Terrain Aware Model Predictive Controller for Autonomous Ground Vehicles

Sina Aghli, Christoffer Heckman
Department of Computer Science
University of Colorado
Boulder, CO 80309

I. Introduction

With rapid progress in the development of AGVs, the need for more stable and accurate controllers which can accept more complicated vehicle dynamics into consideration while handling environmental uncertainties is greater than ever. Research in "enhanced driver assists" like adaptive cruise control [8] or lane-keeping [16, 17] for cars has been widely researched in the past. These control systems largely concern themselves with stabilizing a vehicle to a desired state. Some vehicle models has been significantly simplified in order to have differentiable dynamics while considering other dynamic effects as noise input to the system, e.g. [6, 12]. Common simplifications include ignoring collision forces due to their complicated effects. However, in order to achieve a stable control response it is required to use as much information from the environment as possible to make action devisions. Developments in new sensor technologies and estimation algorithms allow control systems engineers to capture more detailed information from the environment like 3D pose estimation of the robot or depth maps of the environment using cameras[7] or depth sensors [13]. Modern control algorithms should use this information in their determinations.

In contrast to classical control methods in which simplified dynamics of the AGV is discovered in form of explicit equations, and a controller is designed to stabilize that model, model-predictive controllers (MPC) [1] solve the optimal control problem in every time step. To do this, the algorithms are given the current state of the model up to a finite time horizon, but the solution to this problem (control command) is only applied to the system during the next time step and for the step after that it needs to be recalculated. Solving an optimization problem for every time step makes MPC algorithms computationally expensive, specially when dynamics of the vehicle gets more complicated finding a solution to this problem might take a longer time than time step period. Note that control problem requires a goal and this case AGV would need a reference trajectory to follow. Nagy et al. [15] proposed a method to parameterize a trajectory for car-like robots using cubic curvature polynomials which is a good estimate for path of the AGV in flat surfaces but it might not be accessible by the robot in a rough terrain.

In order to find a solution to these problems, we wish to develop an intuitive, computationally efficient method that is robust to uncertainty. In this work, we present an approach to modeling, planning and control of an AGV. In order to model the AGV we use a high-fidelity physics simulation engine [5] which takes the dynamics of the AGV into account as well as terrain roughness and contact forces applied to the AGV. Similar to [10], we use a dynamics engine to solve two problems: 1) the reachability problem between two waypoints constrained to AGV dynamics, and 2) the corresponding optimal control problem.

II. Problem Statement

We take a model predictive approach to AGV control; in this framework, the controller leverages a high fidelity vehicle model and prior environment maps in order to calculate the optimal control signal for the AGV. We also develop a local planning method which finds an optimal reference trajectory between waypoints defined by a higher level global planner. Later this trajectory is used as a reference trajectory for model predictive controller.

In our framework, modeled dynamics of the AGV should capture behavior of AGV as accurately as possible. In order for the AGV to travel from one waypoint to another, it is required to first examine the reachability condition between the two waypoints. If the final waypoint is reachable, then it is possible to design the controller in order to navigate the AGV to the subsequent waypoint.

III. Methodology

A. Modeling

In order to predict behavior of the AGV we create a dynamics model of the vehicle as well as the terrain in a physics engine[5]. In this work we evaluate the performance of this method on a front-steering four wheel drive vehicle with spring-damper suspension on each wheel attached to a cuboid chassis. The terrain can be specified as a dense three-dimensional mesh with a friction coefficient specified for each mesh surface. The governing equations of motion for AGV are

$$\boldsymbol{x}'(t) = \Phi(\boldsymbol{x}(t), \Pi(\boldsymbol{x}(t), \boldsymbol{g}, \boldsymbol{M}), \boldsymbol{u}(t), \boldsymbol{p}), \tag{1}$$

in which $\Phi(\cdot)$ represents the modeled dynamics of the AGV, $\boldsymbol{x}(t)$ is current state of AGV, and $\boldsymbol{x}'(t)$ is the derivative of the state under the governing dynamics. Π represents contact forces applied to the AGV and surface frictions at a given state $\boldsymbol{x}(t)$, with gravity vector $\boldsymbol{g}$ and the map of terrain $\boldsymbol{M}$.

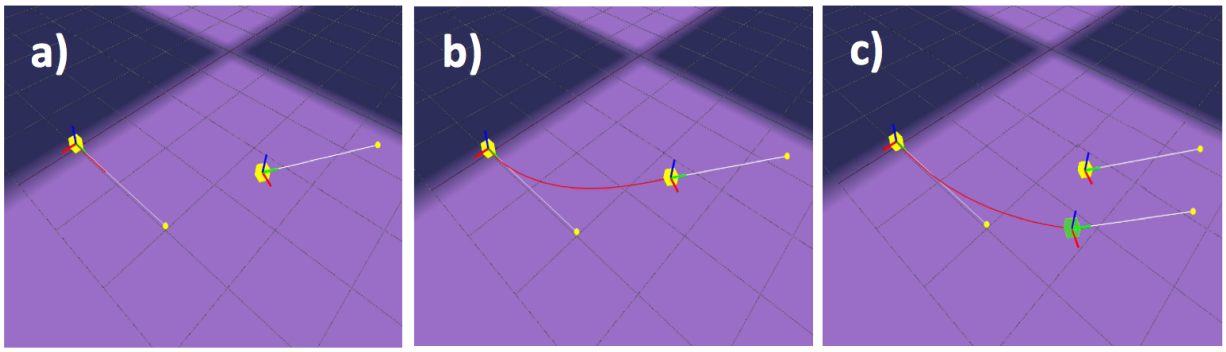

Fig. 1: A perspective view of the graphical front-end to our MPC software. a) Waypoints with pose and velocity constraints represented by the yellow boxes and white lines respectively. b) A resolved solution after solving the 2-point BVP is designated by the red line. The presence of the red line connecting the waypoints designates that the second waypoint is reachable. c) When no solution is feasible (in this case, because environmental parameters were varied), a closest-feasible waypoint is generated in green.

Note also that $\boldsymbol{u}(t)$ are the steering and acceleration inputs, $\boldsymbol{p}$ is vehicle parameter vector. This parameter vector includes vehicle properties like tire and chassis geometry, as well as their inertial properties, linear and rolling friction coefficients of all bodies, suspension anchor points, suspension damping and stiffness coefficients as well as their travel limits, motor speed-torque parameters for steering and acceleration motors and chassis's center of mass location.

The control input $\boldsymbol{u}(t)$ is a continuous signal which must be discretized. As an added challenge, sampling this signal in high frequency would result in a problem with very high computation cost. To avoid this the control input signal, we apply a parametrization of this signal by a quadratic Bézier curve [9]. We may then rewrite $\boldsymbol{u}(t)$ as

$$\boldsymbol{u_p}(t) = (1-t)^2 p_0 + 2(q-t)tp_1 + t^2 p_2. \tag{2}$$

We will later discuss the implications this has in simplifying the corresponding boundary value problem; in particular, it reduces the problem significantly by solving for p_i, $i \in [0, 2]$ rather than discrete points on $\boldsymbol{u}(t)$.

It is possible to solve Eq. (1) with algorithms like MLCP Dantzig, MLCP Projected-Gauss-Seidel or Sequential Impulse Solver, which are already implemented in the Bullet physics engine [4]. Note that as with all algorithm choices, swapping the solver has an effect on the speed of computation and accuracy of solution.

B. Local Planning

Given two waypoints on the mesh we evaluate reachability of waypoints constrained to AGV dynamics. We also come up with a reference trajectory for the AGV to be followed by the MPC algorithm. By design, a waypoint $\boldsymbol{w_i}$ defines a pose and velocity constraint as

$$\boldsymbol{w}_i = [\boldsymbol{d}_i, \boldsymbol{r}_i, \dot{\boldsymbol{d}}_i, \dot{\boldsymbol{r}}_i], \tag{3}$$

in which $\boldsymbol{d}_i$ is position vector of the AGV in global Cartesian coordinates, $\boldsymbol{r}_i$ is vector of roll, pitch, and yaw angles; the next two terms are the velocity constraint vectors in that waypoint.

In order to evaluate reachability of waypoint $\boldsymbol{w}_{i+1}$ from $\boldsymbol{w}_i$ we must construct a BVP from Eq. (1). Dropping the Π and p arguments for simplicity and replacing $\boldsymbol{u}(t)$ from Eq. (2), we have:

$$\begin{aligned} p^*_{bvp} = \min_{p} \quad & \boldsymbol{\lambda} \| \boldsymbol{x}_{\boldsymbol{w}_{i+1}(t_{i+1})} \boxminus \boldsymbol{\Phi}(\boldsymbol{x}_{\boldsymbol{w}_i}(t_i), \boldsymbol{u_p}(t)) \|_2 \\ & + \gamma \| t_{i+1} - t_i \|_2 \end{aligned} \tag{4}$$

where $\boldsymbol{x}_{\boldsymbol{w}_i}(t_i)$ is state of vehicle at time t_i with boundary conditions $\boldsymbol{w}_i$, and $\boldsymbol{\lambda}$ and scalar γ are weighting parameters. The operator $\boxminus$ calculates the velocity and pose error between two vehicle states. The first term in the cost function of optimization problem in Eq. (4) defines an error vector between waypoint $\boldsymbol{w}_{i+1}$ and solution of dynamic model 1 starting from $\boldsymbol{w}_i$ with parameterized input $\boldsymbol{u_p}(t)$. The second term is a regularization term which results in a time-optimal trajectory. Eq. (4) is solved via an optimization framework which is discussed later.

After solving Eq. (4), if a solution is achieved with cost value lower than some ϵ, then $\boldsymbol{w}_{i+1}$ is considered reachable (Figure 1b). However, if the error is greater than ϵ, then the waypoint is considered not reachable (see Figure 1b). In this case, one can decide whether to use the solution to adjust $\boldsymbol{w}_{i+1}$ to the current solution and continue to next waypoint or terminate the algorithm. In addition to solving for p^* from Eq. (4), we also store the trajectory of the AGV between every two consecutive waypoints:

$$\boldsymbol{\tau}_j = \Phi_{\delta_j}(\boldsymbol{x}_{\boldsymbol{w}_i}(t_i), \boldsymbol{c}(p^*_{bvp})) \quad j \in \{0 \ldots \lfloor \frac{t_{i+1} - t_i}{\delta_t} \rfloor\}, \tag{5}$$

where $\delta_j = t_i + j * \delta_t$ and δ_t are the time step sizes to sample the trajectory. $\Phi_{\delta_j}(\cdot)$ is the state after time δ_j, with the input $\boldsymbol{c}(p^*)$ applied to dynamics model.

C. Control

Once a reference trajectory has been calculated as in Eq. (5), we design a model predictive controller such as that employed by [2] to stabilize the AGV to the reference trajectory. For this we define a "cost-to-go" function as in Eq. (6) and construct

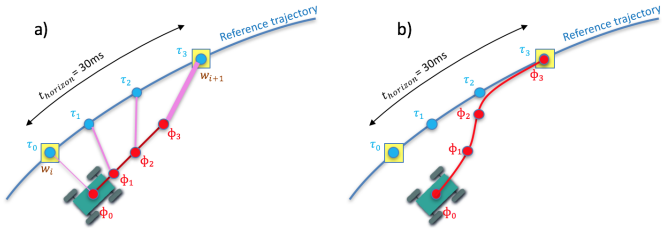

Fig. 2: Model predictive controller cost function with (red dots) discretized vehicle trajectory points, (blue dots) discretized reference trajectory points, (yellow box) waypoint, (purple line) weighted costs. a) AGV with initial control input results in red trajectory before optimization. b) AGV's trajectory converges to reference trajectory after convergence

an optimization problem in order to find the proper control signal:

$$p^*_{mpc} = \min_{p} \sum_{j=0}^{K} \rho_j \| \tau_j \boxminus \mathbf{\Phi}_{\delta_j}(\boldsymbol{x}_{\boldsymbol{w}_i}(t_i), \boldsymbol{c}(p)) \|_2, \quad (6)$$

where ρ_j is weighting coefficient and K is the length of the sequence of states over the discretized time horizon. Since most vehicles have control constraints arising from the steering angle of the wheels, one might apply less weight to closer points to the current position of AGV since control signal can not reduce the error due to nonholonomic constraint of wheels. Figure 2 shows the trajectory of a four-wheeled AGV before and after solving Eq. (6). Once a control command $\boldsymbol{c}(p^*_{mpc})$ is calculated, it gets executed on the AGV for a short period of time—in our case, until next control command is calculated.

D. Optimization

For both BVP and MPC problems we minimize a cost function in a least-squares sense. We solve these optimization problems using the Levenberg-Marquardt [14] algorithm. We also must calculate the Jacobian of the error function with respect to its parameters $\boldsymbol{J} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{p}}$, for which we take a finite differencing approach to calculate derivatives of Eq. (1). Since columns of the Jacobian matrix are not correlated with one another, finite differencing can occur in parallel. Our multi-threaded algorithm is capable of calculating the Jacobian at each iteration of optimization quickly.

IV. Results and Conclusions

In order to test the planning algorithm, a simulator for four wheel drive AGV with suspensions has been developed (figure 3). Implementing the local planning approach from Section III demonstrated that we can validate reachability of a waypoint and also come up with a reference trajectory for the AGV. We also parameterized the control input by Bézier curves Choi et al. [3] to reduce dimensionality of search space from infinite dimensional continues functions to quadratic curves. Figure 3 shows the resulted reference trajectories for eight consecutive waypoints.

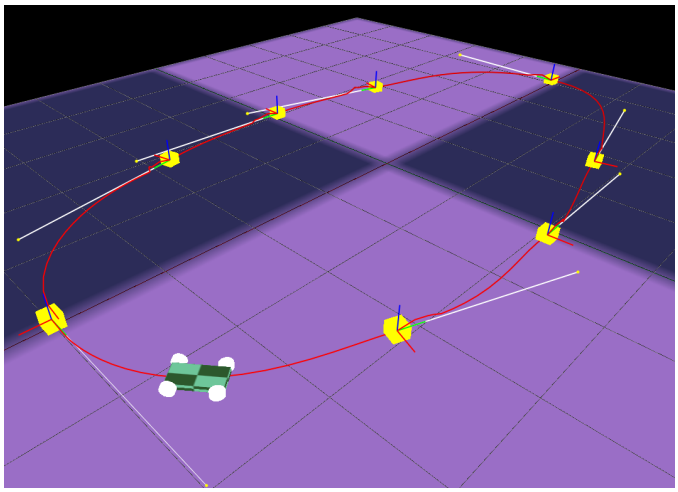

Fig. 3: Simulated four wheel drive AGV with eight waypoints(yellow) and corresponding trajectories(red)

Our experiments showed that dynamics engines may be used to model complicated behavior of AGVs and the same model could be used in local planning and control problems without further modification. This simplifies the control problem to defining geometric and dynamic properties of the AGV and defining some waypoints for the robot to follow.

V. Future Work

Similar to [11], we plan to constantly compare the expected behavior of actual AGV to reality and identify anomalies could define failures in actual robot. This will help mitigate parameter tuning by automatically regressing vehicle calibration parameters. The same algorithm could be used in order to detect behavior changes in robot actions and either recalibrate some parameters or report failures in the robot. We also see potential gain in designing algorithms which initialize input command for both BVP and MPC problems since they could fall in local minimum. Last but not least, we plan to test this approach on an actual AGV to determine robustness to real-world uncertainties in vehicle models, terrain descriptions and perception output.

References

[1] F Borrelli, A Bemporad, and M Morari. Predictive control for linear and hybrid systems lecture notes. *UC Berkley*, 2014.

[2] Patrick Bouffard, Anil Aswani, and Claire Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 279–284. IEEE, 2012.

[3] Ji-wung Choi, Renwick Curry, and Gabriel Elkaim. Path planning based on bézier curve for autonomous ground vehicles. In *World Congress on Engineering and Computer Science 2008, WCECS'08. Advances in Electrical and Electronics Engineering-IAENG Special Edition of the*, pages 158–166. IEEE, 2008.

[4] Erwin Coumans. Bullet physics library. URL http://bulletphysics.org/.

[5] Erwin Coumans. Bullet physics engine. *Open Source Software: http://bulletphysics. org*, 1, 2010.

[6] Alessandro De Luca, Giuseppe Oriolo, and Claude Samson. Feedback control of a nonholonomic car-like robot. *Robot motion planning and control*, pages 171–253, 1998.

[7] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38. Springer, 2010.

[8] Petros A Ioannou and Cheng-Chih Chien. Autonomous intelligent cruise control. *IEEE Transactions on Vehicular technology*, 42(4):657–672, 1993.

[9] Mike Kamermans. A primer on bzier curves. *https://pomax.github.io/bezierinfo/*.

[10] Nima Keivan and Gabe Sibley. Realtime simulation-in-the-loop control for agile ground vehicles. In Ashutosh Natraj, Stephen Cameron, Chris Melhuish, and Mark Witkowski, editors, *Towards Autonomous Robotic Systems*, volume 8069 of *Lecture Notes in Computer Science*, pages 276–287. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-43644-8. URL http://dx.doi.org/10.1007/978-3-662-43645-5_29.

[11] Nima Keivan and Gabe Sibley. Online slam with any-time self-calibration and automatic change detection. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[12] Chang Boon Low and Danwei Wang. Gps-based path following control for a car-like wheeled mobile robot with skidding and slipping. *IEEE Transactions on control systems technology*, 16(2):340–347, 2008.

[13] Frank Moosmann and Christoph Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398. IEEE, 2011.

[14] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[15] Bryan Nagy and Alonzo Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. *Field and Service Robots*, 11, 2001.

[16] Pongsathorn Raksincharoensak, Masao Nagai, and Motoki Shino. Lane keeping control strategy with direct yaw moment control input by considering dynamics of electric vehicle. *Vehicle System Dynamics*, 44(sup1):192–201, 2006.

[17] Masayasu Shimakage, Shigeki Satoh, Kenya Uenuma, and Hiroshi Mouri. Design of lane-keeping control with steering torque input. *JSAE review*, 23(3):317–323, 2002.