Path Planning for Autonomous Vehicles based on Nonlinear MPC with using a Kinematic Bicycle Model

Jonas Wagner

***Abstract*—In this project, NOVA's autonomous vehicle, Hail Bopp, is modeled as a 3DOF Kinematic Bicycle Model and then Nonlinear MPC techniques are used as a path-planning method. This project also explored an implementation of nonlinear MPC that is then simulated in close-loop.**

Copy stuff from Proposal

Include NOVA stuff

I. Introduction

II. Problem Definition

A. Path Planning

The objective is to create high-fidelity MPC controller to produce an optimal trajectory to reach a waypoint given the occupancy map of the environment. This controller will then be used to generate training data for a neural network which will be able to perform an approximation of this controller in real-time.

From the perception stack, the current vehicle states (local and global) will be known to some uncertainty and the surrounding environment will be processed into a predicted occupancy map. This occupancy map will be assumed to already have weights corresponding to where it is safe/ideal for the vehicle to be in the future.

B. Model Definition

1) 3-DOF Bicycle Model: Hail-Bopp is considered to be a standard 4-wheel ackerman vehicle. The vehicle model will be based on the 3-dof bicycle model as derived in [1] and [2].

Due to computation complexity, the kinematic model derivation is used that directly relates the

Simple non-linear kinematics model equations:

$$\begin{cases} \dot{x} = V\cos(\psi + \beta) \\ \dot{y} = V\sin(\psi + \beta) \\ \dot{\psi} = \dfrac{V\cos(\beta)}{l_f + l_r}(\tan(\delta_f) - \tan(\delta_r)) \\ \dot{\theta} = \psi \end{cases} \tag{1}$$

Provide the expanded model and derivation in the appendix

where

$$\beta = \tan^1\left(\frac{l_f\tan(\delta_r) + l_r tan(\delta_f)}{l_f + l_r}\right) \tag{2}$$

2) Model Discretization: The nonlinear kinematic model is discretized using an RK4 approximation. The selected discretization step-size was tested to range between $\Delta t \in [0.01, 5]$ and unlike the Euler update approximation, the RK4 approximation seemed effective at larger time-steps (specifically $\Delta t = 1$ [s] for the selected results).

Elaborate on RK4 approx-

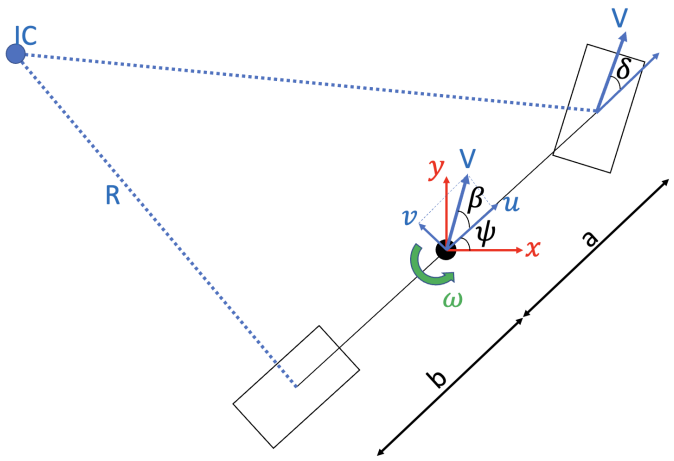

Fig. 1.

III. Problem Solution

The MPC Controller for path planning is formulated with the current state and model update equations as hard constraints, an objective to minimize the time and distance to reach the future waypoint, and introducing the occupancy map as soft-constraints within the objective function.

A. Model Constraints

1) Update Equations: Using the discretized model, an update function $x_{k+1} = f(x_k, u_k)$ which is implemented for each timestep.

2) Input Constraints: To represent the input constraints of the actual vehicle, interval constraints upon both the value and change of the inputs:

$$\begin{gathered} V \in [0, 10] \text{ [m/s]} \\ \theta \in \pm\pi/3 \text{ [rad]} \\ \dot{V} \in [-4, 2] \text{ [m/s}^2\text{]} \\ \dot{\theta} \in \pm\pi/10 \text{ [rad/s]} \end{gathered}$$

This is converted into two half-space representation constraints: $h_u(u_k) \leq 0$ and $h_{\dot{u}}(u_{k+1}, u_{k-1}) = \frac{u_{k+1} - u_k}{\Delta t} \leq 0$. These half-space representations are visualized as two sets shown in Fig. 2.

B. Cost Function

The primary cost function will be linear from the system state as derived the cost map as the sum of the region that the vehicle would occupy. An example is shown in Fig. 3.

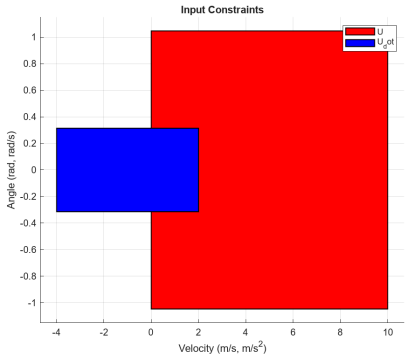

Fig. 2. Halfspace visualization of the input constraints.

Fig. 3. An example occupancy map provided from the perception stack. The

This is first done by directly converting the occupancy map of the vehicle in positions into a function of the system state, visualized in Fig. 4, as just $f_{map}(x_k)$. Note that this is also normalized and the function itself is currently treated as a piecewise function (unfortunately causing issues within implementation).

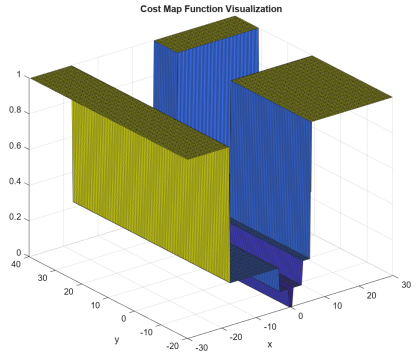

Fig. 4. An example cost map after converting into a direct function of system state and further interpolating.

Additionally, an objective will be included that describes the forward progression along a proposed route, $f_{progress}(x_k)$.

As was found out within actual implementation, the occupancy map was additionally simplified to a static function relative to the initial path for each time-step to remove the piecewise-nature or interpolation required by the full cost-map.

The results section will include simulation results for specific demonstrative cost-functions. Note that in the results, the actual path is generated each time-step and the primary visualization as that which closes the loop and actually implements the desired inputs (i.e. actual MPC). In the actual system, the PID would track the generated state-trajectory while the inputs are not explicitly used (i.e. similar to hieratical MPC).

IV. Simulation and Results

The actual system was modeled MATLAB using multiple toolboxes (yalmip [**?**], MPT3 [**?**], gurobi [**?**], and IPOPT [**?**]) with the code provided in section A.

Many simulations were ran and the few included here are used to demonstrate the simpler operations.

The follwoing parameters were used within these results:

- $N = 15$
- $\Delta t = 1$ [s]
- Multiple $q(x_k, u_k)$ and $p(x_n)$ tested
- $h_u(u_k) \leq 0$
- $h_{\dot{u}}(u_{k+1}, u_{k-1}) = \frac{u_{k+1} - u_k}{\Delta t} \leq 0$

The simulations were ran with $T = 10$ [s] and the results for both fmincon and ipopt are provided.

One success includes a 90 degree turn and just maximization of the distance in that direction, shown in Fig. 5.

One "failure" for fmincon inlcudes that of a U-turn, shown in Fig. 6

Put the most relevant results here.

include the rest of the results

V. Conclusion

The biggest conclusion from this project is that this direct form of nonlinear MPC is not currently a feasible solution for NOVA to do path planning, even in the case for just generating training data for the nueral network.

Additionally, this project was a major demonstration of how using different optimization solvers can have a major impact on the computation time, but also the results themselves. A rough testing comparison of the different testing times is included in **??** and clearly IPOPT was far faster in every test, but even so it is not nearly fast enough to generate a lot of training data let alone for online operation.

TABLE I
Computation time comparison between FMINCON and IPOPT

	FMINCON	**IPOPT**
Min Y - Unstructured	243.79	164.75
Min Y - Final	235.91	155.6
Min Y - Ref X and theta	438.67	71.79
Min X - Final	**1077.6**	**164.58**
Min X with +180	395.71	191.69
Min X with special ref	**344.52**	**165.16**
Max Y - Final	**331.71**	**140.49**
Max Y - Ref X theta	**665.82**	**85.492**

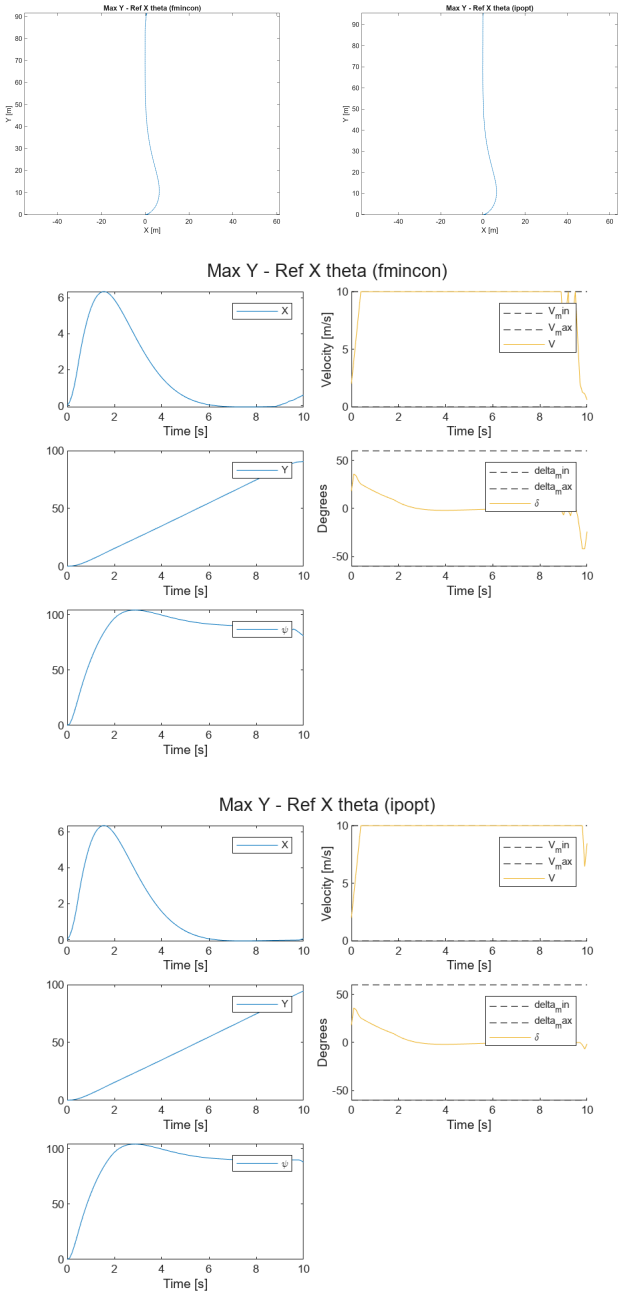

Fig. 5. Success 90 deg result

REFERENCES

[1] D. Casanova, "On minimum time vehicle manoeuvring: The theoretical optimal lap," 2000.

[2] P. P. Ramanata, "Optimal vehicle path generator using optimization methods," Ph.D. dissertation, Virginia Tech, 1998.

ensure all re-frences are in-cluded

APPENDIX

See my github repo for this project for all the source code and additional results: https://github.com/jonaswagner2826/MECH6V29_MPC_FinalProject

MATLAB code is included in submission.

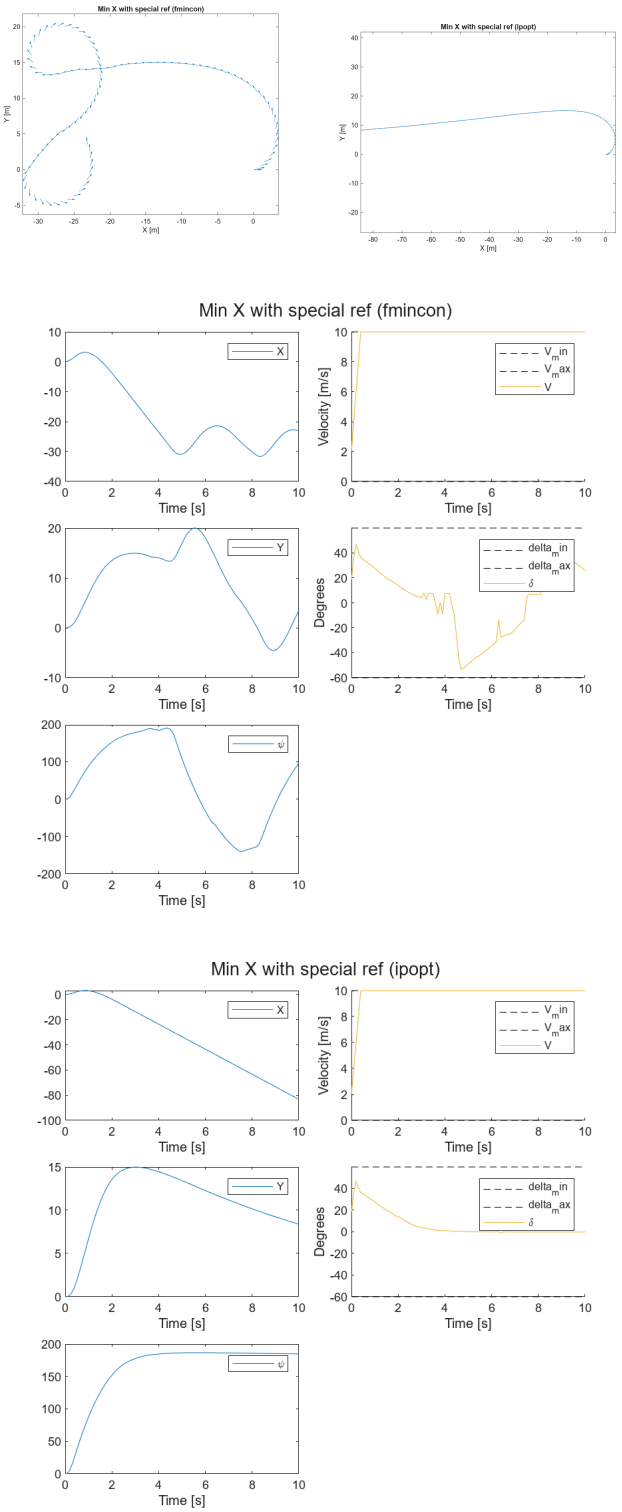

Fig. 6. Failure result