2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Path Following for an Omnidirectional Mobile Robot Based on Model Predictive Control

Kiattisin Kanjanawanishkul and Andreas Zell

Abstract— **In this paper, the path following problem of an omnidirectional mobile robot has been studied. Given the error dynamic model derived from the robot state vector and the path state vector, model predictive control (MPC) is employed to design the control law, which can deal explicitly with the rate of progression of a virtual vehicle to be followed along the path. The distinct advantage over other control strategies is that input and system constraints are able to be handled straightforwardly in the optimization problem so that the robot can travel safely with a high velocity. Unlike nonholonomic mobile robots, omnidirectional mobile robots, which we focus on in this paper, have simultaneously and independently controlled rotational and translational motion capabilities. Then, our purposed MPC controller was validated by experiments with a real omnidirectional mobile robot.**

I. INTRODUCTION

Omnidirectional mobile robots are becoming increasingly popular in mobile robot applications, since they have some distinguishing advantages over nonholonomic mobile robots. They have simultaneously and independently controlled rotational and translational motion capabilities, which means that they can move at each instant in any direction without reorientation [1]. The middle size league of the annual RoboCup competition is an example of a highly dynamic environment where omnidirectional mobile robots have been exploited highly successfully (see RoboCup Official Site: http://www.robocup.org).

Fundamental problems of motion control of autonomous vehicles can be roughly classified in three groups [2], namely point stabilization, trajectory tracking, and path following control. In this paper we focus on the path following problem. An intuitive explanation given in the literature is that a path following controller should look at (i) the distance from the vehicle to the reference path and (ii) the angle between the vehicle's velocity vector and the tangent to the path, and then reduce both to zero, without any consideration in temporal specifications. Pioneering work in this area can be found in [3] as well as [4] and the references therein.

Let $p_d(s) \in \mathbb{R}^2$ be a desired geometric path parameterized by the curvilinear abscissa $s \in \mathbb{R}$, instead of time, which is normally used in a trajectory tracking problem. The underlying assumption in path following control is that the vehicle's forward velocity u_R tracks a desired velocity profile, while the controller determines the vehicle's moving direction to drive it to the path. Typically this controller eliminates the aggressiveness of the tracking controller by forcing convergence to the desired path in a smooth way [5].

The path following problem has been well studied and many solutions have been proposed and applied in a wide range of applications. Samson [6] described a path following problem for a car pulling several trailers. In [7], Altafini addressed a path following controller for an n trailer vehicle. Furthermore, path following controllers for aircraft and marine vehicles have been reported in [5] and [8], respectively. In contrast to trajectory tracking, in path following we have the freedom to select a temporal specification for $s(t)$. In particular, we can consider $s(t)$ as an additional control input. In [2], [9], [10], [11], the rate of progression ($\dot{s}$) of a virtual vehicle has been controlled explicitly. Stringent initial condition constraints that are present in a number of path following control strategies have been overcome, as stated in [2].

Given the state errors between the robot kinematics and the virtual vehicle kinematics, linearized around operating points, we propose another possibility to obtain the rate of progression of a virtual vehicle via model predictive control (MPC). With this linearized time-varying system, the optimization problem can be transformed to a quadratic programming (QP) problem. Since it becomes a convex problem, solving the QP problem leads to global optimal solutions. After solving the QP problem at each time instant, the optimal rate of progression can be obtained. We also integrate input constraints into the QP problem so that we can steer the robot safely with a high forward velocity. Furthermore, the forward velocity monitoring is used to generate the velocity profile, which the robot's forward velocity will track. The velocity profile is shaped to comply with the robot and path constraints and is employed along the predictive horizon of the MPC controller. This lets the MPC controller utilize future information to generate optimal control inputs at each time instant.

The rest of the paper is structured as follows: in Section II, an omnidirectional mobile robot is modeled and its constraints are determined. Section III introduces path following control and problem formulation. Then our control algorithm is developed in Section IV. Section V shows the experimental results. Finally, our conclusions and future work are given in Section VI.

K. Kanjanawanishkul and A. Zell are with Wilhelm-Schickard-Institute, Department of Computer Architecture, University of Tübingen, Sand 1, 72076 Tübingen, Germany {kiattisin.kanjanawanishkul, andreas.zell}@uni-tuebingen.de

II. Kinematic Model of an Omnidirectional Mobile Robot

In Fig. 1, there are two coordinate frames used in the modeling: the body frame (X_m, Y_m) and the world frame (X_w, Y_w). The body frame is fixed at the moving robot with the origin at the center of the robot, whereas the world frame is fixed at the ground. Angles θ_m and θ_t denote the robot orientation and the robot moving direction in the world frame, respectively. Each wheel has the same distance L_w to the robot's center of mass R. δ refers to the wheel orientation in the body frame. The kinematic model of an omnidirectional mobile robot is given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0,$$
$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} \cos\theta_m & -\sin\theta_m & 0 \\ \sin\theta_m & \cos\theta_m & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_m \\ v_m \\ \omega_m \end{bmatrix} , \quad (1)$$

where $\mathbf{x}(t) = [x_m, y_m, \theta_m]^T$ is the state vector in the world frame and $\mathbf{u}(t) = [u_m, v_m, \omega_m]^T$ is the vector of robot velocities observed in the body frame. u_m and v_m are the robot translational velocities and ω_m is the robot rotational velocity. When the wheel velocities are considered, the lower level kinematic model with respect to the robot coordinate can be described by

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta & L_w \\ -\cos\delta & \sin\delta & L_w \\ 0 & -1 & L_w \end{bmatrix} \begin{bmatrix} u_m \\ v_m \\ \omega_m \end{bmatrix} , \quad (2)$$

where $\dot{\mathbf{q}}(t) = [\dot{q}_1,\ \dot{q}_2,\ \dot{q}_3]^T$ is the vector of wheel velocities, which is equal to the wheel's radius multiplied by the wheel's angular velocity. Substituting (2) into (1) results in

$$\dot{\mathbf{x}}(t) = P(\theta_m)\dot{\mathbf{q}}(t) \quad (3)$$

with

$$P(\theta_m) = \frac{2}{3} \begin{bmatrix} \cos(\theta_m + \delta) & -\cos(\theta_m - \delta) & \sin\theta_m \\ \sin(\theta_m + \delta) & -\sin(\theta_m - \delta) & -\cos\theta_m \\ \frac{1}{2L_w} & \frac{1}{2L_w} & \frac{1}{2L_w} \end{bmatrix} .$$

Since translation and rotation are coupled via θ_m in (3), to decouple the θ_m-equation from those of the translational ones, we employ the similar method, as proposed in [12]. For a given θ_m, the linear transformation $P(\theta_m)$ maps the cube $\mathcal{Q}(t) = \{\dot{\mathbf{q}}(t) \mid |\dot{q}_i(t)| \le \dot{q}_{i,max}\}$ into the tilted cuboid $P(\theta_m)\mathcal{Q}(t)$. The matrix $P(\theta_m)$ can be decomposed as a product of a rotation and a θ_m-independent linear transformation

$$P(\theta_m) = R_z(\theta_m)P(0) \ , \quad (4)$$

where

$$R_z(\theta_m) = \begin{bmatrix} \cos\theta_m & -\sin\theta_m & 0 \\ \sin\theta_m & \cos\theta_m & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$
$$P(0) = \begin{bmatrix} \sqrt{3}/3 & -\sqrt{3}/3 & 0 \\ 1/3 & 1/3 & -2/3 \\ 1/(3L_w) & 1/(3L_w) & 1/(3L_w) \end{bmatrix} ,$$

and $\delta = \frac{\pi}{6} rad$.

The linear transformation $P(0)$ maps cube $\mathcal{Q}(t)$ into the titled cuboid $P(0)\mathcal{Q}(t)$ with a diagonal $\dot{q}_3 \le 9.7436 rad/s$, calculated by using $L_w = 0.195m$ and $|\dot{q}_i(t)| \le 1.9 m/s$, along the ω_m axis. The transformation $R_z(\theta_m)$ then rotates this cuboid about the ω_m axis. The problem is to find the solid of revolution that is the intersection of all possible rotations $R(\theta)_z P(0)\mathcal{Q}(t)$ of the cuboid. This solid of revolution is characterized by (see [12] for details)

$$\dot{x}_m^2(t) + \dot{y}_m^2(t) \le r^2(\omega_m) \ , \quad (5)$$

where the radius is

$$r(\omega_m) = \frac{9.7436 - |\omega_m|}{5.1283} . \quad (6)$$

Equation (5) shows the relationship between translational and rotational velocities. For example, in our experiments we set $\sqrt{\dot{x}_m^2(t) + \dot{y}_m^2(t)} \le u_{max} = 1.315 m/s$ as a maximum forward velocity and then the maximum allowable rotational velocity becomes $|\omega_m| \le 3 rad/s$. When the desired rotational velocity is larger than this maximum allowable value, the translational velocity needs to be decreased in order that the desired rotational velocity can be achieved. Using this relationship, we ensure that the maximum wheel velocities will not be violated. Bounds on translational and rotational velocities can either be self-imposed due to desired behavior and safety concerns or be physical due to the actual limitations such as currents and voltages in the motors [15].

Since translation and rotation of omnidirectional mobile robots can be controlled separately, we can rewrite (1) by decoupling translation and rotation:

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_t \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} u_R \cos\theta_t \\ u_R \sin\theta_t \\ \Phi u_R \\ \omega_m \end{bmatrix} , \quad (7)$$

where Φ is the curvature and the robot translational velocities can be determined by

$$\begin{bmatrix} u_m \\ v_m \end{bmatrix} = \begin{bmatrix} u_R \cos(\theta_t - \theta_m) \\ u_R \sin(\theta_t - \theta_m) \end{bmatrix} . \quad (8)$$

To control motions of omnidirectional mobile robots, in [13], Liu et al. implemented trajectory linearization control (TLC), which is based on linearization along the desired trajectory and inversion of the dynamics. Watanabe [14] introduced the PID control, self-tuning PID control, and

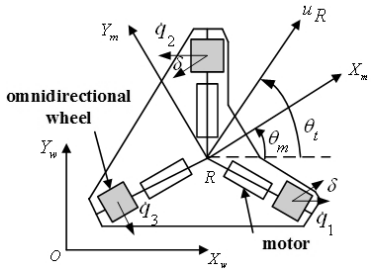


Fig. 1. Coordinate frames of an omnidirectional mobile robot.

fuzzy control of omni-directional mobile robots. However the controllers developed in [14] is based on a linear control method while the robot model is nonlinear. In this paper, we focus on the path following problem and solve this problem with linear MPC including consideration in robot constraints, as described in Section IV.

III. Path Following Control and Problem Formulation

The kinematic model of an omnidirectional mobile robot can be formulated with respect to a Serret-Frenet frame moving along the reference path. This frame plays the role of the body frame of a virtual vehicle that must be followed by the real vehicle. In our path following problem shown in Fig. 2, we let the forward velocity u_R track a desired velocity profile, while the rate of progression of a virtual vehicle $\dot{s}$ converges to u_R. The error state vector $\mathbf{x}_e$ between the robot state vector $\mathbf{x}$ and a virtual vehicle's state vector $\mathbf{x}_r = [x_r, y_r, \theta_r, \theta_b]^T$, where θ_b is the desired orientation and θ_r is the tangent angle to the path, can be expressed in the frame of the path coordinate as follows

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 & 0 \\ -\sin\theta_r & \cos\theta_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m - x_r \\ y_m - y_r \\ \theta_t - \theta_r \\ \theta_m - \theta_b \end{bmatrix} . \tag{9}$$

Using (7) and (8), the error state dynamic model chosen in a rotated coordinate frame (9) is derived as follows

$$\begin{aligned} \dot{x}_e &= y_e \kappa_r(s)\dot{s} - \dot{s} + u_R \cos\alpha_e \\ \dot{y}_e &= -x_e \kappa_r(s)\dot{s} + u_R \sin\alpha_e \\ \dot{\alpha}_e &= \Phi u_R - \kappa_r(s)\dot{s} \\ \dot{\theta}_e &= \omega_m - \omega_b \end{aligned} , \tag{10}$$

where $\omega_b = \dot{\theta}_b$, $\dot{\theta}_r = \kappa_r(s)\dot{s}$, and $\kappa_r(s)$ is the path curvature at path length s.

Linearizing the error dynamics (10) around the reference path $\mathbf{x}_d = [x_d, y_d, \theta_d]^T$, we get the following linear model:

$$\begin{aligned} \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \\ \dot{\theta}_e \end{bmatrix} &= \begin{bmatrix} 0 & \kappa_d u_R & 0 & 0 \\ -\kappa_d u_R & 0 & u_R & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} \\ &+ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \end{aligned} , \tag{11}$$

where

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} -\dot{s} + u_R \cos\alpha_e \\ \Phi u_R - \kappa_r(s)\dot{s} \\ \omega_m - \omega_b \end{bmatrix} . \tag{12}$$

In [1], it is shown that the posture kinematic model of omnidirectional robots is completely controllable.

Besides converging the vehicle to a desired path, assigning a velocity profile to the path can be a task, in which the forward velocity is used as an extra degree of freedom. For example, the forward velocity should be decreased as the vehicle rotates around a sharp corner. This allows handling of constraints by scaling the forward velocity as proposed in [15]. In [10], path following has been combined with obstacle avoidance. The forward velocity has to be controlled when an obstacle is detected. In this paper, the velocity profile is generated by considering path and robot constraints and then our proposed controller will steer the robot to follow that path with a predefined velocity profile. In general, the paths are chosen to be C^2 continuous, so they have continuous curvature and no cusps. The curvature κ of a continuous curvature path is upper bounded by κ_{max}. That is, the steering radius $\rho \geq \rho_{min} = 1/\kappa_{max}$. Additionally, there is an upper bound on the curvature derivative, $\dot{\kappa}$, since a robot must reorient its moving direction with a finite steering velocity. We consider the following velocity constraints:

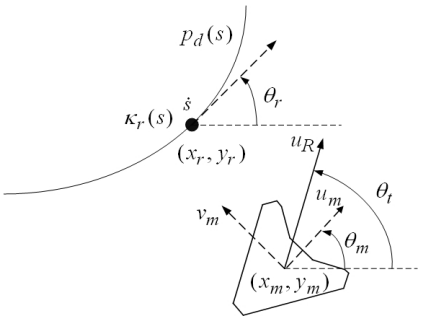


Fig. 2. Illustration of the path following problem.

A. Maximum Velocity Constraints

Maximum velocity constraints are $u \leq u_{max}$. These can be used to enforce maximum speed constraints.

B. Desired Forward and Rotational Velocities

Given the desired forward velocity u_R and the desired rotational velocity ω_b, we can get the forward velocity by using

$$u_d = \min\left(u_R, (6)\right) , \tag{13}$$

where ω_m in (6) is replaced by ω_b.

C. Velocity Magnitude Constraints

To prevent the robot slipping off the path, we have the constraint

$$-\sqrt{\tfrac{\mu g}{|\kappa|}} \leq u \leq \sqrt{\tfrac{\mu g}{|\kappa|}} , \tag{14}$$

where μ is the friction coefficient, g is the acceleration due to gravity, and κ is the curvature.

Thus, the velocity constraints are

$$0 \leq u \leq \min\left(u_{max}, u_d, \sqrt{\tfrac{\mu g}{|\kappa|}}\right) . \tag{15}$$

The strategy for generating the velocity profiles can be integrated into the online path planner. The velocity can be monitored within some lookahead distance. The slower the desired forward velocity u_R, the shorter the lookahead distance can be considered. The number of steps of a velocity profile must be at least equal to the predictive horizon N

of MPC, detailed in the next section. Moreover, we neglect actuator dynamics by assuming that it is much faster than the desired closed loop system dynamics.

IV. Controller Design

Model predictive control (MPC) has become an increasingly popular control technique used in industry. The controller is based on a finite-horizon continuous time minimization of predicted tracking errors with constraints on the control inputs and the state variables. At each sampling time, the model predictive controller generates an optimal control sequence by solving an optimization problem. The first element of this sequence is applied to the system. The problem is solved again at the next sampling time using the updated process measurements and a shifted horizon. A more comprehensive explanation can be found in [16] and [17].

Most model predictive controllers use a linear model of mobile robot kinematics to predict future system outputs. In [18], [19], a model-predictive control based on a linear, time-varying description of the system is used for trajectory tracking control. Generalized predictive control (GPC) is used to solve the path following problem in [20]. The nonlinear predictive controller scheme for a trajectory tracking problem is proposed in [21], [22]. Recently, Falcone et.al. [23] implemented an MPC-based approach for active steering control design. They presented two approaches, i.e., MPC using a nonlinear vehicle model and MPC based on successive online linearization of the vehicle model. The differences of this paper from other work are that (i) this paper deals with an omnidirectional mobile robot and path following control, and (ii) we propose another possibility to obtain the optimal rate of progression of a virtual vehicle to be followed along the path. The MPC law, based on a linearized error dynamics model, is computationally effective and can be easily used in fast real time implementations.

Equation (11) can be given in the state-space form $\dot{\mathbf{x}}_e = A_c\mathbf{x}_e + B_c\mathbf{u}_e$. To design the MPC controller for path following, the linearized system (11) will be written in a discrete state space system as

$$\mathbf{x}_e(k+1) = A\mathbf{x}_e(k) + B\mathbf{u}_e(k), \tag{16}$$

where $A \in \mathbb{R}^n \times \mathbb{R}^n$, n is the number of the state variables and $B \in \mathbb{R}^n \times \mathbb{R}^m$, m is the number of input variables. The discrete matrices A and B can be obtained as follows:

$$A = I + A_cT_s, \quad B = B_cT_s, \tag{17}$$

where T_s is a sampling time.

By creating a state space model (16) of a system, it is possible to use MPC to control it. The approach is to find the control-variable values that minimize the quadratic objective function by solving a quadratic program (QP). The quadratic objective function with a predictive horizon N is given by

$$\begin{aligned} J(k) = &\sum_{j=1}^{N}\{\mathbf{x}_e^T(k+j|k)Q\mathbf{x}_e(k+j|k) \\ &+ \mathbf{u}_e^T(k+j-1|k)R\mathbf{u}_e(k+j-1|k)\}, \end{aligned} \tag{18}$$

where $Q \in \mathbb{R}^n \times \mathbb{R}^n$ and $R \in \mathbb{R}^m \times \mathbb{R}^m$ are the weighting matrices, with $Q \geq 0$ and $R \geq 0$. The double subscript notation $(k+j|k)$ denotes the prediction made at time k of a value at time $k+j$.

First, the following matrices are defined in order to be able to write the problem in a form, which a QP solver can solve. By introducing matrices such that all $\mathbf{u}_e(\cdot)$ and $\mathbf{x}_e(\cdot)$ are conveniently stored, a more compact expression is achieved. Defining the prediction-error vector

$$X(k) = [\mathbf{x}_e^T(k+1|k), \mathbf{x}_e^T(k+2|k), \ldots, \mathbf{x}_e^T(k+N|k)]^T \ ,$$

where $X \in \mathbb{R}^{n\cdot N}$ and the control error vector

$$U(k) = [\mathbf{u}_e^T(k|k), \mathbf{u}_e^T(k+1|k), \ldots, \mathbf{u}_e^T(k+N-1|k)]^T \ ,$$

where $U \in \mathbb{R}^{m\cdot N}$.

It can be shown that

$$X(k) = G(k)\mathbf{x}_e(k|k) + S(k)U(k) \ , \tag{19}$$

where $G(k) \in \mathbb{R}^{n\cdot N} \times \mathbb{R}^{n\cdot N}$ and $S(k) \in \mathbb{R}^{n\cdot N} \times \mathbb{R}^{m\cdot N}$ are defined as follows

$$G(k) = [\alpha(k+1,k), \alpha(k+2,k), \cdots, \alpha(k+N,k)]^T \ ,$$

and

$$S(k) = \begin{bmatrix} \beta_{11}(k) & 0 & \cdots & 0 \\ \beta_{21}(k) & \beta_{22}(k) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1}(k) & \beta_{N2}(k) & \cdots & \beta_{NN}(k) \end{bmatrix} ,$$

with β_{ij} and $\alpha(k,j)$ defined as:

$$\beta_{ij} = \alpha(k+i, k+j)B(k+j-1)$$

$$\alpha(k_1,k_0) = \begin{cases} \mathbf{I} & \text{if } k_1 = k_0 \\ \prod_{i=0}^{N-1} A(k+i) & \text{if } k_1 > k_0 \end{cases} .$$

After defining $\bar{Q} \in \mathbb{R}^{n\cdot N} \times \mathbb{R}^{n\cdot N}$ and $\bar{R} \in \mathbb{R}^{m\cdot N} \times \mathbb{R}^{m\cdot N}$ as block diagonal matrices containing Q and R repeated N times, the objective function of the QP problem can be rewritten

$$\begin{aligned} J(k) =& \sum_{j=1}^{N}\{\mathbf{x}_e^T(k+j|k)Q\mathbf{x}_e(k+j|k) \\ &+ \mathbf{u}_e^T(k+j-1|k)R\mathbf{u}_e(k+j-1|k)\} \\ =& X^T\bar{Q}X + U^T\bar{R}U \\ =& (G\mathbf{x}_e(k|k) + SU)^T\bar{Q}(G\mathbf{x}_e(k|k) + SU) + U^T\bar{R}U \end{aligned} \tag{20}$$

After some algebraic manipulations, we can rewrite the objective function (18) in a standard quadratic form:

$$\bar{J}(k) = \tfrac{1}{2}U^T(k)H(k)U(k) + \mathbf{f}^T(k)U(k) \ , \tag{21}$$

with

$$\begin{aligned} H(k) &= 2(S(k)^T\bar{Q}S(k) + \bar{R}) \\ \mathbf{f}(k) &= 2S(k)^T\bar{Q}G(k)\mathbf{x}_e(k|k) \end{aligned} \ ,$$

where $H \in \mathbb{R}^{m\cdot N} \times \mathbb{R}^{m\cdot N}$ and $\mathbf{f} \in \mathbb{R}^{m\cdot N}$. Since all constants, which do not contain the variable U, do not affect the

optimum, they have been excluded in (21). The matrix $H(k)$ is a Hessian matrix and it is always positive definite. It describes the quadratic part of the objective function, and the vector $\mathbf{f}(k)$ describes the linear part.

To handle the input constraints, we consider the existence of bounds in the amplitude of the control variables:

$$\mathbf{u}_{min} \leq \mathbf{u}(k+j|k) \leq \mathbf{u}_{max} \ , \tag{22}$$

where $j \in [0,\ N-1]$, and $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ stand for the lower and upper bounds, respectively. In our path following problem, we consider the following constraints:

$$\begin{array}{ccccc} 0 & \leq & \dot{s} & \leq & u_{max} \\ -\omega_c - \omega_b & \leq & \omega_m & \leq & \omega_c - \omega_b \end{array} , \tag{23}$$

with $\omega_c = \min(\omega_l, \omega_{m,max})$, where ω_l is calculated by using (6), in which $r(\omega_m)$ is replaced by u_R.

Thus, this standard expression is used in QP problems and the optimization problem to be solved at each sampling time is stated as follows

$$U^\star = \arg \min_{\mathbf{u}_e} \bar{J}(k) \tag{24}$$

subject to

$$\mathbf{u}_{min} \leq \mathbf{u}(k+j|k) \leq \mathbf{u}_{max} \ .$$

After the QP problem at time t_k is solved, an optimal control sequence is generated. The rate of progression of a virtual vehicle $\dot{s}$, the steering angle θ_t, and the rotational velocity ω_m can be calculated from the first element of this sequence and then applied to the system.

V. Experimental Results

We validated our proposed control algorithm with a real omnidirectional mobile robot, shown in Fig. 3. It is equipped with a Pentium-M 2 GHz on-board PC with 1 GB RAM and its wheels are driven by three 60W Maxon DC motors. It has an omnidirectional camera as sole sensor, which is used for self localization. The self localization algorithm [24] applied for the RoboCup field has been employed in our experiments. This self-localization algorithm is based on probabilistic Monte-Carlo localization (MCL). The eight-shaped reference path and the user-defined parameters are given as follows

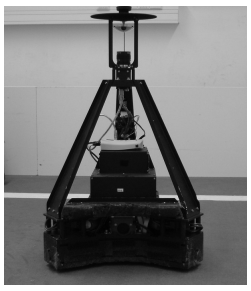


Fig. 3. The omnidirectional mobile robot used in the experiments of path following control.

$$x_d(t) = 1.8\sin(t),\ y_d(t) = 1.2\sin(2t),$$
$$\theta_b = 0,\ \omega_b = 0,\ \mu = 0.18,\ g = 9.81 m/s^2,$$
$$u_R = 1m/s,\ u_{max} = 1.315m/s,\ \omega_{m,max} = 5rad/s.$$

The reference path was numerically parameterized by the curvilinear abscissa s. All tunable parameters used in our experiments are listed:

$Q = \text{diagonal}(300, 300, 7, 70)$,
$R = \text{diagonal}(1, 0.001, 3)$,
N = 3, T_s = $0.05s$.

In our experiments, the package *OOQP* [25], written by E. M. Gertz and S. J. Wright, has been used to solve the QP problem and PID controllers have been implemented for motor velocity control. The experimental results are shown in Fig. 4. As can be seen from the experimental results, the real-time implementation of our control law can be realized. The forward velocity u_R is decreased in order to preserve the curvature radius when the robot made sharp turns, while the velocity commands did not exceed the velocity constraints, as expected.

By using the MPC law, it is well known that the shorter the predictive horizon N, the less costly the solution of the online optimization problem. Thus it is desirable from a computational point of view to implement MPC schemes using short horizons. However, this may lead to poor performance. In case of fast dynamic systems like our system, the predictive horizon must be chosen in such a way that the computing time is smaller than the sampling period. From our experiments, shown in Table I, we have found that $N = 3-5$ is a reasonable choice for the predictive horizon.

It is worth noting that if the robot is far from the reference path, a so-called landing curve can be applied to drive the robot to the path.

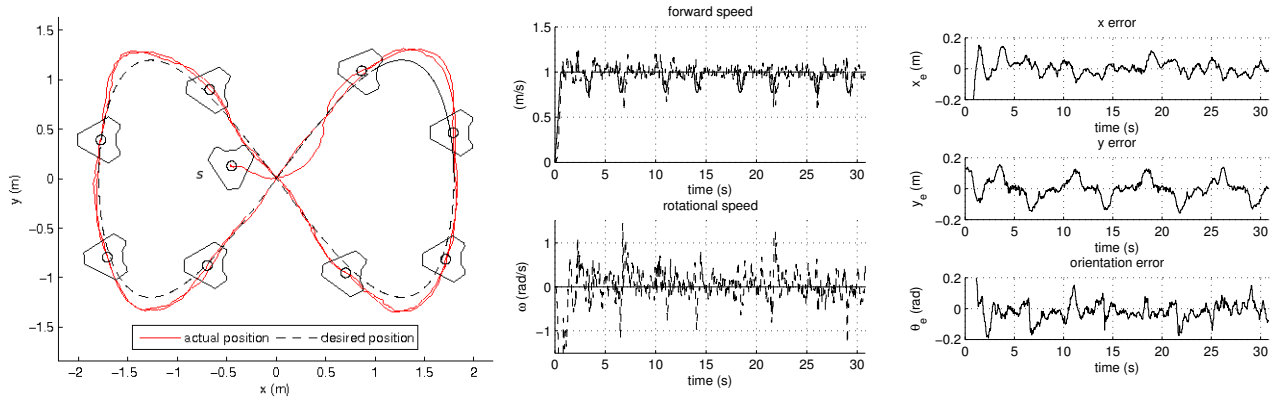
VI. Conclusions and Future Work

In this paper, we present linear MPC to solve the path following problem of an omnidirectional mobile robot with consideration of robot and path constraints. The high forward velocity is kept constant if control input commands do not violate velocity constraints.

The major advantages of employing MPC to design a path following controller in this paper are that (i) it can handle system and input constraints straightforwardly while generating an optimal control sequence, (ii) it can provide the optimal rate of progression of a virtual vehicle in order to overcome stringent initial condition constraints as stated in [2], and (iii) it utilizes velocity information in the future, based on a predefined velocity profile at each time instant. As well known, the basic limitation of the linearization around

TABLE I
Prediction horizon influence on computing time.

Prediction Horizon	Computing Time (ms)
1	0.54
3	1.12
5	2.19
10	9.74
15	29.62



(a) (b) (c)

Fig. 4. The experimental results by using the linear MPC law: (a) the superimposed snapshots, (b) the forward velocity and the rotational velocity, and (c) the pose errors with respect to the path coordinate. S in (a) denotes the initial position of the robot.

paths is that stability is only guaranteed in a neighborhood of the selected operating points. Although a nonlinear version of MPC is now available, the major concerns are a control stability problem and high computational time.

In the future, our robot should work in the real environment. Thus, we will integrate obstacle avoidance into our system and implement an online path planner by using local measurements from exteroceptive sensing, e.g., ultrasonic, laser scanner, vision.

REFERENCES

[1] G. Campion, G. Bastin, and B. D'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 1, pp. 47-62, Feb. 1996.

[2] D. Soeanto, L. Lapierre, and A. Pascoal, "Adaptive non-singular path-following, control of dynamic wheeled robots," in *Proc. of International Conference on Advanced Robotics*, Coimbra, Portugal, June 30 - July 3, 2003, pp. 1387-1392.

[3] A. Micaelli and C. Samson, "Trajectory-tracking for unicycle-type and two-steering-wheels mobile robots," *Technical Report No. 2097*, INRIA, Sophia-Antipolis, Nov. 1993.

[4] C. Canudas de Wit, C. Samson, H. Khennouf, and O. J. Sørdalen, "Nonlinear control design for mobile robots," in *Recent trends in mobile robots* (eds Y. F. Zheng), World Scientific Publishing, vol. 11, pp. 121-156, Apr. 1993.

[5] S. A. Al-Hiddabi and N. H. McClamroch, "Tracking and maneuver regulation control for nonlinear non-minimum phase systems: application to flight control," *IEEE Trans. on Control Systems Technology*, vol. 10, no. 6, pp. 780-792, 2002.

[6] C. Samson, "Control of chained systems: Application to path-following and time-varying point stabilization of mobile robots," *IEEE Trans. on Automatic Control*, vol. 40, no. 1, pp. 64-77, Jan. 1995.

[7] C. Altafini, "Following a path of varying curvature as an output regulation problem," *IEEE Trans. on Automatic Control*, vol. 47, no. 9, pp. 1551-1556, Sep. 2002.

[8] P. Encarnação and A. Pascoal, "3D path following for autonomous underwater vehicle," in *Proc. of 39th IEEE Conference on Decision and Control CDC2000*, Sydney, Australia, Dec. 2000, pp. 2977-2982.

[9] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *IEEE Trans. on Automatic Control*, vol. 46, no. 11, pp. 1777-1782, Nov. 2001.

[10] L. Lapierre, R. Zapata, and P. Lepinay, "Combined path-following and obstacle avoidance control of a wheeled robot," *International Journal of Robotics Research*, vol. 26, no. 4, pp. 361-376, 2007.

[11] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle-like vehicles via Lyapunov techniques," *IEEE Robotics and Automation Magazine*, pp. 27-35, March 1995.

[12] T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly, "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle," *Robotics and Autonomous Systems*, vol. 46, pp. 47-64, 2004.

[13] Y. Liu, X. Wu , J. Jim Zhu, and J. Lew, "Omni-directional mobile robot controller design by trajectory linearization," in *Proc. of American Control Conference*, Denver, Colorado, Jun. 2003, pp. 3423-3428.

[14] K. Watanabe, "Control of omnidirectional mobile robot," in *Proc. of 2nd Int. Conf. on Knowledge-Based Intelligent Electronic Systems*, Adelaide, Australia, Apr. 1998, pp. 51-60.

[15] M. Bak, N. K. Poulsen, and O. Ravn, "Path following mobile robot in the presence of velocity constraints," *Technical Report*, Informatics and Mathematical Modeling, Technical University of Denmark, DTU, 2001.

[16] W. H. Kwon and S. Han, "Receding Horizon Control: Model Predictive Control for State Models," Springer-Verlag, London, 2005.

[17] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, pp. 789-814, 2000.

[18] W. F. Lages and J. A. V. Alves, "Real-time control of a mobile robot using linearized model predictive control," in *Proc. of 4th IFAC Symposium on Mechatronic Systems*, Heidelberg, Germany, Sep. 2006, pp. 968-973.

[19] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460-469, 2007.

[20] A. Ollero and O. Amidi, "Predictive path tracking of mobile robos. Application to the CMU Navlab," in *Proc. of International Conference on Advanced Robotics*, Pisa, Italy, Jun. 1991, pp. 1081-1086.

[21] D. Gu and H. Hu, "Receding horizon tracking control of wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 743-749, Jul. 2006.

[22] R. Hedjar, R. Toumi, P. Boucher, and D. Dumur, "Finite horizon nonlinear predictive control by the Taylor approximation: application to robot tracking trajectory," *Int. Journal of Applied Mathematics and Computer Science*, vol. 15, no. 4, pp. 527-540, 2005.

[23] P Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. on Control Systems Technology*, vol. 15, no. 3, pp. 566-580, May 2007.

[24] P. Heinemann, J. Haase, and A. Zell, "A combined Monte-Carlo localization and tracking algorithm for RoboCup," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, Beijing, China, Oct. 2006, pp. 1535-1540.

[25] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. on Mathematical Software*, vol. 29, no. 1, pp. 58-81, 2003.