Model predictive trajectory tracking control of automated guided vehicle in complex environments

Chen, Chun-Lin; Li, Juncheng; Li, Maoxun; Xie, Lihua

2018

Chen, C.-L., Li, J., Li, M., & Xie, L. (2018). Model predictive trajectory tracking control of automated guided vehicle in complex environments. 2018 IEEE 14th International Conference on Control and Automation (ICCA), 405-410. doi:10.1109/ICCA.2018.8444247

<https://hdl.handle.net/10356/143716>

<https://doi.org/10.1109/ICCA.2018.8444247>

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at <https://doi.org/10.1109/ICCA.2018.8444247>

Downloaded on 26 Apr 2023 18:13:31 SGT

Model Predictive Trajectory Tracking Control of Automated Guided Vehicle in Complex Environments*

Chun-Lin Chen¹, Juncheng Li¹, Maoxun Li¹ and Lihua Xie¹

Abstract—Autonomous navigation in a real-world industrial environment is in many ways a challenging task. One of the key challenges is rapid collision-free planning and execution of trajectories to reach any target position and orientation with high accuracy, taking into account the limitations of imperfectness of the vehicle. The model prediction-based motion planners have been successfully used in recent years to generate feasible motions for imperfect vehicles. This paper develops and implements a Model Predictive Control (MPC)-based trajectory controller for path tracking problem in narrow corridors. To evaluate the performance of the proposed method, we designed comparative simulations and experiments. We confirm that the proposed MPC-based controller can track the trajectory precisely and smoothly in specific complex environments. In addition, the proposed methodology can also be a suitable solution to other way-point tracking situations for an industrial mobile robot.

I. INTRODUCTION

Automatic Guided Vehicles (AGVs) have been applied in manufacturing for goods and product transport. AGVs are often used to transport materials between production facilities and warehouses. More and more researchers and manufacturers began to focus on AGV due to its many advantages such as reducing the damage to transported goods and the number of workers, controlling logistics in real time, improving productivity, quality, working conditions, safety and product management [1][2].

The AGV localization is to estimate robot pose relative to its environment, including location and orientation. The robot localization is an important research topic part in the fields of mobile robots (e.g. [3]). There are many different localization problems for mobile robots [4]. Among them, the position tracking is the simplest one and has received the most attention in the literature [5].

This paper applies a kinematic model for solving the problem of smooth and continuous movement of AGV in a factory environment [6]. Collision avoidance is a basic problem when operating AGV robots. To this end, extensive research has been conducted in the past [7]. Without aiming at completeness, we will outline some main approaches briefly in the following, with the goal of motivating our proposed approaches.

Motion control problem has been addressed by various control approaches, including sliding-mode control [8], PID control [9] and so on. However, nonlinear kinematic model

can not be handled in a systematic way and the input is not optimal in classical control methods. In order to overcome these limitations and achieve more accurate tracking performance, we choose to design a motion controller under the model predictive control (MPC) framework. MPC is an advanced method and has been widely used to solve the path tracking problem of autonomous vehicles (e.g. [10][11]). In MPC, future states can be predicted according to current states and open-loop dynamics of the robot. To minimize a chosen cost function, the optimal sequence of inputs is computed under known constraints on states and inputs. The robot applies the first element input of the derived control input sequence with a given horizon sequences and then repeats the process at next step. In the case of AGV trajectory tracking problem, the model used in state prediction is nonlinear, so nonlinear MPC (NPMC) can be applied. The control objective is to find the optimized longitudinal speed and steering angle for the optimal predicted trajectory of the robot. The most recent research direction is to use the MPC strategy to guarantee the constrained flocking of the second-order swarm or multi-agent systems [12].

This paper is organized as follows. Section II describes the dynamical kinematic model of mobile AGV robot and provides the problem formulation. Section III proposes a model predictive controller for robot to realize collision avoidance. Section IV shows some simulation and experimental results that validate the proposed controller. Section V concludes this paper and provides directions for future research.

II. PRELIMINARIES

A. Kinematic model

In this study, a kinematic model is considered for a wheeled robot [6]. The configuration of the differential wheeled AGV platform is shown in Figure 1 is of the width R_w and length R_l . The kinematic equations for coordinate and direction of the robot are given by:

$$\begin{cases} \dot{x}_c = v \cdot \cos \theta, \\ \dot{y}_c = v \cdot \sin \theta, \\ \dot{\theta} = \omega, \end{cases} \quad (1)$$

where (x_c, y_c) is the center point of AGV, θ is the rotation angle, ω is the angular velocity, v is the linear velocity. Denote (x_h, y_h) as the head point of the AGV.

According to the configuration in Figure 1, the motion equations of v and ω corresponding to the angular velocities of the left and right wheels, ω_L and ω_R , can be obtained as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} r_w/2 & r_w/2 \\ -r_w/2l_w & r_w/2l_w \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}, \quad (2)$$

*This work was supported by National Research Foundation, Nanyang Technological University and Delta Electronics Enterprise in Singapore

¹Chun-Lin Chen, Juncheng Li, Maoxun Li and Lihua Xie are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. e-mail: elhxie@ntu.edu.sg

Fig. 1: Configuration of the differential wheeled AGV platform

where r_w is the radius of wheel. Substituting (2) into (1), we arrive at the following equivalent model of AGV.

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} (r_w/2) \cdot \cos \theta & (r_w/2) \cdot \cos \theta \\ (r_w/2) \cdot \sin \theta & (r_w/2) \cdot \sin \theta \\ -r_w/2l_w & r_w/2l_w \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}, \quad (3)$$

Considering a sample interval T_s and a zero-order hold, (3) changes in discrete-time to:

$$\begin{cases} x_c(k+1) = x_c(k) + \cos[H_w\omega_L(k) + H_w\omega_R(k)] \cdot T_s \\ y_c(k+1) = y_c(k) + \sin[H_w\omega_L(k) + H_w\omega_R(k)] \cdot T_s \\ \theta(k+1) = \theta(k) + [(-H_w l_w^{-1})\omega_L(k) + H_w l_w^{-1}\omega_R(k)] \cdot T_s \end{cases} \quad (4)$$

where $H_w = r_w/2$ is half the radius of wheel. Finally, we can obtain a discrete-time kinematic model of AGV.

B. Problem formulation

In order to save space, many production environments are very crowded with machines, which leaves narrow space for mobile robot operation in a factory environment. Note that, passing through a narrow corridor is a big challenge for AGVs. The width of the narrow corridor is c_w with conditions $R_w < c_w < R_l$. We are particularly interested in the control problem in which we wish to track a time-varying reference trajectory $r(k)$, called the trajectory tracking control problem. We need to design and plan a collision-free reference trajectory $r(k)$ such that the robot trajectory $z(k)$ is as close to the reference trajectory $r(k)$ as possible. See Figure 2. Let

$$\tilde{z}(k) = r(k) - z(k), \quad (5)$$

where the robot movement trajectory $z(k)$ can be represented as $[x_c, y_c]$; the reference trajectory $r(k)$ can be represented as $[x_r, y_r]$. The control objective is to minimize $\tilde{z}(k)$ in certain sense. In the next section, we shall explain how to design a

Fig. 2: Trajectory control of mobile robot in narrow corridor

Fig. 3: Flowchart of the basic particle filter SLAM

trajectory controller to output appropriate control commands in terms of the linear velocity v and the angular velocity ω .

III. METHODOLOGY

This section is divided into two parts: first is the trajectory generation based on an initial environmental navigation map and, second is to use the navigation map to localize the mobile robot.

A. Mapping and localization

First, we want to make sure that the mobile robot can move freely between factory passages. We use the Rao-Blackwellized particle filter to achieve Simultaneous Localization And Mapping (SLAM) capabilities to derive an environmental navigation map [13][14]. The core point of the Rao-Blackwellized particle filter is to estimate the joint posterior $p(x_{1:t}, m | z_{1:t}, u_{1:t-1})$ about the map m and the

poses $x_{1:t} = x_1, \dots, x_t$ of AGV. Based on observations $z_{1:t} = z_1, \dots, z_t$ (e.g. LIDAR scan data L_s) and the robot movements $u_{1:t-1} = u_1, \dots, u_{t-1}$ obtained by AGV, this estimation is performed. In the Rao-Blackwellized particle filter for AGV map creation and localization, we can express the following factorization.

$$p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) = p(m | z_{1:t}, u_{1:t-1}) \cdot p(x_{1:t} | z_{1:t}, u_{1:t-1}) \quad (6)$$

At first, (6) allows us to estimate the robot trajectory and then calculate the map based on the given trajectory. This approach provides an efficient computation, because the maps heavily rely on the attitude estimate of the robot. This algorithm is often called Rao-Blackwellization. The detailed flowchart of particle filter localization is shown in Figure 3.

Second, we shall create a scenes navigation map and then use a Monte Carlo Localization (MCL) methodology [15][16]. MCL is a recursive Bayes filter to estimate the posterior distribution of robot poses conditioned on sensor data. The core point underlying all these works is to represent the posterior belief $Bel(l)$ by a set of N weighted, random samples or *particles* $S = \{s_i | i = 1 \dots N\}$. A sample set constitutes a discrete approximation of a probability distribution. Samples in MCL are of the type

$$\langle (x, y, \theta), p \rangle, \quad (7)$$

where x and y are the robot's coordinates in a world-centered orthogonal coordinate system, and θ is the robot's orientation, and $p[p_1, p_2, \dots, p_N] \geq 0$ is a numerical weighting factor, which is analogous to a discrete probability. For consistency, we assume $\sum_{n=1}^N p_n = 1$. Similar to with the general Markov localization approach, MCL proceeds in two phases: 1) Robot motion, 2) Sensor readings [17]. In the first phase, the robot predicts its new position by applying a simulated motion to each particle, based on a given start command. In the second phase, as the robot senses its environment, it updates its particles to more accurately reflect its position. According to the probability density of each particle, the robot calculates its position in a particle state. Based on the sensory information, the robot can correct its position.

B. Generate a collision-free reference

By now, we have obtained a navigation map from the SLAM algorithm and used the offline map for path planning. This map can be described as an image consisting of white, black and gray pixels as free space, obstacles and unknown areas respectively. In order to present a map for analysis, the offline map is subdivided into cells on a 2D grid, which is called the grid map. A grid node P is located at the center point of each cell. The obstacles on the grid map get inflated in all directions for collision avoidance due to the size of the robot and the measurement inaccuracies. The thickness of the inflated layer is a little bigger than the inscribed radius of the mobile robot. The map with inflated layer is used as a reference map for the path planning algorithm. The flowchart of the collision-free path generation is shown in Figure 4.

Fig. 4: Flowchart of the collision-free path generation

The Dijkstra optimization algorithm is usually utilized to plan the shortest path between two specified grid nodes in a graph [18]. For our robot, this algorithm is used to create a shortest path from the current node of the robot to the goal node. The path cost evaluation function $F(r)$ is given by

$$F(r) = \sum_{k=1}^{m-1} C(P_k, P_{k+1}), \quad (8)$$

where r is the collision-free reference trajectory, P_k and P_{k+1} are a pair of adjacent grid nodes, $C(P_k, P_{k+1})$ is the distance function, m is the number of grid nodes between the current and goal nodes.

The grid nodes occupied by the obstacles are assigned very large $C(P_k, P_{k+1})$ values for a high cost on the evaluation function. The goal is to select an optimal collision-free reference trajectory r to minimize $F(r)$.

C. Time interval method (TIM)

We will explain the operation of the trajectory tracking in this section. As shown in Figure 5, the reference trajectory r , the reference trajectory points number $N_r \in (k = 1, 2, \dots, N_r)$, the total reference path length D_r , the robot maximum linear velocity v_{max} and angular velocity $\omega_{max} = v_{max}/r_w$ are known. The difference of the reference trajectory $\Delta d(k)$ at each moment can be defined as:

$$\Delta d(k) = r(k) - r(k-1) = \sqrt{(x_r(k) - x_r(k-1))^2 + (y_r(k) - y_r(k-1))^2} \quad (9)$$

The difference of switching time at each moment can be defined as:

$$\Delta t(k) = t(k) - t(k-1) = \frac{\Delta d(k)}{v_{max}} \quad (10)$$

We want to obtain the reference position $r(k)$ corresponding to the time-varying motion of the robot during its dynamic movement. The absolute time at the moment j to switch the reference point k can be expressed as:

$$T(j) = \sum_{k=1}^j \Delta t(k) \quad k, j \in N_r \quad (11)$$

Therefore, the time intervals corresponding to the respective switching reference points are shown in Table I.

TABLE I: Switching time for each reference point

Number k	1	2	...	N_r
Switching time	$T(1)$	$T(2)$	...	$T(N_r)$
Reference point	$r(1)$	$r(2)$	...	$r(N_r)$
Relative movement	$\Delta d(1)$	$\Delta d(2)$	...	$\Delta d(N_r)$

Fig. 5: Block diagram of the trajectory control

D. Trajectory control using model predictive controller

In this subsection, we design a trajectory tracking controller based on MPC. Let $z_i = [x_i, y_i, \theta_i]^T$ be the state of the vehicle, $i=1,2, \dots, H$ with H the prediction horizon, and $u_i = [v_i, \omega_i]^T$ be the control input. Typically, a standard MPC is formulated as:

$$\min_{u_0, \dots, u_H} \sum_{i=1}^H h(z_i, u_i) \quad (12a)$$

$$\text{s.t. } z_{i+1} = f(z_i, u_i), \quad (12b)$$

$$z_0 = z(k), \quad (12c)$$

$$z_i \in \mathbb{Z}, u_i \in \mathbb{U}, \quad (12d)$$

where $h(z_i, u_i)$ is the objective function and $\mathbb{Z}, \mathbb{U}$ are the sets of feasible states and inputs. (12b) represents the kinematic model which is used for state prediction, (12c) is the initial state and (12d) describes the state and input constraints.

The objective function is defined to achieve high tracking accuracy as well as moving smoothness. Therefore, it should penalize the deviations from the reference trajectory and the input change rate magnitude. We formulate the cost function as

$$\min_{u_0, \dots, u_{H-1}} \sum_{i=1}^H \tilde{z}_i^T Q \tilde{z}_i + \sum_{i=0}^{H-1} \tilde{u}_i^T R \tilde{u}_i + \Delta u_i^T J \Delta u_i \quad (13)$$

where

$$\tilde{z}_i = r(k+i) - z_i, \quad (14a)$$

$$\tilde{u}_i = \begin{bmatrix} v_{ref} \\ 0 \end{bmatrix} - u_i, \quad (14b)$$

$$\Delta u_i = u_i - u_{i-1}, \quad (14c)$$

and $Q \in \mathbb{R}^{3 \times 3}$, $R \in \mathbb{R}^{2 \times 2}$, $J \in \mathbb{R}^{2 \times 2}$ are the state penalization, input penalization and differential input penalization matrices, they are diagonal and positive definite.

In this paper we assume that a local optimal solution to the Problem(12) can be obtained in real time. At time step k , by using solvers for nonlinear optimization, the optimal state and input sequence $z^*(k)$ and $u^*(k)$ can be obtained.

$$\begin{aligned} z^*(k) &= [z_1^*, z_2^*, \dots, z_N^*], \\ u^*(k) &= [u_0^*, u_1^*, \dots, u_{N-1}^*] \end{aligned} \quad (15)$$

(a) Simulation on stage

(b) Display on rviz

Fig. 6: Mobile robot simulation environment on LinuxOS system

TABLE II: Parameters of simulation

	R_l	R_w	l_h	r_w	l_w	c_w
Values (m)	1.025	0.55	0.59	0.075	0.2585	1.0

From the optimization result $u^*(k)$, the input of the vehicle $u(k)$ can be decided as:

$$u(k) = u_0^*. \quad (16)$$

IV. SIMULATION AND EXPERIMENTAL RESULTS

A. Simulation settings and results

We are using the open source *Robot Operating System* (ROS) to build our simulation environment on LinuxOS system and *stage* package to complete simulation of mobile robot trajectory control. The *stage* simulator is a 2D mobile robot simulator that provides a mobile robot with the capability of customizing a can virtual environment. This software can build similar factory environment even simulation out appearance, movement, attitude and sensor model as shown in Figure 6a. However, in order to obtain real-time mobile robot poses and sensor information, we use *rviz* package to present the results seen in Figure 6b.

The simulation environment has the same modules and interfaces as the experiments, so we can use the simulation environment to tune the controllers and get sufficiently accurate results. We consider the situation which is shown in Figure 2 with environment settings and AGV model parameters as Table II. Figure 7 shows the path planning and trajectory tracking results. The MPC's prediction horizon H set to be 20 and the parameters in controller are all fine tuned. In this case, our proposed controller has the ability to pass through narrow corridors successfully.

Additionally, to evaluate the performance of the proposed control method, a PID controller will be compared to the MPC. Figure 8 shows trajectory deviation along the whole path using the two control methods. As listed in Table III, we can see that our proposed MPC outperforms the other controller.

Fig. 7: Trajectory tracking using MPC (simulation)

Fig. 9: MPC trajectory tracking experiment

Fig. 8: Trajectory deviation (simulation)

TABLE III: Tracking error and velocity comparison (simulation)

	E_{RMES}	E_{MAX}	V_{MEAN}	V_{MAX}
PID	0.0022	0.0117	0.24	0.41
MPC	0.0007	0.0031	0.37	0.49

B. Experimental results

The proposed trajectory tracking control is implemented on a robotic vehicle for experiments. The robot is outfitted with a LIDAR for localization and a mini-computer *UP board*[®] onboard. The processor is fast enough to operate at a control frequency of 20Hz. As the simulation environment is similar enough to the experiment, we choose to implement the control in the same way with simulation. Also, the parameters in the control law are set to the value which has been tuned in the simulation environment.

Figure 9 shows the experimental results. As expected, the robot can track the trajectory very well. As described before, to achieve moving smoothness of the robot, the input change rate should not be too large. Figure 10 shows the input of the robot along the whole trajectory. Both linear and angular velocity are changing at a reasonable rate, indicating that the moving is smooth enough. To evaluate the tracking

Fig. 10: MPC Trajectory tracking input

TABLE IV: Trajectory tracking error and velocity comparison

	E_{RMES}	E_{MAX}	V_{MEAN}	V_{MAX}
PID	0.0012	0.0054	0.25	0.40
MPC	0.0005	0.0025	0.38	0.52

performance of our proposed MPC controller, we use the comparison shown in Figure 11. As listed in Table IV, we can see similar results data with the simulation cases. In both the simulation and experiment, the MPC controller can achieve smaller trajectory tracking error and faster moving speed.

V. CONCLUSIONS

This paper has presented the design and implementation of a MPC-based trajectory controller for path tracking problem in narrow corridors. By tuning the MPC parameters, both desired trajectory tracking accuracy and moving smoothness can be achieved. A simulation was implemented in *ROS* on *LinuxOS* system. The simulation environment has the same settings as the real environment, so tuning the controller can

Fig. 11: Trajectory deviation

be fully done in the simulation. We conducted two comparative studies: using classic PID controller and MPC. The simulation results showed that our proposed control approach performs well in narrow corridor situation. Experiments under the same condition were done after the simulation. The experimental results show good agreement with the simulation ones. From overall results, we confirmed that the proposed MPC controller is a good solution to achieve proper performance and design flexibility in trajectory tracking problems. Although our design and experiments were analyzed under the narrow corridor situation, the proposed methodology can be applied to various way-point trajectory tracking conditions.

ACKNOWLEDGMENT

The research is partially supported by the *Delta - NTU Corporate Lab* through the *NRF corporate lab@university* scheme. The authors would like to thank Mr. *Yongjun Wee* and Dr. *Jeffrey Soon* for providing us use cases and many discussions during the course of this work.

REFERENCES

- [1] M. Eghtesad and D. S. Neculescu, "Experimental study of the dynamic based feedback linearization of an autonomous wheeled ground vehicle," *Robotics and autonomous systems*, vol. 47, no. 1, pp. 47–63, 2004.
- [2] H. Maaref and C. Barret, "Sensor-based fuzzy navigation of an autonomous mobile robot in an indoor environment," *Control Engineering Practice*, vol. 8, no. 7, pp. 757–768, 2000.
- [3] W. D. Rencken, "Concurrent localisation and map building for mobile robots using ultrasonic sensors," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1993, pp. 2192–2197.
- [4] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.
- [5] B. Schiele and J. L. Crowley, "A comparison of position estimation techniques using occupancy grids," *Robotics and autonomous systems*, vol. 12, no. 3-4, pp. 163–171, 1994.
- [6] Y.-H. Chang, C.-L. Chen, W.-S. Chan, and H.-W. Lin, "Type-2 fuzzy formation control for collision-free multi-robot systems," *International Journal of Fuzzy Systems*, vol. 15, no. 4, 2013.

- [7] Y.-H. Chang, C.-L. Chen, W.-S. Chan, H.-W. Lin, and C.-W. Chang, "Fuzzy formation control and collision avoidance for multiagent systems," *Mathematical Problems in Engineering*, vol. 2013, 2013.
- [8] A. Filipescu, V. Minzu, B. Dumitrascu, A. Filipescu, and E. Minca, "Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots," in *Information and Automation (ICIA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 27–32.
- [9] A. Pandey, S. Jha, and D. Chakravarty, "Modeling and control of an autonomous three wheeled mobile robot with front steer," in *Robotic Computing (IRC), IEEE International Conference on*. IEEE, 2017, pp. 136–142.
- [10] P. F. Lima, M. Nilsson, M. Trincavelli, J. Mårtensson, and B. Wahlberg, "Spatial model predictive control for smooth and accurate steering of an autonomous truck," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 4, pp. 238–250, 2017.
- [11] W. Y. Choi, C. M. Kang, S. H. Lee, and C. C. Chung, "Waypoint tracking predictive control with vehicle speed variation," in *2017 11th Asian Control Conference (ASCC)*, Dec 2017, pp. 61–66.
- [12] B. Zhu, L. Xie, D. Han, X. Meng, and R. Teo, "A survey on recent progress in control of swarm systems," *Science China Information Sciences*, vol. 60, no. 7, p. 070201, 2017.
- [13] K. P. Murphy, "Bayesian map learning in dynamic environments," in *Advances in Neural Information Processing Systems*, 2000, pp. 1015–1021.
- [14] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.
- [15] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.
- [16] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.
- [17] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization," *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.
- [18] M. Noto and H. Sato, "A method for the shortest path search by extended dijkstra algorithm," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 2316–2320.