Path Following with an Optimal Forward Velocity for a Mobile Robot

Kiattisin Kanjanawanishkul, Marius Hofmeister, and Andreas Zell

Department of Computer Architecture, University of Tübingen, Sand 1, 72076 Tübingen, Germany
(e-mail: {kiattisin.kanjanawanishkul, marius.hofmeister, andreas.zell}@uni-tuebingen.de)

Abstract: In this paper, we present a novel solution for a path following problem in partially-known static environments. Given linearized error dynamic equations, model predictive control (MPC) is employed to produce a sequence of angular velocities. Since the forward velocity of the robot has to be adapted to environmental constraints and robot dynamics while the robot is following a path, we propose an optimal solution to generate the velocity profile. Furthermore, we integrate an obstacle-avoidance behavior using local sensor information with a path-following behavior based on global knowledge. To achieve this, we introduce new waypoints in order to move the robot away from obstacles while the robot still keeps following the desired path. Extensive simulations and experiments with a physical unicycle mobile robot have been conducted to illustrate the effectiveness of our path following control framework.

Keywords: Robot control, autonomous mobile robots, obstacle avoidance, path following, model predictive control.

1. INTRODUCTION

Fundamental problems of motion control of autonomous mobile robots can be roughly classified into three groups (Morin and Samson, 2008), namely point stabilization, trajectory tracking, and path following. In this paper, we focus on the path following problem. Pioneering work in this area can be found in (Micaelli and Samson, 1993). The underlying assumption of this problem is that the robot's forward velocity tracks a desired velocity profile, while the controller determines the robot's moving direction to drive it to the path without any consideration in temporal specifications. Typically this controller eliminates the aggressiveness of the tracking controller by forcing convergence to the path in a smooth way (Al-Hiddabi and McClamroch, 2002).

The path following problem has been well studied and many solutions have been proposed and applied in a wide range of applications. Samson (Samson, 1995) described a path following problem for a car pulling several trailers. Altafini (Altafini, 2002) addressed a path following controller for an n trailer vehicle. Path following controllers for aircraft and marine vehicles were reported in (Al-Hiddabi and McClamroch, 2002) and (Encarnação and Pascoal, 2000), respectively.

In this work, we wish to achieve three objectives: static obstacle avoidance, path following, and forward velocity selection. An illustrative example for these objectives is car driving. A driver controls a car to follow a road using a steering maneuver. The driver may decelerate the car if he or she sees obstacles blocking the road or is making a sharp turning, or is driving on an icy road. Besides all these situations, safety concerns and human comfort also influence the desired forward velocity.

We separate our problem into three parts as shown in Fig. 1. The MPC block produces a sequence of angular velocities. Its detail is given in Section 2. The velocity selection block, described in Section 3, adapts the forward velocity of the robot to environmental constraints and robot dynamics. The reference path generator block, explained in detail in Section 4, provides the desired reference for path following control and replans the path if the robot moves close to obstacles. Simulation and experimental results are shown in Section 5. Finally, our conclusions and future work are given in Section 6.

2. THE PATH FOLLOWING PROBLEM

In this section, the model predictive control (MPC) framework is used to generate a sequence of angular velocities. MPC has become an increasingly popular control technique used in industry (Kwon and Han, 2005; Mayne et al., 2000). It 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.

Most model predictive controllers use a linear model of mobile robot kinematics to predict future system outputs. In (Lages and Alves, 2006; Klančar and Škrjanc, 2007), model-predictive control based on a linear, time-varying description of the system was used for trajectory tracking control. Generalized predictive control was used to solve path following control in (Ollero and Amidi, 1991). A nonlinear predictive controller for a trajectory tracking problem was proposed in (Gu and Hu, 2006). An MPC-based approach for active steering control was implemented in (Falcone et al., 2007). The differences of this paper from other work are that (i) this paper deals with a linearized model for path following control, (ii) we take into account

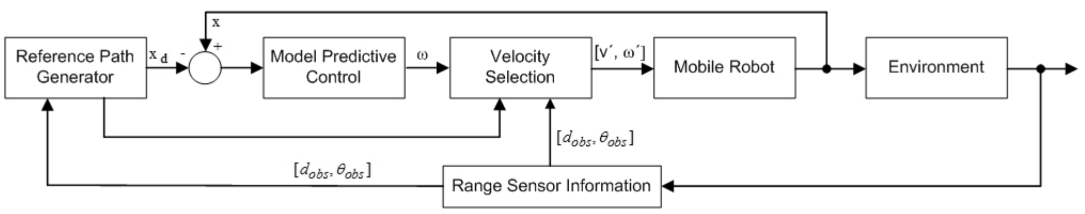


Fig. 1. The block diagram describes our solution for the path following control problem.

obstacle avoidance, and (iii) the forward velocity selection is introduced.

In general, a linear MPC framework is computationally effective and can be easily used in fast real time implementations. To apply this framework, we first formulate our problem. The kinematics of a mobile robot, depicted in Fig. 2 together with a spatial path Γ to be followed, can be described by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{1}$$

where $\mathbf{x}(t) = [x, y, \theta]^T$ denotes the state vector in the world frame. v and ω are the linear and angular velocities, respectively. We wish to find control law ω such that the robot converges to the path while v tracks velocity profiles. The path error with respect to the path frame is given by

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix} \tag{2}$$

where the state vector of the reference point, $[x_d, y_d, \theta_d]^T$, is computed by using a numerical projection from the robot's current state onto the path.

Then, the linearized version of the error dynamics $\mathbf{x}_e = [y_e,\ \theta_e]^T$ (the lateral and angular deviations, respectively) results in

$$\begin{aligned} \dot{y}_e &= v_d\theta_e \\ \dot{\theta}_e &= \omega - \omega_d \end{aligned} \tag{3}$$

where v_d and ω_d are the desired linear and angular velocities, respectively. We can transform the optimization problem of MPC to a quadratic programming (QP) problem by using this linearized model. Since it becomes a convex problem, solving the QP problem leads to global optimal solutions. Equation (3) 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 (3) will be written in a discrete state space system as

$$\mathbf{x}_e(k+1) = A\mathbf{x}_e(k) + B\mathbf{u}_e(k) \tag{4}$$

where $A \in \mathbb{R}^n \times \mathbb{R}^n$, n is the number of 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:

$$\begin{aligned} A &= I + A_cT_s \\ B &= B_cT_s \end{aligned} \tag{5}$$

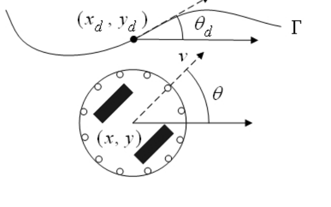


Fig. 2. A graphical representation of a mobile robot and a path. A small circle ○ denotes a distance sensor.

where T_s is a sampling time.

Given a state space model (4) of a system, it is possible to use MPC to control it. To achieve this, we have to minimize a quadratic objective function by solving a quadratic program in order to obtain control-variable values. The quadratic objective function with a prediction horizon N is given by

$$\begin{aligned} J(k) = \sum_{j=1}^{N} \{ &\mathbf{x}_e^T(k+j|k)Q\mathbf{x}_e(k+j|k) \\ &+ \mathbf{u}_e^T(k+j-1|k)R\mathbf{u}_e(k+j-1|k)\} \end{aligned} \tag{6}$$

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 $k+j$. Furthermore, the matrix Q is adapted to lateral deviations as follows

$$Q(1,1) = \frac{c_1}{1 + c_2|y_e|} \tag{7}$$

where c_1 and c_2 are positive. When the robot is far away from the path, the weighting gain for lateral deviations will become smaller, resulting in more importance in angular errors. When the robot moves closer to the path, $Q(1,1)$ becomes larger, leading to more importance in lateral errors, as seen in Fig. 3.

After some algebraic manipulations, we can rewrite the objective function (6) in a standard quadratic form:

$$\bar{J}(k) = \frac{1}{2}U^T(k)H(k)U(k) + \mathbf{f}^T(k)U(k) \tag{8}$$

where $U(k) = [\mathbf{u}_e^T(k|k), \mathbf{u}_e^T(k+1|k), \ldots, \mathbf{u}_e^T(k+N-1|k)]^T$. The matrix $H(k) \in \mathbb{R}^{m\cdot N} \times \mathbb{R}^{m\cdot N}$ is a Hessian matrix and it is always positive definite. It describes the quadratic part of the objective function and the vector $\mathbf{f}(k) \in \mathbb{R}^{m\cdot N}$ describes the linear part. The unconstrained control law can be obtained by minimizing the objective function with respect to U as follows

$$\frac{\partial\bar{J}(k)}{\partial U(k)} = H(k)U(k) + \mathbf{f}(k) \tag{9}$$

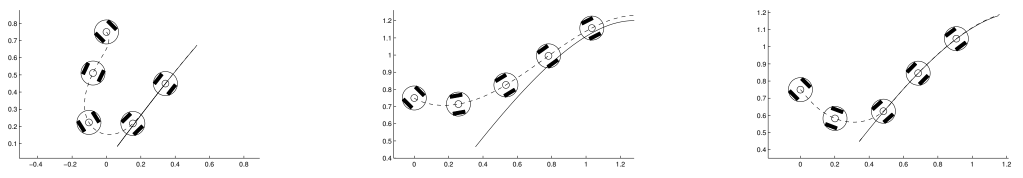
and the control vector becomes

$$U(k) = H^{-1}(-\mathbf{f}(k))\,. \tag{10}$$

This control vector contains a sequence of angular velocities.

3. FORWARD VELOCITY SELECTION

Besides steering the robot to the desired path, assigning a velocity profile to the robot can be an additional task, in which the forward velocity is used as an extra degree of freedom. For example, in (Bak et al., 2001), the forward velocity decreases as the robot rotates around a sharp corner by scaling the forward velocity. In (Lapierre et al., 2007), the forward velocity is controlled when an obstacle is detected. In this paper, the velocity profile is shaped to comply with environmental constraints and robot dynamics along some lookahead distance corresponding to the N-step prediction horizon of the MPC framework. We consider bounds on the forward velocity as follows:



(a) (b) (c)

Fig. 3. The path following results by using the MPC framework, where (a) $Q = \mathrm{diag}(100, 1)$, (b) $Q = \mathrm{diag}(10, 1)$, and (c) the adapted matrix Q with $c_1 = 1000$ and $c_2 = 100$.

- *Environmental constraints*: These constraints can be self-imposed due to desired behaviors based on high-level tasks, safety concerns, obstacle avoiding, sharp turning, slippery floors, or human comfort. In this work, the desired forward velocity, which satisfies high-level tasks, safety, and human comfort, is given by an operator. To take into account obstacle avoidance, we determine the velocity as follows

$$v = \begin{cases} v_{\min} & \text{if } d_{obs} < d_{\min} \\ \dfrac{v_d - v_{\min}}{d_{\max} - d_{\min}} (d_{obs} - d_{\min}) + v_{\min} & \\ \qquad \text{if } d_{\min} \leq d_{obs} \leq d_{\max} \text{ and } |\theta_{obs}| < \theta_{obs,\max} & \\ v_d & \text{otherwise} \end{cases} \tag{11}$$

where d_{obs} is the distance between the robot and the obstacle, and θ_{obs} is the angle of the obstacle with respect to the robot frame. $d_{\min}$ is the minimum safe distance. $d_{\max}$ and $\theta_{obs,\max}$ are the distance and the angle, which the obstacle can influence robot motions, respectively. Equation (11) means that when the robot moves close to the obstacle, the robot will slow down and then the orientation command will push it away from the obstacle. Moreover, to prevent slippage, we impose the following constraints

$$\begin{aligned} a_{T,\min} \leq a_T = \frac{dv}{dt} \leq a_{T,\max} \\ a_{L,\min} \leq a_L = \kappa v^2 \leq a_{L,\max} \end{aligned} \tag{12}$$

where a_T and a_L are the longitudinal and lateral accelerations, respectively. κ is the curvature and the dynamic limitations are the maximum (minimum) longitudinal acceleration $a_{T,\max}$ $(a_{T,\min})$ and the maximum (minimum) lateral acceleration $a_{L,\max}$ $(a_{L,\min})$. For the constant linear velocity, the smaller the curvature is, the smaller the lateral acceleration is induced. This constraint forces the robot to slow down in sharp turns. Note that the acceleration and deceleration constraints may not be the same and a_T is also the physical constraint of the robot.
- *Robot dynamics*: The physical constraints are due to the actual limitations such as currents and voltages in the motors, resulting in maximum wheel velocities and accelerations. The limitations are derived from the wheel velocities: $v = (v_r + v_l)/2, \ \omega = (v_r - v_l)/(2b)$, where v_r and v_l are the linear velocities of the right and left wheels, respectively, and $2b$ is the distance between the left wheel and the right wheel. The maximum rotation and translation can be decoupled to obtain $v_{\max}$ and $\omega_{\max}$. The authors in (Bak et al., 2001; Indiveri et al., 2007) proposed an approach to maximize feasible linear velocity, resulting in less conservative bounds. However, we focus on the different objective, i.e., we wish the robot to converge to the reference path by reducing the lateral and angular deviations to zero while traveling at a desired forward velocity. This desired forward velocity must satisfy both the environmental constraints and the robot dynamics. Thus, the maximum translation can be completely decoupled from the maximum rotation.

Given the environmental constraints and the robot dynamics, we utilize one of the advantages from the MPC framework, i.e., the future information of the reference path can be incorporated into the MPC framework. The N-step prediction horizon can be seen as a lookahead distance in the path following problem. In general, the slower a desired forward velocity is, the shorter a lookahead distance can be determined. After the velocity selection module receives sensor information and a sequence of control inputs from the MPC framework, the following procedure is employed to generate the velocity profile with respect to the constraints mentioned above.

<u>Procedure 1</u>: Velocity selection

(1) Calculate the actual orientation commands ω along the N-step prediction horizon by using (3)

(2) Perform the velocity scaling in order to preserve the curvature radius

```
**if** $|\omega(k+j|k)| > \omega_{\max}$ **then**
    $v_p(k+j|k) = v(k+j|k)\omega_{\max}/|\omega(k+j|k)|$
    $\omega_p(k+j|k) = \mathrm{sign}(\omega(k+j|k))\omega_{\max}$
**else**
    $v_p(k+j|k) = v(k+j|k)$
    $\omega_p(k+j|k) = \omega(k+j|k)$
**end if**
```

where $j \in [0, N-1]$

(3) With the profile given in Step (2), the objective function to be minimized is

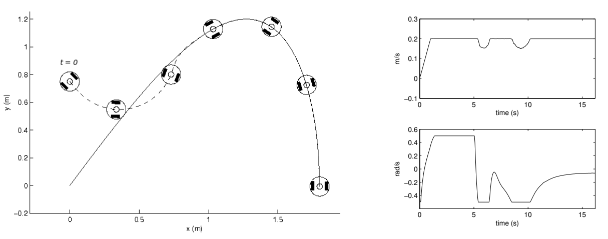
$$\min_{a,\epsilon} \left(\sum_{j=1}^{N} \gamma_1 \|v_p(k+j|k) - v_m(k+j|k)\|^2 + \gamma_2 \|\epsilon\| \right)$$

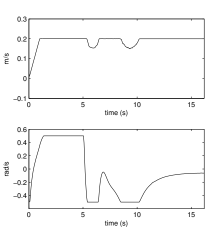
$$\begin{aligned} \text{s.t. } & 0 \leq \epsilon \\ & |\kappa v_m^2(k+1|k)| - a_{L,\max} \leq \epsilon \\ & a(k+j|k) = \frac{v_m(k+j+1|k) - v_m(k+j|k)}{T_s} \\ & |a(k+j|k)| \leq a_{T,\max} \end{aligned}$$

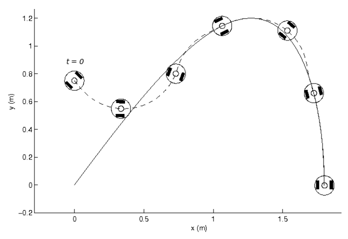
where γ_1 and γ_2 are positive weights

(4) $v_p(k+1|k)$ and $\omega_p(k+1|k)$ are sent to control robot motions

The hard constraint of the lateral acceleration may lead to the infeasible solution. To soften the constraint, the slack variable ϵ is introduced. However, we impose this softening constraint only on the first element of the velocity profile in order to decrease computational time. As seen in Fig. 4, the optimal


(a)


(b)


(c)

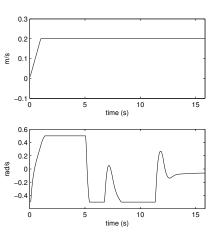

(d)

Fig. 4. The path following results with $a_{T,\max} = 0.2$ m/s^2, $a_{L,\max} = 0.2$ m/s^2, $\omega_{\max} = 0.5$ rad/s, $v_0 = 0.2$ m/s, and $N = 50$ steps (the lookahead distance = 0.5 m and $T_s = 0.05$ s): (a) using the optimal velocity strategy, (b) the velocity profiles corresponding to (a), (c) using the constant forward velocity, and (d) the velocity profiles corresponding to (c).

velocity selection module can reduce the lateral deviations because the robot moves at a lower velocity (see Fig. 4(b)) while it is making a sharp turning. Obviously, the lateral deviations increase, if the constant velocity is chosen, as shown in Fig. 4(c).

4. THE PATH REPLANNING STRATEGY

Typically, the desired reference is generated by a planning algorithm based on a map of the environment and this reference is assumed to be collision-free. During the movement in partially structured environments, an obstacle can suddenly appear on the robot's path, which had not been present in the planning phase. To avoid the obstacle, a sensory system should detect the obstacle, measure its distance and orientation for replanning the robot's path. In (Lapierre et al., 2007), an obstacle avoidance algorithm based on the use of a continuous Deformable Virtual Zone (DVZ) is combined with a path following controller. In (Maček et al., 2009), global path planning, path following, and a collision avoidance scheme are integrated in a unified framework, namely the Traversability-anchored Dynamic Path Following (TADPF). In this paper, we combine a path-following behavior using global knowledge with an obstacle-avoidance behavior based on local sensor information. Furthermore, we keep tracking the curvilinear abscissa $s(t) \in \mathbb{R}$, which is used to parameterize the reference path. This path's parameter s can be used to detect whether the robot is stuck in a loop and to lead the robot back to the path when the obstacle-avoidance behavior becomes inactive.

The obstacle-avoidance behavior becomes active as the robot moves closer to obstacles than $d_{\max}$ and one of these obstacles blocks the robot. The path replanning module then locally generates new waypoints to deform the reference path in order to bring the robot away from the obstacle. These new waypoints are tangential to the edges of the obstacle with an offset d_r. Procedure 2 below describes our solution in detail.

Procedure 2: Path replanning strategy

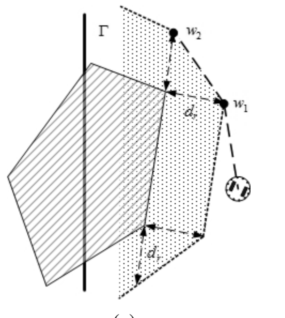
(1) Receive sensor information, estimate the edges of all detected obstacles, and evaluate the number of the detected obstacles. If the distance between two detected points is larger than the distance d_s, a new obstacle is introduced. If it is smaller, it means the robot cannot get through.
(2) Create convex hulls around the obstacle edges with an offset distance d_r. If two polygons overlap, they are combined into one convex polygon (see Fig. 5(b) as an example).
(3) Evaluate the relevant obstacles, which influence robot motions and then make a decision based on the generated polygons around the detected obstacles and the reference path to make a right turn or a left turn in order to avoid the obstacles.
(4) From Step (3), keep following on that side along the edges of the convex polygon toward the first visible vertex (see Fig. 5(a) and Fig. 5(b) as examples) until that obstacle no longer influences robot motions.

Note that the distances d_s and d_r are dependent on the robot size and the sensor accuracy.

Fig. 6 illustrates the results from applying Procedure 2. "$\times$" represents a detected point by a range sensor and dashed lines indicate the convex polygons around the detected points. However, the robot may make a wrong decision in some situations because of insufficient sensor information. For example, the robot may move far away from the reference path because it keeps following the boundary of the obstacle. In this case, global knowledge may be needed to solve this problem.

5. SIMULATION AND EXPERIMENTAL RESULTS

Our path following control framework has been evaluated in both computer simulations written in MATLAB and a physical unicycle mobile robot. To show the effectiveness of our approach, the simulations were first conducted using an arbitrarily constructed environment including obstacles. We assume that


(a)

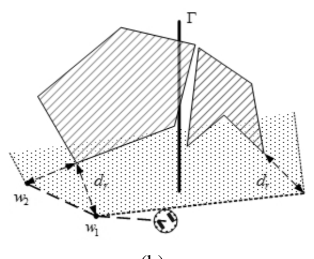
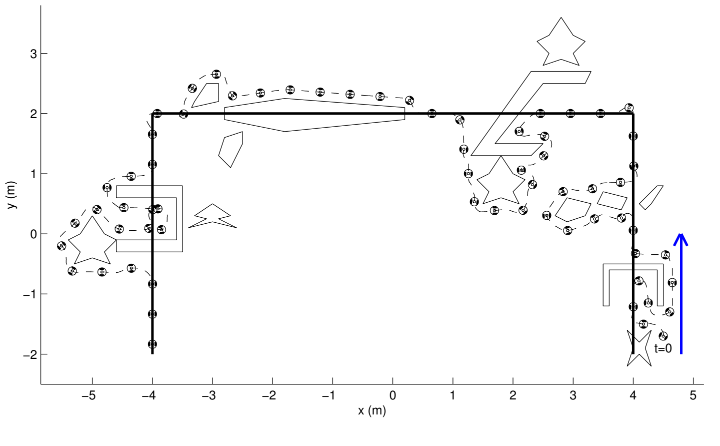

(b)

Fig. 5. Two cases are shown as examples. A robot detects an obstacle and convex hulls are then created around the edges of the detected obstacle. The robot will follow the first visible vertex along the edges of the convex polygon. (a) A convex polygon is constructed around an obstacle, and (b) a convex polygon covers two detected obstacles because their convex polygons overlap. Dashed lines represent the replanned path and dotted lines are part of the convex polygons around the detected obstacles.


(a)

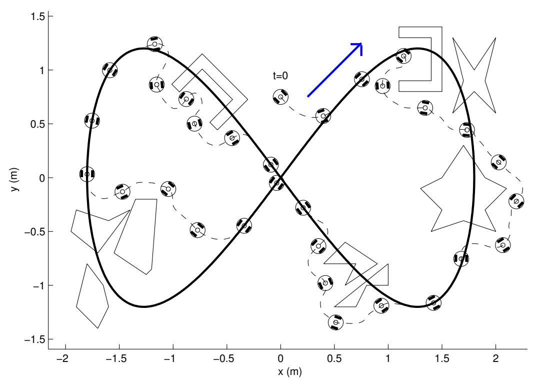
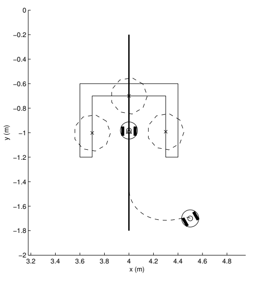

(b)

Fig. 7. The simulation results by using our path following control framework: (a) The robot starting at $[4.5, -1.7, 2\pi/3]^T$ is required to follow a reference path represented by thick lines, and (b) the robot starting at $[0, 0.75, -\pi/4]^T$ is required to follow an eight-shaped curve. The polygons are obstacles while the small circles are snapshots of robot location every 2.5 s. The robot trajectories are shown as dashed curves.

prior knowledge of the workspace was available but the location of all the static polygonal obstacles in the workspace were unknown to the robot. The robot was modeled as a small circle (12 cm in diameter) and 12 virtual sensors mimic infra-red sensors placed in the form of a circle along the circumference of the robot. They were spaced by 30° and they had a distance range of 30 cm. In our implementation, we also applied hysteresis to the state transition in order to avoid a chattering situation when switching between two behaviors occurs.

The user-defined parameters in the simulation were set as follows

$$a_{\text{T,max}} = 0.5 \text{ m/s}^2,\ \ a_{\text{L,max}} = 0.5 \text{ m/s}^2,\ \ \omega_{\max} = 2 \text{ rad/s},$$
$$v_0 = 0.2 \text{ m/s},\ \ N = 50 \text{ steps},\ \ T_s = 0.05 \text{ s},\ \ d_r = 0.2 \text{ m},$$
$$d_{\max} = 0.2 \text{ m},\ \ d_{\min} = 0.1 \text{ m},\ \ \theta_{obs,\max} = 75°,\ \ d_s = 0.5 \text{ m},$$
$$Q(1,1) = \frac{1000}{1 + 100|y_e|},\ \ Q(2,2) = 1,\ \ R = 0.01.$$


(a)

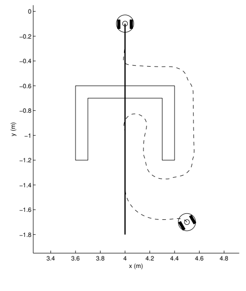
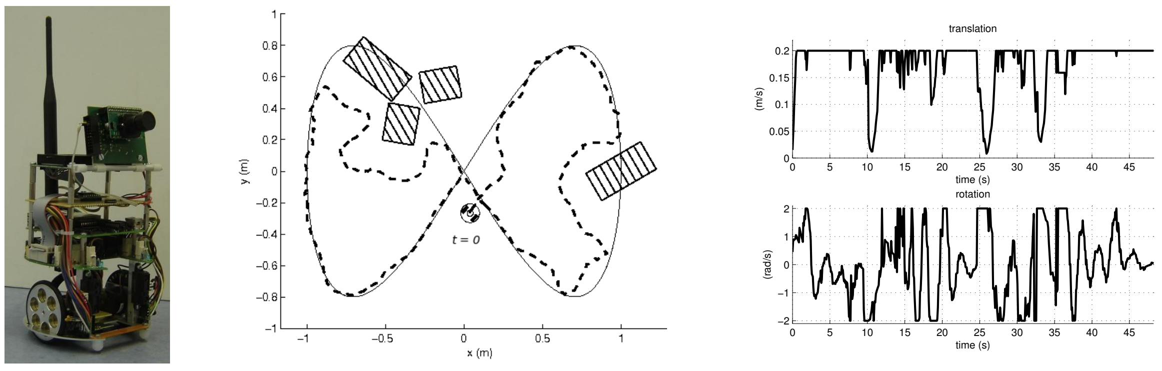

(b)

Fig. 6. (a) The obstacle-avoidance behavior becomes active because sensor information indicates that there are three obstacles detected and the one in front of the robot can influence the robot motion. Then, the path replanning module needs to decide that the robot should turn left or turn right. (b) The planner selects right turning. The robot keeps following that obstacle on the right until no relevant obstacle is detected. When no obstacle is detected, the path-following behavior becomes active; the robot keeps following the reference path.

The simulation results obtained in complex scenarios with different obstacle configurations are presented in Fig. 7. As seen in the results, it can be concluded that the robot successfully follows the reference path and avoids all obstacles. Notice that in Fig. 7(a), the robot at coordinate $(-4, 0)$ decided to turn left and then it was going to be caught in a trap. The trap was discovered by monitoring the path's parameter. If the path's parameter before the obstacle-avoidance behavior becomes active is greater than the path's parameter after the obstacle-avoidance behavior becomes inactive, this implies that the robot is very likely about to run in a closed loop. To avoid trap-situations, the robot followed the obstacle at its right at this time (see Fig. 7(a)).

To show the usefulness of our approach, the unicycle-type mobile robot, shown in Fig. 8(a), was used in real-world experiments. The robot controller is an ATMEGA644 microprocessor with 64 KB flash program memory, 16 MHz clock frequency and 4 KB SRAM. The robot orientation was measured by a Devantech CMPS03 compass. The localization was given by a camera looking down upon the robot's workplace and a PC was used to compute control inputs and then sent these inputs to the robot via WLAN. An eight-shaped curve similar to the path in the second simulation was employed in this experiment. Gaussian noise with a fixed variance was added to all distance measurements, given by virtual distance sensors. Due to high computational time in optimization solving, only 10 steps were used for the prediction horizon and the cycle time was set to 0.1 s. Our software was developed in C++ with the additional use of some geometrical calculations from CGAL (Computational Geometry Algorithms Library - online available: http://www.cgal.org). The free package DONLP2 (Spellucci, 1998) was used to solve the optimization problem.

The experimental results are plotted in Fig. 8(b). As depicted in Fig. 8(c), the robot was able to travel at the desired speed $v_0 = 0.2$ m/s in case of no obstacle or no sharp turning curve. The velocity selection module optimized the forward and rotational velocities, if environmental and/or robot constraints were violated, while the path replanning module locally handled obstacle avoidance.



(a) (b) (c)

Fig. 8. The experimental results by using our path following control framework: (a) the mobile robot (12 cm in diameter) used in our experiments, (b) the robot trajectories (dashed curves), the reference path (solid curves), and the obstacles (hatched rectangles), and (c) the robot's velocities.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel approach to satisfy our three objectives, i.e., path following, obstacle avoidance, and forward velocity selection. It integrates an obstacle-avoidance behavior using sensor information with a path-following behavior based on global knowledge and then applies this solution to partially-known static environments. Using sensor information, the path replanning module generates new waypoints to bring the robot away from the obstacles. The MPC law is used to produce a sequence of angular velocities and the forward velocity selection module provides a forward velocity by taking into account robot dynamics and environmental constraints. Simulation and experimental results clearly show that our strategy is able to control a robot to follow a reference path in complex environments.

Currently, we are integrating more realistic sensor models to our framework including global path planning for complex environments. One problem in our strategy is that an abrupt change of path orientation and curvature causes the angular commands to be non-smooth. This situation can be improved by connecting smoothly the current reference position with the next waypoint, which is the topic of future work. Furthermore, we will extend our approach to accomplish the path following task in a dynamic environment with moving obstacles.

REFERENCES

Al-Hiddabi, S.A. and McClamroch, N.H. (2002). Tracking and maneuver regulation control for nonlinear non-minimum phase systems: application to flight control. *IEEE Trans. on Control Systems Technology*, 10(6), 780–792.

Altafini, C. (2002). Following a path of varying curvature as an output regulation problem. *IEEE Trans. on Automatic Control*, 47(9), 1551–1556.

Bak, M., Poulsen, N.K., and Ravn, O. (2001). Path following mobile robot in the presence of velocity constraints. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark.

Encarnação, P. and Pascoal, A. (2000). 3d path following for autonomous underwater vehicle. In *Proc. of the IEEE Conf. on Decision and Control*, 2977–2982. Sydney, Australia.

Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., and Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Trans. on Control Systems Technology*, 15(3), 566–580.

Gu, D. and Hu, H. (2006). Receding horizon tracking control of wheeled mobile robots. *IEEE Trans. on Control Systems Technology*, 14(4), 743–749.

Indiveri, G., Nüchter, A., and Lingemann, K. (2007). High speed differential drive mobile robot path following control with bounded wheel speed commands. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2202–2207. Roma, Italy.

Klančar, G. and Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6), 460–469.

Kwon, W.H. and Han, S. (2005). *Receding horizon control: Model predictive control for state models*. Springer-Verlag, London.

Lages, W.F. and Alves, J.A.V. (2006). Real-time control of a mobile robot using linearized model predictive control. In *Proc. of IFAC Symposium on Mechatronic Systems*, 968–973. Heidelberg, Germany.

Lapierre, L., Zapata, R., and Lepinay, P. (2007). Combined path-following and obstacle avoidance control of a wheeled robot. *Int. Journal of Robotics Research*, 26(4), 361–376.

Maček, K., Philippsen, R., and Siegwart, R. (2009). Path following for autonomous vehicle navigation based on kinodynamic control. *Journal of Computing and Information Technology*, 17(1), 17–26.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M. (2000). Constrained model predictive control: stability and optimality. *Automatica*, 36, 789–814.

Micaelli, A. and Samson, C. (1993). Trajectory-tracking for unicycle-type and two-steering-wheels mobile robots. Technical Report 2097, INRIA Sophia-Antipolis.

Morin, P. and Samson, C. (2008). *Springer Handbook of Robotics*, chapter 34. Motion control of wheeled mobile robot, 799–826. Springer Berlin Heidelberg.

Ollero, A. and Amidi, O. (1991). Predictive path tracking of mobile robots: Application to the CMU Navlab. In *Proc. of Int. Conf. on Advanced Robotics*, 1081–1086. Pisa, Italy.

Samson, C. (1995). Control of chained systems: Application to path-following and time-varying point stabilization of mobile robots. *IEEE Trans. on Automatic Control*, 40(1), 64–77.

Spellucci, P. (1998). An SQP method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82(3), 413–448.