Nonlinear Predictive Control of an Omnidirectional Robot Dribbling a Rolling Ball

Xiang Li and Andreas Zell

***Abstract*— This paper focuses on the dribbling control problem of an omnidirectional mobile robot and a rolling ball in the RoboCup Middle Size domain. Because the ball easily slides away from the robot when the ball moves along a curve, dribbling control is more challenging than the normal mobile robot motion control problem. Based on an introduced reference point with respect to the robot body and a sophisticated planning method of robot pose, the nonlinear predictive control is used to steer the robot to follow the planned poses so as to prevent the ball from leaving the robot. Real-world experiments showed that nonlinear predictive control is capable of solving the pose following problem in a real-time application.**

I. INTRODUCTION

Dribbling control is a challenging task in the RoboCup Middle Size domain. As the rules only allow one third of the ball to be covered by the robot, the dribbling behavior has high requirements on the robot motion control. For RoboCup robots, dribbling refers to a robot maneuvering a rolling ball through consecutive and short contacts in a dynamical obstacles environment, such that the robot is able to bypass opponents with the ball and shoot a goal.

The greatest challenge in dribbling control is not to loose the ball when the robot moves along a curve. Due to unknown factors of the interactions between the ball and the robot, such as contact points and friction coefficients, it is hard to calculate the desired forces, and the robot actions which can exert the desired forces on the ball. Considering the difficulty of determining the ball's motion when it is in continuous contact with a robot, many RoboCup teams use artificial intelligence methods, such as neural networks ([1], [2]) or reinforcement learning ([3], [4], [5]) to learn some basic skills of the robot, such as kicking and dribbling. Although suitable simulation systems can support learning before the experiments with a real robot, the skill learning takes a long time and results in high computational cost. Other teams fulfill the dribbling task by planning robot moving paths [6] or designing the behavior-based approach [7], where the mobile robot is controlled towards the desired postures. However, the employed control methods only consider to decrease the error between the current robot state and the desired one, but do not take the robot constraints into account.

Along the line of our earlier research ([8]), we focus on solving the dribbling problem by controlling the robot to follow desired poses, and take an introduced reference point as the controlled object instead of the robot's center of mass. In this work, we utilize the nonlinear predictive control (NPC) method, such that the robot constraints and more information of the given path can be considered in the controller design. The real experimental results show that the nonlinear predictive controller effectively solve the pose following problem, and the robot can dribble the ball in a small field avoiding collisions with a static or moving obstacle.

Xiang Li and Andreas Zell are with the Department of Computer Architecture, Wilhelm-Schickard-Institute, University of Tübingen, Sand 1, 72076 Tübingen, Germany {xiang.li, andreas.zell}@uni-tuebingen.de

Fig. 1. The real omnidirectional robot and the ball

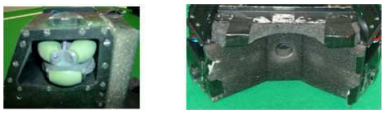


Fig. 2. The wheel and the dribbling mechanism of the robot

II. ROBOT AND BALL

Fig. 1 shows our omnidirectional robot. It has three Swedish 90-degree wheels mounted symmetrically with 120 degrees between each other. Three Maxon 60W motors drive the robot with a maximum speed of 2.6 m/s. Because of six rollers mounted along the wheel's periphery seen in Fig. 2, the robot is capable of moving perpendicular to the normal rotating direction of the wheel's axis. Besides the wheel encoders, the only sensor on the robot is an AVT Marlin 50fps color camera with a resolution of 780×580. It is assembled pointing up towards a hyperbolic mirror on the top of robot, and transmits images via IEEE 1394 to a Pentium-M 2GHz onboard PC with 1GB RAM [9].

In the view of hardware, an effective dribbling mechanism can help a lot in keeping the ball. Our robot's dribbler system shown in Fig. 2 consists of a front part and an upper part. The front one is comprised of three spongy blocks in a concave form, which prevents the ball from sliding away

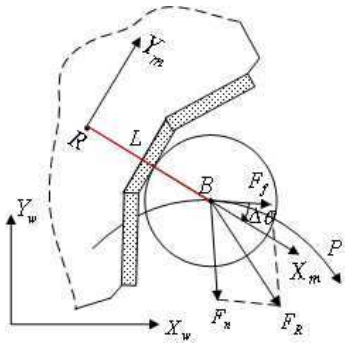


Fig. 3. Forces analysis in a ideal dribbling situation, where the ball moves along a curve and locates at the front of the robot

from the robot. The upper one is a plastic board, which gives a backwards spin to the ball, such that the ball can move back when it loses contact with the robot. The size of the dribblers is also designed to meet the rule of ball manipulation in the RoboCup Middle Size league.

The official tournament ball used in Robocup matches a FIFA standard size 5 football. The only difference is that the ball has orange color, which is helpful for the ball tracking in the indoor Robocup environment.

III. DRIBBLING ANALYSIS

Because of the difficulty of modeling the interactions between the robot and the ball during the dribbling process, we concentrate on solving the dribbling control problem by steering the robot to follow the planned poses.

Fig.3 illustrates an ideal situation, where the ball moves along a curve P and locates at the front of the robot with respect to the world coordinate system $[X_w, Y_w]$ fixed on the field. In order to provide the ball with enough pushing force F_t and centripetal force F_n, the robot orientation θ needs to have some deviation $\Delta\theta$ to the tangent direction θ_P of the curve, and this angle deviation has a relationship with the centripetal acceleration cv_d^2, where c is the curvature of the path and v_d is the ideal moving velocity of the ball. We select $\Delta\theta$ proportional to the centripetal acceleration with a positive parameter k_θ: $\Delta\theta = k_\theta c v_d^2$. Then the ideal robot orientation θ^d is given by

$$\theta^d = \theta_P + \Delta\theta, \tag{1}$$

In order to get the ideal robot pose, we have to determine the ideal robot position. It is noticed in Fig.3, if the ball locates at the ideal position, its center B has the coordinate $(L, 0)$ with respect to the robot coordinate system $[X_m, Y_m]$, whose X_m axis denotes robot orientation. L is the ideal distance between the robot's center of mass R and B. When we introduce a reference point E having the fixed coordinate $(L, 0)$ in the robot coordinate system, point E will follow the path P in ideal situations. Therefore, the ideal robot position can be determined with the ideal orientation and the position of point E as follows;

$$x_R^d = x_E^d - L\cos\theta^d, \tag{2}$$

$$y_R^d = y_E^d - L\sin\theta^d, \tag{3}$$

where (x_R^d, y_R^d) and (x_E^d, y_E^d) are the ideal robot position and point E's position with respect to the world coordinate system, respectively. Then the dribbling control is to steer the robot to follow the ideal pose (x_R^d, y_R^d, θ^d).

During the dribbling process, the ball can only be pushed but not be pulled. To decrease the ball's speed, the robot has to move ahead and hindes the ball's movement. when increasing the ball's speed, the robot needs to stay behind the ball and push it. Therefore, varying ball speeds will unsmooth the robot movement and increase the possibility of losing the ball. In our research, we control the ball with a constant high speed moving along the desired path, which not only facilitates the robot motion control, but also ensures the ball's fast moving in the RoboCup matches.

When the ball keeps a constant velocity, point E also moves along the desired path with this constant velocity in ideal situations. To deal with this requirement, a direct control of the robot is not a convenient way, because it brings quadratic constraints to robot inputs. But if we control point E to follow the desired path and the robot to take the desired orientations, which only ensures that the robot has the desired poses, but also induces only linear constraints of system inputs.

To plan the desired path, a potential field and grid-based path planning method have been employed in our experiments ([10]). By subdividing the environment into a grid space with unit sized grid cells and calculating the potential value of these cells, a path composed of a set of points is planned from the target to the robot. This backwards planning method avoids the robot directly heading into obstacles, but leads the robot to move around them. Replacing the robot with point E in the path planning method, a desired path for point E is generated. With this path and the desired robot orientation, our task is to design a controller such that point E can best follow the path and the robot can best take the desired orientations.

IV. SYSTEM KINEMATIC MODEL

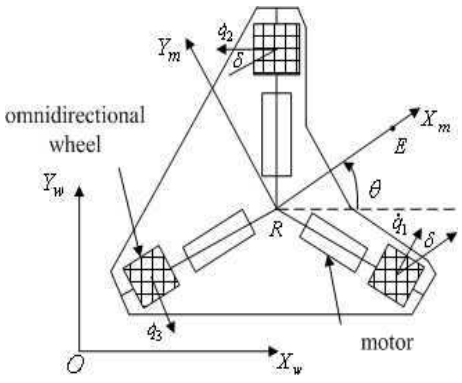


Fig. 4. Kinematics diagram of the base of an omnidirectional robot

Fig. 4 shows the base of our omnidirectional robot. Each wheel has the same distance L_w to the robot's center of mass

R. δ refers to the wheel orientation in the robot coordinate system and is equal to 30 degrees. θ denotes the robot orientation, which is the direction angle of the axis X_m in the world coordinate system. The kinematic model of the robot is as follows:

$$\dot{\mathbf{x}}_R^w = \begin{bmatrix} \frac{2}{3}cos(\theta+\delta) & -\frac{2}{3}cos(\theta-\delta) & \frac{2}{3}sin\theta \\ \frac{2}{3}sin(\theta+\delta) & -\frac{2}{3}sin(\theta-\delta) & -\frac{2}{3}cos\theta \\ \frac{1}{3L_w} & \frac{1}{3L_w} & \frac{1}{3L_w} \end{bmatrix} \dot{\mathbf{q}}, \quad (4)$$

where $\mathbf{x}_R^w$ is the robot state vector with respect to the world coordinate system, which is composed of the robot position x_R^w, y_R^w and orientation θ; $\dot{\mathbf{x}}_R^w$ is the robot velocity vector, which includes the robot translation velocities $\dot{x}_R^w$ and $\dot{y}_R^w$ and robot rotation velocity ω; $\dot{\mathbf{q}}$ is the vector of wheel velocities $[\dot{q}_1\ \dot{q}_2\ \dot{q}_3]^T$, and $\dot{q}_i (i = 1, 2, 3)$ is the i-th wheel velocity, which is equal to the wheel's radius multiplied by the wheel's angular velocity. As the motor's voltage and current are magnitude limited, the maximum wheel velocity is limited by $\dot{q}_m$, namely $|\dot{q}_i| \leq \dot{q}_m$.

Based on the definition of coordinate systems, we get the following kinematic model of point E :

$$\dot{\mathbf{x}}_E^w = \begin{bmatrix} \dot{x}_E^w \\ \dot{y}_E^w \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}, \quad (5)$$

where $\dot{\mathbf{x}}_E^w$ is the velocity vector of point E observed in the world coordinate system; The system input vector **u** includes point E's translation velocities $\dot{x}_E^m$ and $\dot{y}_E^m$ observed in the robot coordinate system and the robot rotation velocity ω.

Because point E is considered with constant coordinate $(L, 0)$ with respect to the robot coordinate system, its velocities have the following relationship with those of the robot,

$$\begin{bmatrix} \dot{x}_R^m \\ \dot{y}_R^m \end{bmatrix} = \begin{bmatrix} \dot{x}_E^m \\ \dot{y}_E^m - L\omega \end{bmatrix}, \quad (6)$$

Substituting (5) and (6) into the inverted equations of the robot kinematic model (4), we get the following constraints of system input **u** with respect to the limited wheel velocity:

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos(\theta+\delta) & \sin(\theta+\delta) & -L\sin\delta + L_w \\ -\cos(\theta-\delta) & -\sin(\theta-\delta) & -L\sin\delta + L_w \\ \sin\theta & -\cos\theta & L + L_w \end{bmatrix} \mathbf{u},$$

$$-\dot{\mathbf{q}}_m \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_m \quad (7)$$

V. POSE FOLLOWING CONTROL

The pose following problem is illustrated in Fig. 5. P denotes the given path of point E. Point Q is the orthogonal projection of E on the path P. l denotes the distance error. θ^d is the desired robot orientation at point Q. The angular error is defined as $\tilde{\theta} = \theta^d - \theta$.

Pose following control is to find proper inputs such that the distance error l and angular error $\tilde{\theta}$ tend to zero. Most pose following control methods are based on finding the errors between the current robot poses and the desired poses, then

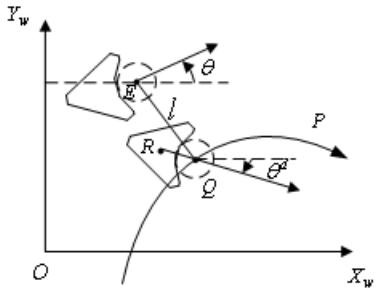


Fig. 5. Illustration of the pose following problem

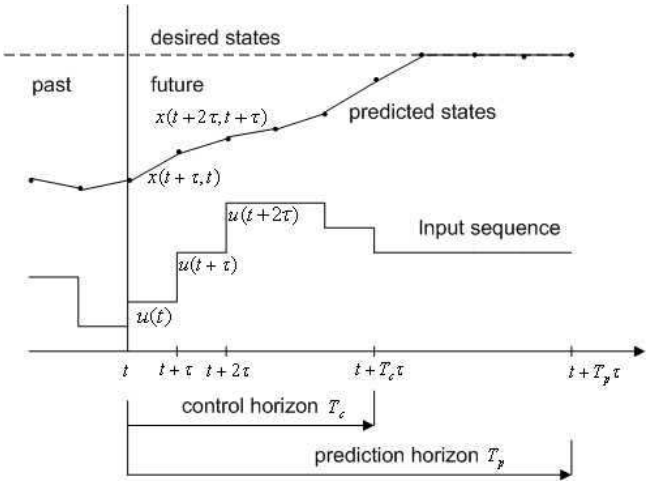


Fig. 6. Principle of nonlinear predictive control

use a PID controller ([11]) or nonlinear controller ([12], [13]) to decrease these errors. However, only using the current errors makes these methods ignore the potential opportunity of improving the control performance by considering more information of the given path. Moreover, these methods rarely take the robot's dynamic and kinematic limitations into account. When a robot makes a sharp turn with high translation and rotation velocities, motors will easily come into saturation, which can degrade the robot's performance, even destroy the stability ([14], [15]).

As an attractive optimal control method, the nonlinear predictive control has been used in our pose following problem, because it can easily handle the system constraints and take future information into the controller design ([16]). Fig. 6 shows the basic principle of nonlinear predictive control. where τ is the predicted sampling interval; Control values $\mathbf{u}(t + i\tau)$ for $i \in [0, ..., T_c]$ are constant on each interval; $\mathbf{x}(t+j\tau, t+(j-1)\tau)$ for $j \in [1, ..., T_p]$ denotes the predicted state value at time $t+j\tau$ based on the one at time $t+(j-1)\tau$. The basic idea of nonlinear predictive control (NPC) includes three steps: (i) at each time step t, predict the future behavior of the system over a prediction horizon T_p, (ii) calculate inputs during a control horizon T_c ($T_c \leq T_p$) such that a predefined open-loop objective function is optimized under the system and inputs constraints, (iii) take the first optimal input value as the current input.

A. NPC Formulation

In order to obtain the predicted states in the future, we discretise the system (5) as:

$$\theta(k+1) = \theta(k) + \omega(k)\tau, \tag{8}$$

$$x_E^w(k+1) = x_E^w(k) + \frac{\dot{x}_E^m(k)}{\omega(k)}[\sin(\theta(k+1)) - \sin(\theta(k))] + \frac{\dot{y}_E^m(k)}{\omega(k)}[\cos(\theta(k+1)) - \cos(\theta(k))], \tag{9}$$

$$y_E^w(k+1) = y_E^w(k) - \frac{\dot{x}_E^m(k)}{\omega(k)}[\cos(\theta(k+1)) - \cos(\theta(k))] + \frac{\dot{y}_E^m(k)}{\omega(k)}[\sin(\theta(k+1)) - \sin(\theta(k))], \tag{10}$$

if $\omega(k) = 0$:

$$x_E^w(k+1) = x_E^w(k) + [\dot{x}_E^m(k)\cos(\theta(k)) - \dot{y}_E^m(k)\sin(\theta(k))]\tau, \tag{11}$$

$$y_E^w(k+1) = y_E^w(k) + [\dot{x}_E^m(k)\sin(\theta_R(k)) + \dot{y}_E^m(k)\cos(\theta_R(k))]\tau. \tag{12}$$

We define the following objective function:

$$J(t) = \sum_{i=1}^{T_p} \|\hat{\mathbf{x}}_E^w(t+i\tau, t+(i-1)\tau) - \mathbf{x}_E^d(t+i\tau)\|_Q^2 + \sum_{j=0}^{T_c-1} \|\mathbf{u}(t+(j+1)\tau) - \mathbf{u}(t+j\tau)\|_R^2 \tag{13}$$

where $\hat{\mathbf{x}}_E^w(t+\tau, t)$ denotes the predicted state of point E at time $t+\tau$ based on the one of time t, which is not exactly same as the actual one. $\mathbf{x}_E^d$ refers to the desired state. Q and R are symmetric positive weight matrices with appropriate dimensions. The first summand presents the performance of the system following the desired states, the second one limits the input changing and smooths the robot's movement.

Then at each time step t, the following finite horizon open-loop optimal problem is to be solved:

$$\min_{\mathbf{u}(t)\ldots\mathbf{u}(t+T_c\tau)} (J(t)) \tag{14}$$

subject to:

$$\hat{\mathbf{x}}_E^w(t+i\tau) = \mathbf{f}(\mathbf{x}_E^w(t+(i-1)\tau), \mathbf{u}(t+(i-1)\tau)), \tag{15}$$

$$\mathbf{g}(\hat{\mathbf{x}}_E^w(t+i\tau), \mathbf{u}(t+i\tau)) \leq 0, \tag{16}$$

$$\|\mathbf{u}(t+j\tau)\| \leq \mathbf{u}_m,\ \forall j \in [0, ..., T_c], \tag{17}$$

$$\mathbf{u}(t+j\tau) = \mathbf{u}(t+T_c\tau),\ \forall j \in [T_c, ..., T_p] \tag{18}$$

where the function $\mathbf{f}(\mathbf{x}_E^w, \mathbf{u})$ denotes the discretized system equations (8-12), function $\mathbf{g}(\mathbf{x}_E^w, \mathbf{u})$ denotes the system constraints (7), $\mathbf{u}_m$ is the inputs limitation.

B. Control Stability

It is well known that the above finite horizon strategy can not guarantee closed-loop stability. In many proposed methods, adding a terminal penalty to the cost function and constraints to the system terminal state is a computationally feasible method to achieve closed-loop stability ([16], [17]). Although the treatment of the stability conditions is not considered in this work, the choice of a suitable terminal penalty and terminal constrains will be our work in the future.

C. Nonlinear Programming Solver

Although the high computational demands of solving the upper nonlinear finite optimization problem make NPC hard to be implemented in applications with fast sampling time and limited computational resources ([16]), many research results show the possibility of applying nonlinear predictive controllers in some real-time processes, such as control of an autonomous vehicle ([18]) and a full-scale aircraft ([19]). As one often used method, sequential quadratic programming (SQP) is utilized in our project. We use the software *donlp2-intv-dyn* by P. Spellucci ([20], [21]), which is a general purpose nonlinear optimizer and can be found at *http://plato.la.asu.edu/donlp2.html*. This optimizer implements a sequential equality constrained quadratic programming method.

D. Computational Delays

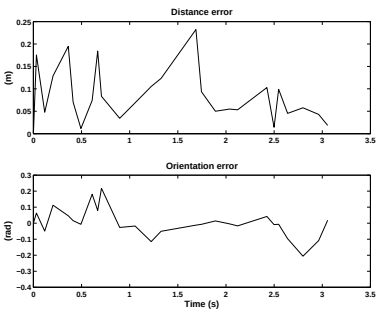
In reality, on-line solving the optimization problem requires time τ^c. At every time step t, when the optimal input serial in the control horizon is found, the first control action is taken at $t+\tau^c$. This computational delay may influence the control performance and system stability. Therefore, it is important to take the computational delay into account. A common method is to estimate the value of τ^c at time step t, predict the system state $\hat{\mathbf{x}}(t+\tau^c)$ with $\mathbf{x}(t)$ and $\mathbf{u}(t)$, solve the optimization problem based on $\hat{\mathbf{x}}(t+\tau^c)$, and take the first optimal control value as the system input.

VI. EXPERIMENTAL RESULTS

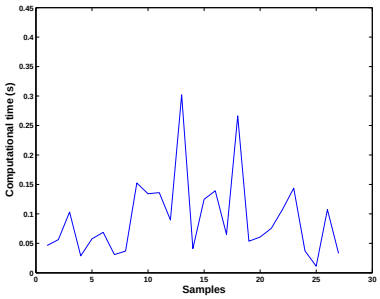
The real-world experiments were made in our robot laboratory having a carpet covered field with a size of 5.1×4.2 m^2. Based on the real-time output signal of the camera, a self-localization algorithm described in [22] determines the robot's position in the play field, and a fast object detection algorithm is used to get the real world positions of other objects as introduced in [23].

Considering the desired constant velocity v_d , we transform the input vector $\mathbf{u}$ in kinematics model (5) into $[v_d\cos\alpha, v_d\sin\alpha, \omega]^T$, where α is point E's moving direction with respect to the robot coordinate system. Therefore, we have two inputs $\|\alpha\| \leq \frac{\pi}{2}$ and $\|\omega\| \leq \omega_m$ to control our robot.

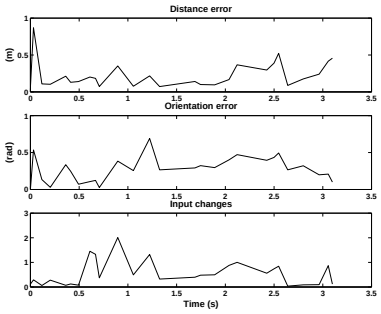
In the experiments, the following parameters have been used in the nonlinear predictive control: the predicted sampling time: $\tau = 0.4$s; the predictive horizon is equal to the control horizon: $T_p = T_c = 3$, constraints on wheel



(a) Measured distance and orientation errors

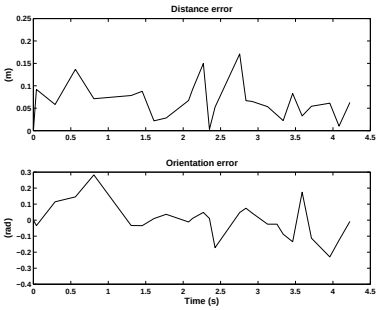


(b) Computational time at each time step

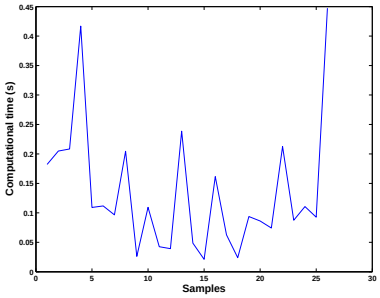


(c) NPC objective function's values

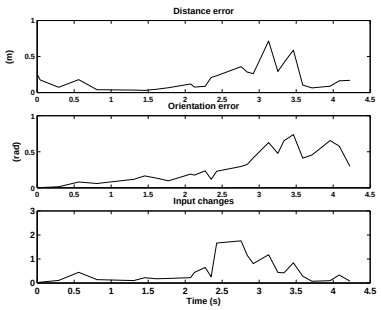
Fig. 7. Results in the scenario with a static obstacle



(a) Measured distance and orientation errors

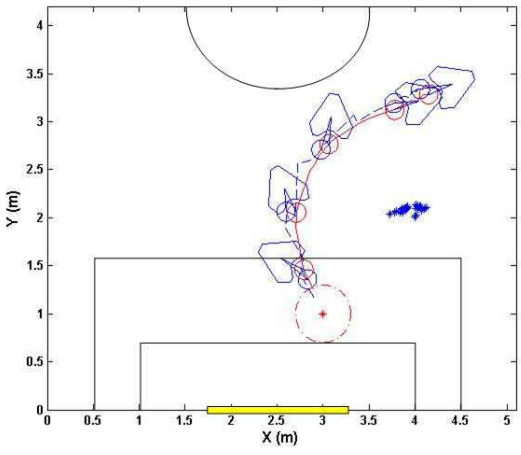


(b) Computational time at each time step

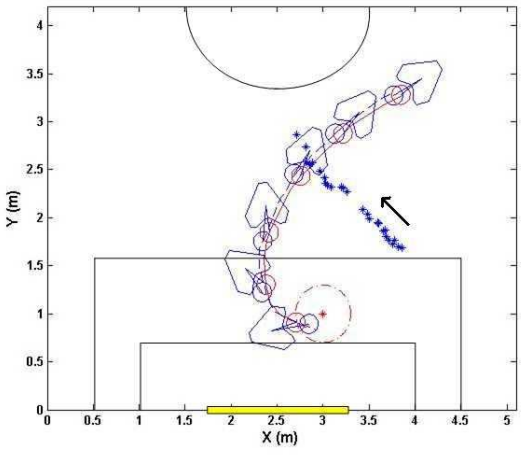


(c) NPC objective function's values

Fig. 8. Results in the scenario with a moving obstacle



(a) Static obstacle



(b) Moving obstacle

Fig. 9. Measured trajectories in the scenarios with a static and moving obstacle. (solid line: point E, dashed line: ball, asterisk: obstacle; The Arrow points the obstacle's moving direction)

velocity: $q_m = 1.9$m/s, constraints on robot rotation velocity: $\omega_m = 2.0$rad/s, weight matrices$Q = [5.0\,0\,0; 0\,5.0\,0; 0\,0\,6.0]$, $R = [1.0\ \ 0; 0\ \ 1.0]$. the computational delay: $\tau^c = 0.15$s.

In the experiments an obstacle is located between the ball's initial position and the target position $(3, 1)$. The robot is required to dribble the ball to the target position without colliding with the obstacle. When point E has a distance of less than 0.3m to the target, the robot stops moving. At every sampling time, the real time path from the current position of point E to the target position is created by the path planner, and we select $k_\theta = 0.5$ to calculate the desired robot orientation.

In two experiment scenarios, the obstacle was set to be static or to move linearly, respectively. The desired ball's velocity v_d was set to 1.3m/s in the static obstacle scenario, and 1.0 m/s in the moving obstacle scenario. Fig. 7(a) and 8(a) show that the nonlinear predictive control can solve the real-time pose following problem with good performance, because at most time steps, the distance errors are less than 0.3m, the orientation errors are less than 0.25rad. Fig. 7(c) and 8(c) show that the NPC get small values of the cost function, and Fig. 7(b) and 8(b) show that the NPC is fast enough to control our robot, with a computational time less than 0.25s at most time steps. Fig. 9(a) and 9(b) show the measured trajectories of point E, ball and obstacle. Although we can see that the ball is not always located at the position of point E, the robot can keep the controlling of the ball in the dribbling process.

VII. CONCLUSIONS

In this paper a new control method for an omnidirectional robot dribbling a rolling ball is presented. Based on the analysis of the dribbling process, this approach solves the control problem of the consecutive mobile robot pushing operation by introducing a reference point as the controlled object and a sophisticated planning method of the desired robot orientations. The nonlinear predictive control method is used to steer the reference point to follow the given path and the robot to take the desired orientations. To test the control method, real-world experiments were made in a small lab field, where the ball was dribbled from an initial position to a target with a static or moving obstacle. The desired velocities of the ball were set to 1.3 m/s and 1.0 m/s in the two different scenarios. Experimental results show that the nonlinear predictive control effectively solved the pose following problem in real-time, and the desired robot pose can guarantee the ball moving to the target position without colliding with the obstacle.

REFERENCES

[1] V. Ciesielski and S. Y. Lai, "Developing a dribble-and-score behaviour for robot soccer using neuro evolution," in *Workshop on Intelligent and Evolutionary Systems*, Dunedin, New Zealand, 2001, pp. 70–78.

[2] B. Hecht, M. Simon, O. Tenchio, *et al.* (2005) A neural coach for teaching robots using diagrams. [Online]. Available: http://page.mi.fu-berlin.de/∼rojas/2005/NeuralCoach_Hecht_Rojas.pdf/

[3] M. Riedmiller and A. Merke, "Using machine learning techniques in complex multi-agent domains," *erspectives on Adaptivity and Learning*, 2002.

[4] P. Jonker, B. Terwijn, J. Kuznetsov, and B. van Driel, "Algorithmic function of the clockwork orange robot soccer team," in *Proc. 6th Int. Workshop on the Algorithmic Foundations of Robotics*, Zeist/Utrecht, Netherland, 2004, pp. 1–10.

[5] T. Gabel and M. Riedmiller, "Learning a partial behavior for a competitive robotic soccer agent," *Künstliche Intelligenz*, vol. 20(2), pp. 18–23, 2006.

[6] T. Weigel, W. Auerbach, M. Dietl, B. Dmler, *et al.*, "CS Freiburg: Doing the right thing in a group," in *RoboCup 2000: Robot Soccer World Cup IV*, Melbourne, Australia, 2001, pp. 52–63.

[7] R. Emery and T. Balch, "Behavior-based control of a non-holonomic robot in pushing tasks," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001, pp. 2381–2388.

[8] X. Li, M. Wang, and A. Zell, "Dribbling control of omnidirectional soccer robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, 2007, pp. 2623–2628.

[9] P. Heinemann, H. Becker, J. Haase, and A. Zell, "The attempto Tübingen robot soccer team 2006," in *Robot Soccer World Cup X*. Springer, 2007.

[10] P. Heinemann, "Cooperative multi-robot soccer in a highly dynamic environment," Ph.D. dissertation, University of Tübingen, 2007.

[11] R. Rojas and A. G. Foerster, "Holonomic control of a robot with an omnidirectional drive," *künstliche Intelligenz*, vol. 20(2), 2006.

[12] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Path-following for nonminimum phase systems removes performance limitations," in *IEEE Transactions on Automatic Control*, vol. 50(2), 2005, pp. 234–239.

[13] K. Maček, I. Petrović, and R. Riegwart, "A control method for stable and smooth path following of mobile robots," in *Proc. 2nd European Conference on Mobile Robots - ECMR 2005*, Ancona, Italy, 2005, pp. 128–133.

[14] G. Indiveri, J. Paulus, and P. G. P. öger, "Motion control of swedish wheeled mobile robots in the presence of actuator saturation," in *Proc. 10th annual RoboCup International Symposium*, 2006.

[15] A. S. conceicao, A. Moreira, and P. j. Costa, "Trajectory tracking for omni-directional mobile robots based on restrictions of the motor's velocities," in *Proc. 8th International IFAC Symposium on Robot Control*, 2006.

[16] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," in *21st Benelux Meeting on Systems and Control*, Veldhoven, Netherlands, 2002.

[17] D. Gu and H. Hu, "Receding horizon tracking control of wheel mobile robots," *IEEE Transaction on Control Systems Technology*, vol. 14, july 2006.

[18] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat, "Predictive control approach to autonomous vehicle steering," in *Proc. American Control Conference*, Minneapolis, Minnesota USA, 2006.

[19] T. Keviczky and G. J. Balas, "Software-enabled receding horizon control for autonomous uav guidance," in *Proc. AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, California USA, 2005.

[20] P. Spellucci, "A new technique for inconsistent qp-problems in the sqp-method," *Mathematical Methods of Operations Research*, vol. 46, pp. 355–400, 1998.

[21] ——, "An "sqp" method for general nonlinear programs using only equality constrained subproblems," *Mathematical Programming*, vol. 82, pp. 413–448, 1998.

[22] P. Heinemann, J. Haase, and A. Zell, "A combined monte-carlo localization and tracking algorithm for robocup," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 1535–1540.

[23] P. Heinemann, T. Rückstiess, and A. Zell, "Fast and accurate environment modelling using omnidirectional vision," *Dynamic Perception*, pp. 9–14, 2004.