultural applications.

By developing a controller which can intelligently reason about how to deploy different steering modes we equip the robot with the ability to closely track more complex or challenging reference trajectories. This ability is important in certain agricultural applications such as surveying or herbicide spraying, in which the robot carries a payload whose orientation needs to be controlled at various points along the trajectory.

6.2 Limitations and Future Work

The modelling and control accomplished until now has been assuming flat ground. Although this is a reasonable assumption in many agricultural settings, if the robot is truly to be used in other off-road applications, the activation of the differential mechanism described in Section 3.1.2 should be included in the control formulation. Seegmiller and Kelly have presented enhanced 3D kinematic modeling in [17] and [18] that would form the basis of this.

Additionally, a kinematic vehicle model is used in the MPC formulation. Although this is a reasonable assumption at the low operational speeds that are the domain of this robot, the kinematic model does not account for wheel slip, which is a common occurrence when working on crops in looser soils.

As detailed in section 3.3, the gearboxes on the steering and drive motors introduce a not insignificant amount of backlash into the drivetrain. In order to achieve better trajectory tracking performance, we can leverage existing controller formulations which explicitly reason about actuator backlash, as in [30].

The on-robot results presented for each controllers only test planar trajectory following. In the future, (x, y, ψ) trajectories should also be benchmarked, as this would more clearly demarcate the performance differences between Pure Pursuit and MPC.

Appendix A

Herbicide Spraying

The robot was also taken for a 10-day field test and systems integration at the Southwest Florida Research and Education Center in Imokalee, Florida. As mentioned in Section 1.3, the grant for this project was a joint effort between the Carnegie Mellon University and University of Florida. This thesis documents the work done at Carnegie Mellon in developing the robot platform to be deployed, and this appendix briefly discusses the spraying system and integration between the two.

Figure A.1: Spraying unit mounted on robot

The spraying system consists of 22 independently operated nozzles arranged in a linear array spanning the entire width of a crop row (Figure A.1). A camera is mounted a known distance ahead of the nozzles and is used to detect and classify weeds. Once the weeds are identified, the corresponding nozzles are turned on so as to perform a targeted spray.

Figure A.2 shows the robot driving through the plasticulture beds at the testing site.

Figure A.2: Robot driving through the field and spraying

Bibliography

[1] Boaz Arad, Jos Balendonck, Ruud Barth, Ohad Ben-Shahar, Yael Edan, Thomas Hellström, Jochen Hemming, Polina Kurtser, Ola Ringdahl, Toon Tielen, and Bart van Tuijl. Development of a sweet pepper harvesting robot. *Journal of Field Robotics*, 37(6), September 2020. ISSN 1556-4959. doi: 10.1002/rob.21937. 1.2

[2] Owen Bawden, Jason Kulk, Ray Russell, Chris McCool, Andrew English, Feras Dayoub, Chris Lehnert, and Tristan Perez. Robot for weed species plant-specific management. *Journal of Field Robotics*, 34(6):1179–1199, 2017. doi: https://doi.org/10.1002/rob.21727. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21727. 1.2

[3] E.F. Camacho and C.B. Alba. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2013. ISBN 9780857293985. 4.2.2

[4] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA, January 1992. 4.2.1

[5] Lars Grimstad and Pål Johan From. The thorvald ii agricultural robotic system. *Robotics*, 6(4), 2017. ISSN 2218-6581. doi: 10.3390/robotics6040024. URL https://www.mdpi.com/2218-6581/6/4/24. 1.2

[6] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. 4.2.2

[7] Reza N. Jazar. *Steering Dynamics*, pages 387–495. Springer New York, New York, NY, 2014. ISBN 978-1-4614-8544-5. doi: 10.1007/978-1-4614-8544-5_7. URL https://doi.org/10.1007/978-1-4614-8544-5_7. 2.2

[8] JPL. The perseverance rover wheels, 2020. URL https://mars.nasa.gov/mars2020/spacecraft/rover/wheels/. 3.1.1

[9] Q. Kong, T. Siauw, and A. Bayen. *Python Programming and Numerical Methods: A Guide for Engineers and Scientists*. Elsevier Science, 2020.

ISBN 9780128195499. URL https://pythonnumericalmethods.berkeley.edu/notebooks/chapter17.03-Cubic-Spline-Interpolation.html. 2.3

[10] F Leymarie. Notes on menger curvature, 2003. URL https://web.archive.org/web/20070821103738/http://www.lems.brown.edu/vision/people/leymarie/Notes/CurvSurf/MengerCurv/index.html. 2.2

[11] Kevin M Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge Univeristy Press, 2017. ISBN 978-1107156302. 3.1.3

[12] Tim Mueller-Sim, Merritt Jenkins, Justin Abel, and George Kantor. The robotanist: A ground-based agricultural robot for high-throughput crop phenotyping. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3639, 2017. doi: 10.1109/ICRA.2017.7989418. 1.2

[13] Swift Navigation. Moving baseline rtk configuration, 2022. URL https://support.swiftnav.com/support/solutions/articles/44001907899-moving-baseline-rtk-configuration. 4

[14] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, may 2009. 3.5

[15] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011. ISBN 9781461414339. 4.1.1

[16] Joseph D. Rounsaville, Joseph S. Dvorak, and Timothy S Stombaugh. Methods for calculating relative cross-track error for asabe/iso standard 12188-2 from discrete measurements. *Transactions of the ASABE*, 59(6):1609–1616, 2016. doi: 10.13031/trans.59.11902. URL https://doi.org/10.13031/trans.59.11902. 4.3

[17] Neal Seegmiller and Alonzo Kelly. Enhanced 3d kinematic modeling of wheeled mobile robots. In *Proceedings of Robotics: Science and Systems (RSS '14)*, July 2014. 6.2

[18] Neal Seegmiller and David Wettergreen. Control of a passively steered rover using 3-d kinematics. pages 607–612, 09 2011. doi: 10.1109/IROS.2011.6048617. 6.2

[19] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. Technical Report CMU-RI-TR-09-08, Carnegie Mellon University, Pittsburgh, PA, February 2009. 4.2.1

[20] Yaguang Tao, Alan Both, Rodrigo I. Silveira, Kevin Buchin, Stef Sijben, Ross S. Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. A comparative analysis of trajectory similarity measures. *GIScience & Remote*

Sensing, 58(5):643–669, 2021. doi: 10.1080/15481603.2021.1908927. URL https://doi.org/10.1080/15481603.2021.1908927. 4.3

[21] Blue River Technologies. Our products, 2023. URL https://bluerivertechnology.com/our-products/. 1.2

[22] Teknic. Cpm-schp-3446p-elnb, 2023. URL https://teknic.com/model-info/CPM-SCHP-3446P-ELNB/?model_voltage=48VDC. 3.2

[23] Teknic. Cpm-schp-3446p-elnb, 2023. URL https://teknic.com/model-info/CPM-SCHP-3446P-ELNB/?model_voltage=75VDC. 3.2

[24] James Patrick Underwood, Mark Calleija, Zachary Taylor, Calvin Hung, Juan I. Nieto, Robert C. Fitch, and Salah Sukkarieh. Real-time target detection and steerable spray for vegetable crops. 2015. 1.1, 1.2

[25] Michiel van de Panne. Parametric curves, 1996. URL https://www.cs.helsinki.fi/group/goa/mallinnus/curves/curves.html. 2.3

[26] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993, 2010. doi: 10.1109/ROBOT.2010.5509799. 4.2.2

[27] Henry A.M. Williams, Mark H. Jones, Mahla Nejati, Matthew J. Seabright, Jamie Bell, Nicky D. Penhall, Josh J. Barnett, Mike D. Duke, Alistair J. Scarfe, Ho Seok Ahn, JongYoon Lim, and Bruce A. MacDonald. Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms. *Biosystems Engineering*, 181:140–156, 2019. ISSN 1537-5110. doi: https://doi.org/10.1016/j.biosystemseng.2019.03.007. URL https://www.sciencedirect.com/science/article/pii/S153751101830638X. 1.2

[28] Matthew Woodall, Gunnar Baechler, Iman Salehinia, and Nicholas Pohlman. Mars rover rocker-bogie comparative differential study. 9:1874–1888, 04 2020. 3.1.1

[29] Rui Xu and Changying Li. A modular agricultural robotic system (mars) for precision farming: Concept and implementation. *Journal of Field Robotics*, 39(4):387–409, 2022. doi: https://doi.org/10.1002/rob.22056. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22056. 1.2

[30] Haslinda Zabiri and Yudi Samyudia. A self detection and compensation of actuator backlash in the framework of constrained mpc design. volume 2, pages 1186 – 1194 Vol.2, 08 2004. ISBN 0-7803-8873-9. doi: 10.1109/ASCC.2004.185025. 6.2