HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 3 (2023)

Smart Path Generation using Model Predictive Control

Sandeep Bajamahal

Saint Francis High School

ABSTRACT

This paper presents an investigation into path generation techniques using Model Predictive Control (MPC) and Pure Pursuit algorithms, implemented and evaluated in a simulated environment using Python. The objective of the study was to compare the performance of these two approaches in terms of path tracking and obstacle avoidance. The research focused on the applicability of MPC and Pure Pursuit algorithms in autonomous navigation systems, with a specific emphasis on addressing the challenge of generating smooth and dynamically feasible paths while ensuring collision avoidance. The simulation environment provided a platform for conducting experiments, allowing for testing and analysis of the algorithms. The results of the study demonstrated that MPC successfully generated paths while effectively avoiding obstacles. The MPC algorithm exhibited robustness and adaptability to dynamically changing environments, allowing the autonomous agent to navigate through complex scenarios. However, the investigation also revealed a limitation with Pure Pursuit in terms of curvature volatility. The Pure Pursuit algorithm showed inconsistent performance due to abrupt changes in curvature, which impacted the smoothness and stability of path tracking. Overall, this research highlights the significance of selecting an appropriate path generation algorithm based on the specific requirements of the autonomous navigation system. The study serves as a foundation for future investigations and advancements in path planning and control techniques, enabling the development of more efficient and reliable autonomous systems.

Introduction

Path generation is the process of determining a sequence of waypoints that guide a certain entity to a desired destination. Path generation is commonly used in GPS navigation, autonomous driving, robotics, and flight control in order to avoid obstacles or minimize travel time. There are various path generation algorithms aimed at addressing these use cases. One challenge that these path generation algorithms face is dealing with a dynamic environment, where a path may need to be modified during traversal time. Furthermore, path generation algorithms often must remain within certain constraints on motion, such as vehicle limitations and energy efficiency. There are algorithms that address these issues, such as Probabilistic Roadmaps (PRMs). PRMs reframe their environment into a sampling of nodes, from which they create a graph. The graph is then searched for the shortest path from the entity to the desired node, using shortest-path algorithms such as Dijkstra's algorithm or A* search. However, one limitation of PRMs is their inability to deal with moving obstacles, which may eliminate the viability of a chosen shortest path. A solution to path generation in dynamic environments is a hybrid path generation and following algorithm between Model Predictive Control (MPC) and Pure Pursuit. MPC is a range of algorithms that are used to optimize a system input in order to approach a future output. MPC has varied uses because of its ability to handle multiple inputs and constraints. Some of these uses are in power systems, spacecraft fuel optimization, and chemical refining plants. While MPC is used to optimize systems, it can be modeled to optimize a path iteratively for the on the fly path generation. Since MPC takes into account multiple inputs and constraints, it can deal with object avoidance and minimize curvature or path distance. To simulate the MPC path, the Pure Pursuit path following algorithm can be used as it is adaptable to the MPC's path optimizing during traversal time.

Overview of Model Predictive Control

Model Predictive Control requires a model of a system dynamic with inputs and outputs. This dynamic model can be a set of equations that defines how the system works, or a simulation. Next, a desired system output must be supplied. In path generation, this would be a coordinate position. To get to the desired system output, the MPC algorithm will use a prediction horizon. The prediction horizon is a time period into the future which is divided into a finite number of steps. At each step, the MPC algorithm will calculate the optimal set of inputs to get to the desired system output. This set of inputs, or path, can be optimized to minimize any objective function provided. The first input from the optimal set will be performed, and the MPC algorithm will recalculate the next set of inputs from the system's new output. This prediction horizon is shifted forward during this process.

Overview of Pure Pursuit

Pure pursuit is a path-following algorithm commonly used in robotics and autonomous navigation. The algorithm requires a desired path, either as a set of waypoints or a parametric curve. The tuning factor of the algorithm is the lookahead point, which is a point on the desired path that is some distance ahead of the vehicle. The algorithm calculates the curvature to this lookahead point a geometrical calculation (insert image here).

Design Overview

The proposed combination of Model Predictive Control and Pure Pursuit would work as follows. MPC would be used for high level path planning that takes into account obstacle avoidance, constraints, and optimizations. Once MPC generates a set of optimal waypoints, the Pure Pursuit algorithm can calculate the steering necessary to reach the next waypoint. Any error from the Pure Pursuit path following algorithm will be used to readjust the MPC generated path for the next set of optimal waypoints.

Procedures

Simulation Overview

The First Robotics Competition (FRC) Python package was used to access the Worchester Polytechnic Institute Robotics Library (WPILib), a robotics library used by FRC teams to program industrial-scale robots. WPILib has simulation capabilities for motors, drivetrains, and field positioning. For the purposes of this experiment, a four-wheel drivetrain with a wheelbase distance of two meters was simulated. A physics engine was also written in order to calculate encoder readings from the robot's movement in order to simulate actual robot sensor input. In the simulation, the robot has the capability to move around in a two-dimensional space.

Algorithm Implementation

The MPC Algorithm was programmed to generate a parabola from the robot's current position to a desired destination. The parabola was constrained to not intersect an obstacle in the shape of a rectangle, and was minimized for the least curvature possible. Minimizing curvature also minimizes distance, as the path becomes closer to a straight line, which is the shortest path between any two points. The SciPy library was used to perform the optimizations under specific constraints for the parabola. The first trial of tests included a single obstacle which the robot must go around to reach a desired destination using parabola paths.

The Model Predictive Control and Pure Pursuit algorithms were implemented in a loop as follows. Every twenty milliseconds, the Model Predictive Control generates an optimal parabola path for the robot to reach the target point. A lookahead point is taken from this parabola, and the curvature from the robot's current position to the lookahead point is calculated using the following formula.

$$p(x) = a\,x^2 + b\,x + c$$

$$C = \frac{2a}{|1 + b^2 - 4a\,c + 4a\,p(x)|^{\frac{3}{2}}}$$

The curvature is fed into a curvature drive function provided by WPILib, which sends input to the simulated left and right motors of the robot drivetrain accordingly such that they drive at a certain curvature.

Data and Analysis

During the simulation, robot position, time, and path details were collected and analyzed. The robot began at the origin of a coordinate plane facing directly to the right and had a target point at (5, 5). The obstacle was a square bound from (2, 2) to (3, 3). Every instant, the robot calculates a path from its position to the target point.

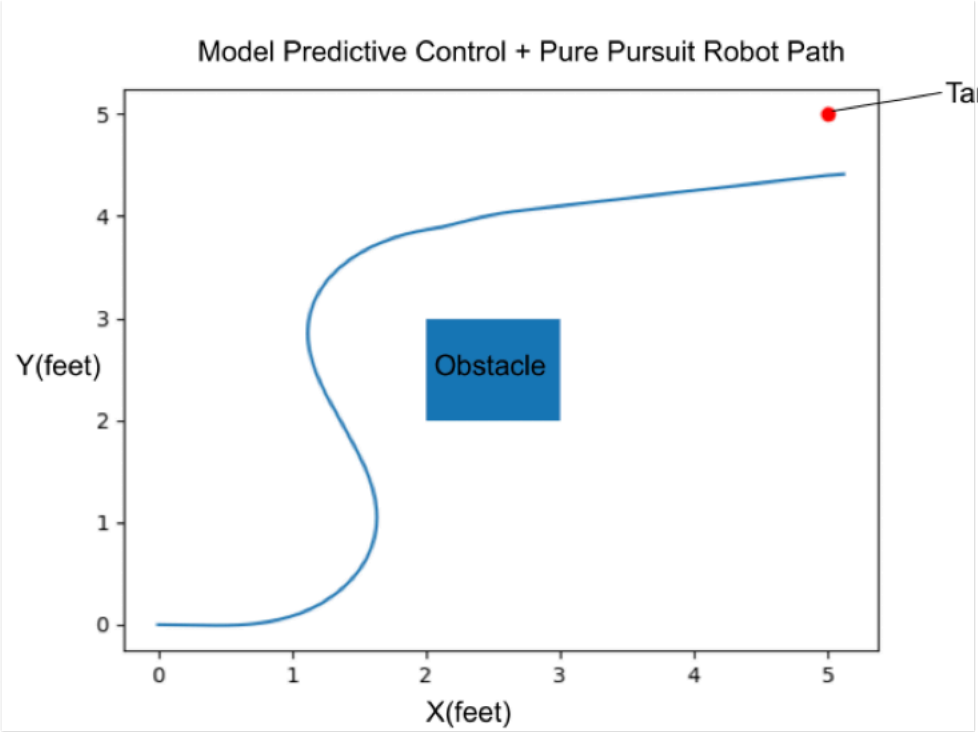

Figure 1: Model Predictive Control + Pure Pursuit Robot Path plotted using Matplotlib

The robot's final position is at (5, 4.405). The robot manages to avoid the obstacle and approach the target point, but is not able to accurately steer to the correct position.

Table 1: Robot position and curvature over time.

Time (Seconds)	X (Feet)	Y (Feet)	Curvature (1/Foot)
1.0	1.267	0.260	-0.014
2.0	1.605	1.316	-0.015
3.0	1.145	2.651	0.637
4.0	1.565	3.686	0.171
5.0	2.790	4.044	0.058
6.0	4.286	4.195	0.050
7.0	5.000	4.405	-0.017

The robot path did not always approach the target point. Often the curvature-following algorithm caused the robot to take more of a turn than necessary when avoiding obstacles. Furthermore, the accuracy of the robot was dropped by the curvature path following algorithm, as the robot position was an average of 0.102 feet away from the waypoint it was expected to be throughout the simulation. Out of all the paths generated by the MPC algorithm, 75% of the paths were valid and avoided obstacles. The remaining paths were not optimized or valid for the robot.

Discussion

Conclusion

The results of the simulation of MPC path generation along with Pure Pursuit path following indicate that the two algorithms are incompatible with a quadratic system, although there are various path models that remain to be tested. The curvature approach to Pure Pursuit also proved to be inaccurate when combined with a changing path. However, the path generation capabilities of MPC proved to be sufficient for obstacle avoidance, with a high rate of valid paths generated throughout the simulation.

Applications

The adaptable path-generation techniques used in this experiment would be applicable to volatile environments found in flight control and autonomous driving. Furthermore, the combination of path-following algorithms and path-generation algorithms must be optimized depending on their usage. This is key for vehicles that have physical constraints on curvature change, such as trucks or buses due to their high center of gravity. MPC is optimal for vehicles with higher curvature thresholds. The nature of parabola paths and Pure Pursuit proved to be capable of obstacle avoidance, however a more robust equation should be used to handle both obstacle avoidance and minimal distance path generation.

Limitations

With a more robust path-following algorithm and simulation setup, MPC path generation could have also been tuned for optimizing velocity, which would be more representative of a real-world environment.

Future Research

There are various path equations that can be used with MPC to handle obstacle avoidance and shortest-distance path generation. Furthermore, the combination of MPC and Pure Pursuit may require a different approach from curvature, such as angle-based Pure Pursuit. This would allow the robot to split a path into a sequence of straight lines and turns which can be followed accurately with a proportional term controller.

Acknowledgements

I would like to thank my school robotics team for providing me with guidance and the technologies required to perform this study.

References

Camacho, E. F. (2013). *Model predictive control in the process industry.* Springer London Ltd.

Coulter, C. R. (1990). *Implementation of the Pure Pursuit Path Tracking Algorithm.* https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf.

Probabilistic roadmaps for path planning in high-dimensional ... (n.d.). https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/PRM/prmbasic_01.pdf