Investigation of MPC for Ground Robots on Imperfect Terrains

Manoj Kumar Reddy Manchala*, Odysseus Hadjioannou†
Department of Mechanical and Aerospace Engineering, University of California San Diego
Email: *mmanchala@ucsd.edu, †ohadjioa@ucsd.edu

***Abstract*—Autonomous navigation in agricultural settings presents unique challenges due to unpredictable terrain variations and the demand for precision to avoid crop damage. This paper proposes an MPC-based navigation system for tractors that explicitly adapts to irregular terrain conditions. The MPC formulation incorporates a detailed tractor model that reacts to terrain-induced disturbances. The effectiveness of the system is demonstrated through simulation.**

***Index Terms*—Robust MPC, Terrain adaptation, Agricultural Robotics**

I. Introduction

This project explores the development of a robust Model Predictive Control (MPC) system for an autonomous ground vehicle designed to navigate complex agricultural fields, specifically between tree lanes while avoiding collisions.

Current autonomous navigation techniques often rely on simplified or idealized representations of terrain. However, real-world agricultural settings present a significant challenge due to their irregular surfaces, posing a risk for traditional control algorithms. MPC presents a compelling solution due to its ability to handle constraints and optimize control decisions over a receding horizon. Robust MPC specifically enhances the controller's resilience by addressing uncertainties and disturbances inherent in agricultural environments.

This paper focuses on the following key contributions:

Robust MPC Formulation: Development of a robust MPC controller expressly designed for Autonomous Ground Vehicle navigation within the constraints of tree lanes and the challenges of irregular agricultural terrain.
Uncertainty Modeling: Characterization of uncertainties and disturbances typical of agricultural settings (e.g., uneven ground, soil variation), and their incorporation into the MPC model.
Performance Evaluation: Thorough evaluation of the proposed robust MPC system through simulation. Comparison with baseline navigation approaches will highlight the advantages of the robust MPC solution in the context of agricultural navigation.

II. Problem Statement

To simplify the initial controller design, the tractor's dynamics are approximated using a Dubin's car model. The core challenge lies in modeling the effects of agricultural terrain as disturbances to the steering angle of the Dubin's car. The objective is to develop a robust Model Predictive Control (MPC) system that:

Compensates for disturbances: Mitigates the impact of terrain-induced steering disturbances, ensuring the tractor follows a desired path within the constraints of tree lanes.

Adapts to uncertainties: Accounts for the inherent variability in terrain characteristics and potential modeling errors in the disturbance representation.

Assumptions: It is assumed that the environment has no moving obstacles and the only disturbance present is the disturbance induced in the steering due to the imperfections in the terrain.

Desired Results: Design a robust Model Predictive Control (MPC) system tailored for an autonomous ground vehicle represented by a Dubin's car model, operating under terrain-induced disturbances. Demonstrate, via simulation the controller's ability to navigate safely and efficiently between tree lanes despite the impact of uneven terrain

III. Technical Approach

A modified Dubin's Car model is created to introduce the effects of irregular terrain. The model emulates varying friction and varying slope of the terrain.

Dubin's Car Model - Disturbance induced

The Dubin's car model is a simplified representation of a car-like vehicle with the following key characteristics and constraints:

Forward motion only: The vehicle cannot move in reverse.

Minimum turning radius: The vehicle's turning capability is limited, and it must follow paths with a curvature above a certain threshold. This translates into a minimum turning radius, denoted by R.

Nonholonomic constraint: The vehicle cannot move sideways, only in the direction of its heading. This is reflected in the absence of direct control over $\dot{x}$ and $\dot{y}$.

Kinematic Equations

A typical representation of the Dubin's car model uses the following kinematic equations:

$$\begin{aligned}\dot{x} &= v\cos(\theta)\\ \dot{y} &= v\sin(\theta)\\ \dot{\theta} &= \frac{v}{R}u + \delta(t)\end{aligned}$$

Where:

x and y represent the position of the vehicle in a 2D plane. θ represents the vehicle's heading angle. v is the forward velocity (assumed constant in the basic model). R is the minimum turning radius. u is the control input (steering), typically bounded between -1 and 1. $\delta(t)$ is the time varying disturbance.

Estimating $\delta(t)$: Sensor data (IMU or a terrain mapping sensor) can be utilized to estimate the distrubance in real time. But for simplicity, this disturbance will be represented with a stochastic process potentially informed by the field measurements

A. Robot Dynamics

Consider a robot navigating a terrain characterized by spatial variations in friction and elevation. The state of the robot is defined by its position (x, y), yaw angle θ, and elevation z. The elevation is also considered as a state to keep track of the effects it has on the other states. The robot's motion is governed by its linear velocity v and angular velocity ω, subject to the terrain's properties.

Terrain Interaction: The terrain's influence is represented through friction and elevation maps, affecting the robot's motion as follows:

- Friction, $f(x, y)$, affecting the robot's translational motion.
- Elevation, $h(x, y)$, influencing the robot's potential energy and inducing yaw disturbance.

Dynamics Equations: The dynamics of the robot can be modeled by considering the interaction with the terrain:

$$v_{\text{drive}} = v \cdot f(x, y) \tag{1}$$

This equation represents the driving force, modified by the terrain friction.

$$\Delta h = h(x, y) - z \tag{2}$$

Δh is the elevation change affecting the robot's vertical motion.

Motion Equations: The robot's linear and angular motions are influenced by the terrain as follows:

$$\Delta x = v_{\text{drive}} \cdot \cos(\theta) \cdot \Delta t \cdot (1 - \alpha \cdot |\Delta h|) \tag{3}$$

$$\Delta y = v_{\text{drive}} \cdot \sin(\theta) \cdot \Delta t \cdot (1 - \alpha \cdot |\Delta h|) \tag{4}$$

where Δx and Δy are the changes in the robot's position, Δt is the time step, and α is a coefficient representing the influence of elevation change on motion efficiency.

$$yaw\_disturbance = \beta \cdot \Delta h \cdot \text{sign}(v) \tag{5}$$

$$\Delta\theta = \omega \cdot \Delta t + yaw\_disturbance \tag{6}$$

$$\Delta\theta = \omega \cdot \Delta t + \beta \cdot \Delta h \cdot \text{sign}(v) \tag{7}$$

$\Delta\theta$ represents the change in yaw angle, with β quantifying the yaw disturbance due to elevation changes.

State Update: The robot's state is updated as:

$$x_{\text{next}} = x + \Delta x \tag{8}$$

$$y_{\text{next}} = y + \Delta y \tag{9}$$

$$\theta_{\text{next}} = \theta + \Delta\theta \tag{10}$$

$$z_{\text{next}} = h(x + \Delta x, y + \Delta y) \tag{11}$$

These equations collectively describe the robot's dynamics as influenced by the terrain, accounting for both the translational and rotational movements.

B. Terrain Model

The terrain model is constructed to simulate the environmental conditions a robot might encounter, featuring variations in elevation and friction. The model incorporates both structured and random elements to mimic real-world terrain features like furrows and ruts. The terrain height is a linear combination of patterns. Low frequency 2D sinusoidal patterns represent large terrain features like the side of a hill. A high frequency 1D sinusoid represents plough furrows. The conditions underneath the car were obtained by linearly interpolating the height and friction between the grid nodes.

Field Dimensions and Grid Resolution: The terrain is defined over a rectangular field of dimensions 20×15 meters. The field is discretized into a grid using the following resolution:

- x dimension: 40 points, resulting in a grid spacing along the x-axis.
- y dimension: 30 points, resulting in a grid spacing along the y-axis.

Base Elevation: The base elevation of the terrain features a smooth, undulating surface created by a sinusoidal function, which varies with both x and y coordinates:

$$base\_elevation = 0.2 \sin\left(\frac{x}{5}\right) \cos\left(\frac{y}{4}\right)$$

This component models gentle terrain undulations.

Furrows: To simulate furrows typical in agricultural fields, periodic dips are added along the x-axis:

$$furrow\_pattern = 0.4 \sin\left(2\pi \frac{x}{1.5}\right)$$

These furrows, with a periodicity of 1.5 meters, represent structured troughs in the terrain.

Random Ruts: Random ruts are introduced to model the irregularities found in real terrain, particularly along paths frequently traveled by equipment. These are concentrated along certain paths and are deeper than the base variation:

$$rut\_paths = \left|\sin\left(2\pi \frac{y}{5}\right)\right| < 0.1$$
$$ruts = 0.5 \times randn(size(x)) \times rut\_paths$$

Combined Elevation: The total elevation of the terrain is the sum of the base elevation, furrows, and random ruts:

$$elevation = base\_elevation + furrows + ruts$$

Friction: The terrain's friction coefficient is designed to reflect moisture variation, with furrows being wetter (and thus, having lower friction) than the surrounding areas:

$$friction = 0.8 + 0.15 \cos\left(\frac{2 \cdot x}{size\_x}\right) - 0.1 \left|\sin\left(2\pi \frac{x}{1.5}\right)\right|$$

This equation decreases friction in the furrows, simulating the effect of moisture on soil grip.

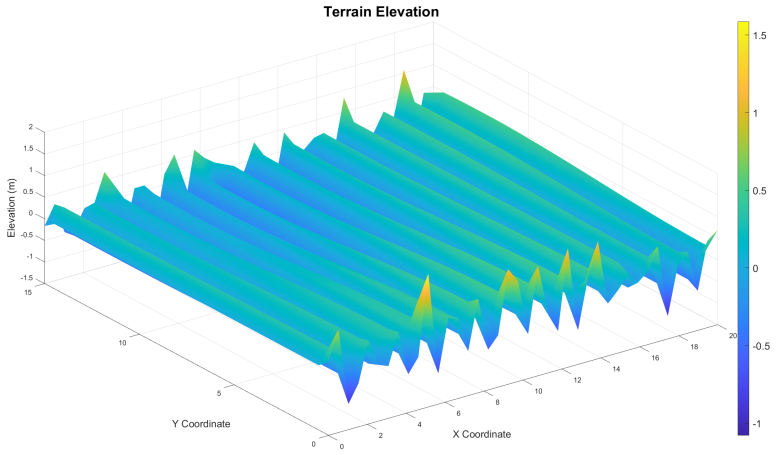

Fig. 1. Visulization of the terrain

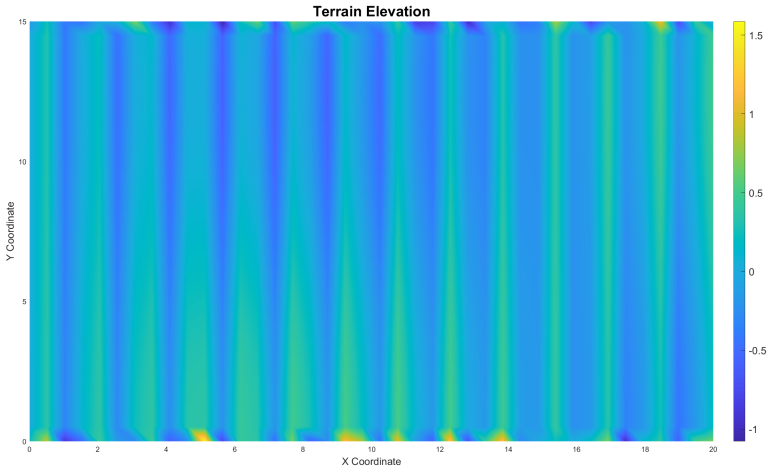

Fig. 2. Top View of the terrain

C. Obstacle Proximity Cost

The obstacle proximity cost function is designed to quantify the risk associated with the robot's proximity to obstacles within the environment. The function calculates a cost based on the robot's distance to each obstacle, with closer distances incurring higher costs.

Given the robot's position $x = [x_1, x_2]^\top$ and a set of obstacles, each defined by their position and a radius, the cost is computed as follows:

- For each obstacle $obs = [obs_x, obs_y, r]$, where (obs_x, obs_y) is the obstacle's position and r its radius, compute the distance from the robot to the obstacle's perimeter.
- If the robot is within a critical distance (the obstacle's radius plus a buffer zone), the obstacle contributes to the cost.

The mathematical formulation of the cost function is:

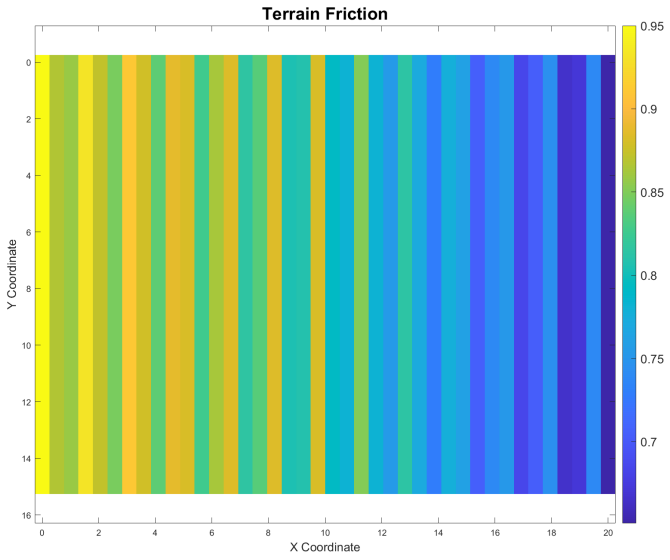

Fig. 3. Visulization of the friction across the terrain

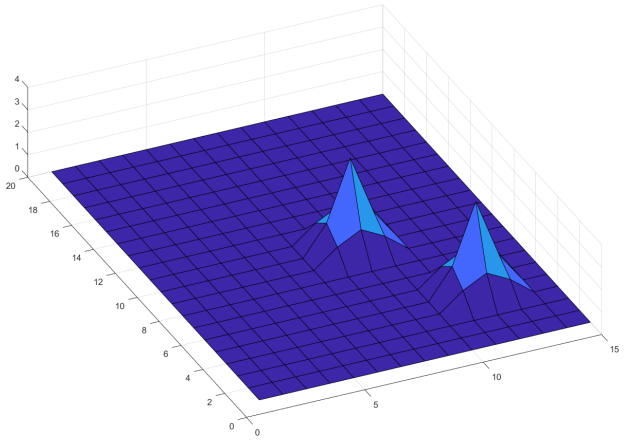

Fig. 4. Obstacle Proximity Cost

$$cost = \sum_{i=1}^{n} \begin{cases} \frac{1}{(distance_i)^2} & \text{if } distance_i < r_i + 1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $distance_i = \|x - obs_{i,1:2}\|$ is the Euclidean distance between the robot and the i-th obstacle, and r_i is the radius of the i-th obstacle. The cost increases significantly as the robot approaches an obstacle, reflecting the increased risk of collision.

D. Optimal Control Strategy

The function `optimize_control_sequence` calculates the optimal sequence of control inputs to guide a robot from an initial state x_0 to a goal state, while avoiding obstacles. Let's break down the process step by step.

E. Initial Guess

The initial guess for the control sequence is determined heuristically based on the robot's current state and the goal position:

- The direction to the goal is calculated using the angle between the robot's current position and the goal.
- The angular difference to the goal direction is adjusted to be within $[-\pi/2, \pi/2]$ to provide a reasonable initial guess for the angular velocity.

$$direction\_to\_goal = \tan^{-1}\left(\frac{goal_y - y}{goal_x - x}\right)$$

$$\text{Linear Speed: } v_0 = 0.2$$

$$\text{Angular Velocity: } \omega_0 = \text{sign}(\theta_{\text{diff}}) \cdot \min(|\theta_{\text{diff}}|, \frac{\pi}{6})$$

where θ_{diff} is the angular difference between the robot's current orientation and the direction to the goal.

The initial control sequence U_{initial} is created by repeating this initial guess for each of the N time steps.

F. Cost Function

- Time increment dt, to encourage faster arrival.
- Euclidean distance to the goal from the current position, promoting goal achievement.
- Obstacle proximity cost, to enhance navigation safety.

$$\begin{aligned} cost\_fn(u) = dt + &\|goal - robotPosition\| \\ &+ obstacle\_proximity\_cost \end{aligned} \tag{13}$$

The cost function computes the total cost by summing these components for each time step.

G. Optimization

The optimal control inputs are found by minimizing the cost function using a nonlinear optimization method, typically constrained by the robot's physical capabilities:

- The bounds for linear and angular velocities are set based on the robot's design specifications.

The optimization employs Sequential Quadratic Programming (SQP), a robust method for nonlinear constrained optimization. SQP iteratively solves a series of quadratic programming subproblems to approximate

the nonlinear objective and constraint functions. In this context:

- The nonlinear cost function $cost\_fn(u)$ is minimized subject to the control input bounds.
- The SQP algorithm iteratively updates the control inputs u, converging to the optimal solution that minimizes $cost\_fn(u)$ within the specified bounds.

The process is initialized with the computed initial guess and iterates until the change in the cost function is below a predefined threshold.

The optimization problem is formulated as:

$$u = \text{argmin}_u \, cost\_fn(u) \tag{14}$$

subject to $u \in [lb, ub]$, where lb and ub represent the lower and upper bounds for the control inputs.

The optimizer (`fmincon`) is used to find the optimal control sequence U_{seq} that minimizes the cost function.

Algorithm 1 Optimal Control Sequence Calculation

procedure $(x0, goal, obstacles, N, dt, terrain)$
  Calculate initial guess U_{initial}
  Define cost function $J(U)$ using `total_cost` function
  Define lower bounds lb and upper bounds ub on control inputs
  Run optimizer to find $U_{\text{seq}} = \text{fmincon}(J(U), U_{\text{initial}}, lb, ub)$
  return U_{seq}
end procedure

IV. Additional Functions

The function `total_cost` computes the total cost for a given state x, control sequence U, goal, obstacles, prediction horizon N, time step dt, and terrain map. It simulates the robot's dynamics using `robot_dynamics2` and calculates the total cost based on distance to the goal and obstacle proximity cost.

The optimal control strategy aims to determine the best set of control inputs $u = [v, \omega]^\top$ that steer the robot towards the goal while avoiding obstacles and minimizing travel time. The control inputs consist of linear velocity v and angular velocity ω.

A. Model Predictive Control (MPC) Implementation

The MPC framework in the agricultural robot context is designed to iteratively compute the optimal control inputs that guide the robot towards a goal while navigating around obstacles. The control strategy is executed over a finite prediction horizon.

1) MPC Framework: At each time step, the MPC algorithm performs the following steps:

1) Initialize the predicted trajectory with the current state x_0.
2) For each step in the prediction horizon N:
   a) Compute the optimal control input u using the current state, goal, and obstacle information.
   b) Simulate the robot's dynamics forward one time step using the optimal control input.
   c) Update the predicted trajectory and the current state for the next iteration.
   d) Check if the goal is reached within a predefined tolerance, terminating early if so.
3) The final optimal control input u is applied to the robot.

The process is mathematically depicted as follows:

For $i = 1$ to $N - 1$:

$u_i = \text{optimal\_controls}(x_i, goal, obstacles, dt, terrain)$,

$x_{i+1} = \text{robot\_dynamics}(x_i, u_i, dt, terrain)$,

if $\|x_{i+1}(1:2) - goal\| < 0.5$: break,

where x_i and u_i are the state and control input at the i-th step, respectively.

2) Conceptual Overview: MPC optimizes control inputs over a sequence of future time steps. At each step, a cost function is minimized to find the optimal control action. This cost function typically incorporates factors like the distance to the goal, the presence of obstacles, and the desired trajectory.

- x_0: Initial state of the robot.
- $goal$: Desired final position of the robot.
- $obstacles$: Known obstacles in the environment.
- N: Prediction horizon, the number of steps to look ahead.
- dt: Time increment for each simulation step.
- $terrain\_map$: The map describing the terrain over which the robot is navigating.

The key idea is to use the model of the robot's dynamics and the environment to predict the future trajectory and to adjust the control inputs accordingly to optimize the performance criteria.

V. Results

The algorithm was very slow to execute, taking up to 10 minutes depending on the position of the goal. This made testing many situations difficult.

First, a test was done with perfectly smooth ground, to verify our MPC implementation and provide a control case.

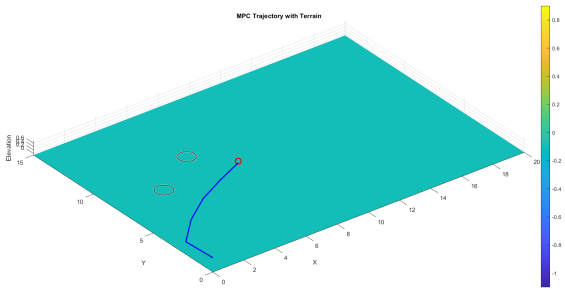

Fig. 5. Flat Terrain

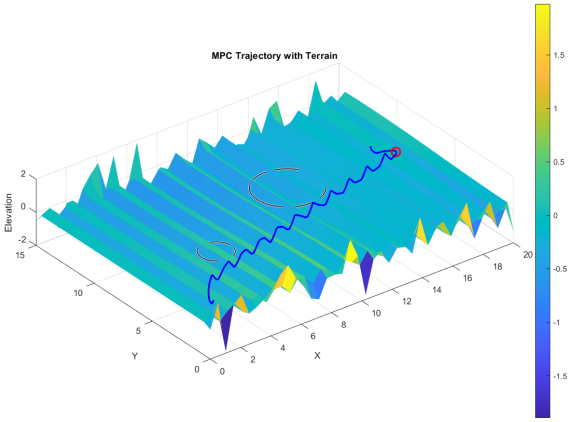

Fig. 6. Rough Terrain

Next, a series of tests were done with increasingly rough terrain, to determine the effect of the terrain roughness. The car followed a similar path to the test with flat terrain, but each furrow deflected the path around. The car was able to avoid obstacles and reach the goal even across unrealistically rough terrain.

Then, tests were done with one obstacle, and the car successfully navigated around the obstacle. However, the path taken does not seem smooth, the car seems to make abrupt turns, and loops around the obstacle farther than seems necessary before turning back to the goal.

The goal was placed closer to the start, and this seemed to make the path smoother.

Finally, a second obstacle was added. The car successfully avoided the obstacles, looping around between them.

However, if the obstacles were moved so a larger deflection was required, the car failed to avoid them.

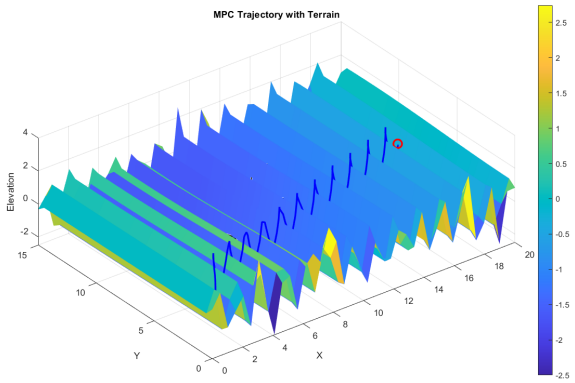

Fig. 7. Absurd Terrain

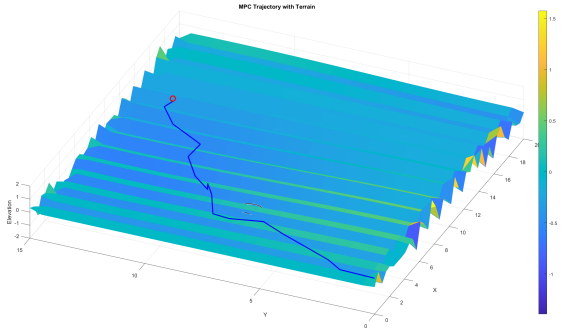

Fig. 8. 1 Obstacle

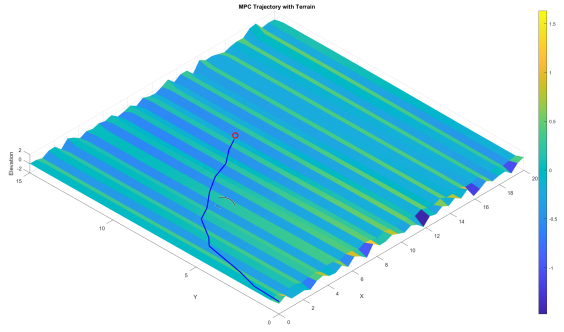

Fig. 9. 1 Obstacle With Closer Goal

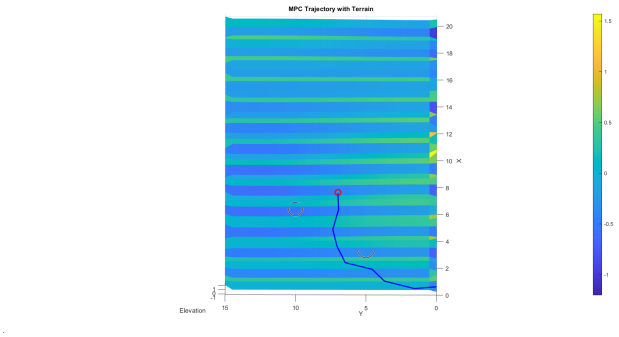

Fig. 10. 2 Obstacles

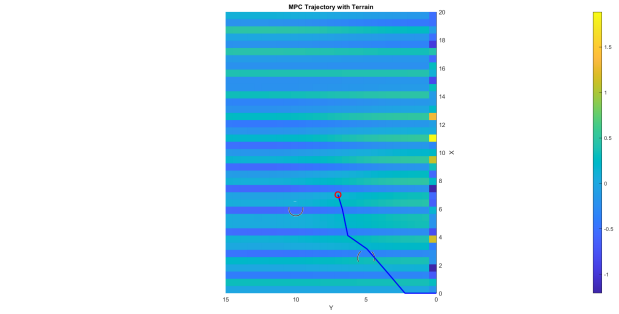

Fig. 11. The car failing to avoid an obstacle

VI. Conclusion

MPC was able to maintain it's course across very rough terrain. It was also able to avoid obstacles across rough terrain. However, if avoiding an obstacle required too much deflection (yet still within the maneuvering limitations of the car), it was unable to avoid the obstacle.

This may have been because our MPC algorithm did not have a hard constraint to avoid obstacles. Instead, a high cost was given to intersecting them. A way of improving this would be to switch to a "safe" controller when near an obstacle, so that the car avoids the obstacle at all costs instead of balancing avoiding the obstacle with approaching the goal. One such controller is Backward Reachable Tube computation, which would find the areas which the car must not enter or it will inevitably contact an obstacle.

References

[1] Terrain Aware Model Predictive Controller for Autonomous Ground Vehicles - Sina Aghli, Christoffer Heckman Department of Computer Science, University of Colorado Boulder

[2] Model Predictive Control for Autonomous and Semiautonomous Vehicles by Yigi Gao

[3] A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments by Jiechao Liu, Paramsothy Jayakumar Jeffrey L.Stein and Tulga Ersal - University of Michigan