2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Reactive Trajectory Tracking for Mobile Robots based on Non Linear Model Predictive Control

Stavros G. Vougioukas, *Member, IEEE*

***Abstract*— In this paper, a nonlinear model predictive tracking (NMPT) controller for mobile robots is presented. The basic idea is to use a motion model for the vehicle and compute in real-time an optimal M-step-ahead control sequence, which minimizes the total M+1 step tracking error of the projected motion. In the presence of obstacles, the controller deviates from the reference trajectory by incorporating into the optimization obstacle-distance information from range sensors (e.g., laser scanner, ultrasound). Numerous simulations were performed and the NMPT consistently converged to the desired trajectories and followed them accurately, despite large initial errors and discontinuities in the desired velocities and orientations. The controller's performance depended strongly on parameters such as the optimization horizon M, and the cost-weights assigned to the various tracking errors. The optimization horizon regulates a trade-off between timely obstacle avoidance and tracking quality (large M) vs. consistently fast convergence (small M). The cost-weights affect tracking quality and also the shape of the path, by regulating trade-offs among position, orientation, and velocity errors. Overall, NMPT seems to offer a promising approach for advanced precision guidance applications, and deserves further investigation.**

I. Introduction

High precision motion tracking is desirable in many mobile robot applications, as well as in tractor auto-steering for precision farming. The desired path may be defined by a number of waypoints and orientations, or by analytical expressions. In some cases the desired vehicle speed is either constant, or it is free to take any values within a certain operational range. The goal is that the average and maximum deviation between the vehicle's traveled path and the desired path are minimized. This problem is called path tracking and the path tracking error is defined as the shortest distance between the tractor's control point and the desired path. Various approaches have been proposed for path tracking, such as pure-pursuit [1], sliding-mode control [2], [3], nonlinear proportional control [4], and vector pursuit [5]. In some applications the desired path is accompanied by a desired velocity, or even acceleration profile. This problem is referred to as trajectory tracking and a desired trajectory point must be available to the robot digital controller at every sample.

Manuscript received January 31, 2007.

S. G. Vougioukas is with the Agricultural Engineering Department, Aristotle University, 54124 Thessaloniki, GREECE, (e-mail: bougis@auth.gr).

The tracking error is defined at each controller sample as the difference between the desired and actual trajectory points. Various trajectory tracking controllers have been developed for industrial robots, with PID control being the most widely used technique. However, linear controllers cannot offer good tracking performance for non-holonomic robot complex maneuvers, such as sharp turns and reverse motions. This is a well known theoretical result [6] for under-actuated non holonomic systems, such as wheeled vehicles under the no-slip constraint. Various control techniques have been proposed for trajectory tracking for car-like robots. In [7] a time-varying LQR controller is computed for the non holonomic system linearized about its path. One problem with this approach is that the LQR gains are path-dependent and hence the controller requires extensive tuning for each type of path. In [8] iterative model predictive control was used. This approach is similar to non linear model predictive control, but does not consider optimization or constraints explicitly; instead it uses a gradient based algorithm to reduce the predicted state error after a fixed number of trajectory points. The look-ahead point in this approach must be carefully selected; a distant point may result in corner-cutting and increased tracking error, just like in pure pursuit.

In addition to small tracking errors, real-world applications require robust navigation in the presence of obstacles. Collision-free motion trajectories are typically computed by some motion planning algorithm based on a map of the environment. During the actual motion execution it is possible that obstacles appear in the vehicle's path, which had not been present in the planning phase (e.g., animals, humans, machines). This may also happen because of imprecision in the field map, or vehicle localization errors. The on-line alteration of a vehicle's path in situations like these – while respecting the motion constraints - has been addressed by various researchers. A common approach is to modify the entire pre-planned path based on current range-sensor data [9], [10].

In this paper, a reactive trajectory tracking controller based on nonlinear model predictive control is presented, along with an iterative algorithm for its real-time implementation. Given a desired trajectory, the controller minimizes the total tracking error along an entire future motion segment, based on the vehicle's motion equations. Pure pursuit [1] and path deformation [10] are special cases of the proposed controller when an appropriate cost function or optimizing horizon is used, respectively. In the presence of obstacles, the controller deviates from the reference

trajectory by incorporating into the optimization obstacle-distance information from range sensors. Simulations were performed to test the algorithm's performance for different motions and real-time computing scenarios.

Section II gives a brief review of non linear model predictive control. Section III describes the development of a high-level path tracking controller based on non linear model predictive control. Next, section IV describes in detail the numerical optimization procedure. In section V sensor range data are incorporated into the numerical optimization resulting in reactive tracking behavior. Section VI presents experimental simulation results and finally, section VII concludes the paper and suggests possible directions for future work.

II. Non Linear Model Predictive Control

A brief description of the non linear model predictive control methodology is given next [11]. Let a system's discrete state equation be of the form:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), k \geq 0 \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbb{R}^m$. The state and control vectors are subject to constraints of the form:

$$\mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}. \tag{2}$$

Let $\mathbf{x}_k^d, k = 0,1,...N$ be a desired state trajectory and $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_k^d$ be the error between the actual and the desired trajectories. The basic idea behind nonlinear model predictive control is to solve at every time step k a finite-horizon optimal control problem. Given the actual state $\mathbf{x}_k$, an optimal M-step-ahead control sequence $\underline{\mathbf{u}}_{k,M}^*$ is computed, which minimizes a cost function J:

$$\underline{\mathbf{u}}_{k,M}^* = [\mathbf{u}_k^* \ \mathbf{u}_{k+1}^* ... \mathbf{u}_{k+M}^*] = \underset{\mathbf{u}_k, \mathbf{u}_{k+1}, ..., \mathbf{u}_{k+M}}{\arg\min} J \tag{3}$$

The cost function has the following form:

$$J = \theta(\mathbf{e}_{k+M+1}) + \sum_{j=k}^{\min(N, k+M)} \psi(\mathbf{x}_j, \mathbf{u}_j) \tag{4}$$

where the tracking error of the last state of the finite horizon is penalized by a function θ which typically has a quadratic form:

$$\theta(\mathbf{e}_{k+M+1}) = \mathbf{e}_{k+M+1}^T \mathbf{Q}_e \mathbf{e}_{k+M+1}^T \tag{5}$$

At each step, the state, tracking error and control effort along the optimizing horizon can be penalized also by a quadratic form:

$$\psi(\mathbf{x}_j, \mathbf{u}_j) = \mathbf{x}_j^T \mathbf{Q}_x \mathbf{x}_j + \mathbf{e}_j^T \mathbf{Q}_e \mathbf{e}_j^T + \mathbf{u}_j^T \mathbf{Q}_u \mathbf{u}_j \tag{6}$$

At sample k+1 the system will have moved to a new state $\mathbf{x}_{k+1}$ which differs from the predicted state due to disturbances and model errors. The optimization problem is solved again, until the system reaches its goal state $\mathbf{x}_N^d$. Given that an optimal feasible open-loop control and trajectory for the problem exists, and that all the cost matrices $\mathbf{Q}$ are positive-definite, the stability of the closed-loop non linear model predictive controller can be proved [12].

III. Path Tracking

In this work we consider the motion of a non-holonomic car-like robot in the plane (Fig. 1). The x and y coordinates give the position of the car's rear wheel axle midpoint $\mathbf{P} = [x\ y]^T$. The unit vector $\mathbf{v}$ has its origin at $\mathbf{P}$ and lies along the direction of motion. The robot's orientation is given by θ, the angle between the positive x-axis and $\mathbf{v}$. The vehicle's wheelbase is L.

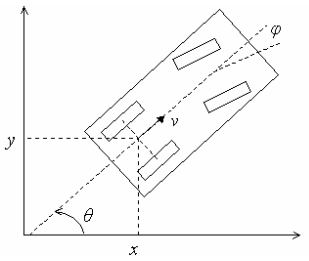

Fig. 1 Car-like robot model

A simple kinematical model for this front-wheel steered robot in the plane is the following:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= v\frac{\tan\phi}{L} \end{aligned} \tag{7}$$

The proposed approach to path tracking is to use non linear model predictive control as a high-level tracking controller (NMPT), which uses a simple kinematical model to predict vehicle motion, and a dynamic model for steering and speed control (Fig. 2). The control variables are v and φ, i.e., the desired linear velocity and steering angle. This approach avoids the complexities of full dynamic modeling, while retaining the required accuracy for low working-speeds.

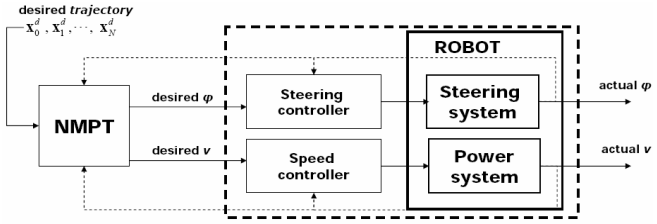

Fig. 2 High level NMPT controller

Of course, complex dynamic models could be used for the NMPT, but still, unknown model parameters (e.g., related to tire-soil interaction) need to be identified. At a lower-level, a steering controller model can be used to create an "abstraction" of the steering system dynamics [15]; the same holds for speed controllers. In this paper, the combination of the steering controller and steering linkage is modeled as a 1st order system with time-lag τ_φ; an analogous abstraction is used for speed control.

$$\begin{aligned}\dot{\phi} &= -\frac{1}{\tau_\phi}\phi + \frac{1}{\tau_\phi}u_\phi \\ \dot{v} &= -\frac{1}{\tau_v}v + \frac{1}{\tau_v}u_v\end{aligned} \tag{8}$$

This means that the NMPT provides a desired steering angle u_φ to the steering controller and the actual wheels' steering angle follows the command with first order dynamics. Similarly, velocity commands u_v are issued by the NMPT to the speed controller. Of course, different models (e.g. second order, time-delay) could be used, as long as their parameters can be identified. From the NMPT point of view the vehicle is described by (7), (8) the system state is $\mathbf{x} = [x\ y\ \theta\ \phi\ v]^{\mathrm{T}}$ and the control is $\mathbf{u} = [u_v\ u_\phi]^{\mathrm{T}}$. The tracking error is defined as the difference between the desired and actual trajectory points at each controller sample. The cost function is defined by (4), (5), and (6). Note that if the cost ψ is everywhere zero, NMPT is equivalent to pure-pursuit [1], whereas if M is big enough to include the final state, the NMPT will compute a deformation for the entire future path [10]. Also, if only position and orientation errors are penalized, NMPT is equivalent to geometrical path tracking. Next, a numerical procedure for solving the NMPT will be presented.

IV. Numerical Solution

In the general case, non linear model predictive control optimization can only be solved numerically. The system equations must be discretized so that they can be expressed in the form of (1) and solved. In this paper the indirect approach was used, which uses gradient descent in order to minimize the problem's Hamiltonian [13].

The Hamiltonian of the optimal control problem is

$$H = \psi(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\lambda}_{k+1}^T \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{9}$$

where $\boldsymbol{\lambda}_k$ is the costate sequence. It can be shown [13], [14] that for fixed initial state, the first-order variation of the cost function is given by:

$$\begin{aligned}\delta J = &\left[\frac{\partial\theta}{\partial\mathbf{x}_N} - \boldsymbol{\lambda}_N\right]\delta\mathbf{x}_N + \\ &\sum_{k=0}^{N}\left\{\left[\frac{\partial H}{\partial\mathbf{x}_k} - \boldsymbol{\lambda}_k\right]^T\delta\mathbf{x}_k + \left[\frac{\partial H}{\partial\mathbf{u}_k}\right]^T\delta\mathbf{u}_k\right\}.\end{aligned} \tag{10}$$

If the costate sequence satisfies the equation

$$\boldsymbol{\lambda}_j = \frac{\partial H}{\mathbf{x}_j} = \frac{\partial\psi}{\partial\mathbf{x}_j} + \frac{\partial\mathbf{f}^T}{\partial\mathbf{x}_j}\boldsymbol{\lambda}_{j+1},\ j = k,\ldots,k+M-1 \tag{11}$$

with terminal condition

$$\boldsymbol{\lambda}_{k+M} = \frac{\partial\theta}{\partial\mathbf{x}_{k+M}} \tag{12}$$

the cost variation becomes:

$$\delta J = \sum_{k=0}^{N}\left[\frac{\partial H}{\partial\mathbf{u}_k}\right]^T\delta\mathbf{u}_k \tag{13}$$

where

$$\frac{\partial H}{\partial\mathbf{u}_j} = \frac{\partial\psi}{\partial\mathbf{u}_j} + \frac{\partial\mathbf{f}^T}{\partial\mathbf{u}_j}\boldsymbol{\lambda}_{j+1} \tag{14}$$

The main idea behind a gradient descent solution algorithm is the following: if we start from a nominal solution $\underline{\mathbf{u}}_{k,M}^*$ which is close to the optimum, in order to minimize J, a variation $\delta\underline{\mathbf{u}}_{k,M}^*$ of this control must be computed, such that the variation δJ should be always negative. This can be achieved by moving the control in the opposite direction of the Hamiltonian's control gradient (steepest descent).

$$\mathbf{u}_j^{i+1} = \mathbf{u}_j^i - K\frac{\partial H}{\partial\mathbf{u}_j^i},\ j = k,\ldots k+M \tag{15}$$

where the gain K is a sufficiently small positive number.

The algorithm proceeds as follows: at each step k, given an initial $\underline{\mathbf{u}}_{k,M}^*$ the gradient descent iteration index i is initialized to zero and (7),(8) are used to compute the open-loop trajectory $\underline{\mathbf{x}}_{k,M}^* = [\mathbf{x}_k^*\ \mathbf{x}_{k+1}^*\ldots\mathbf{x}_{k+M+1}^*]$. Next, $\boldsymbol{\lambda}_j^i$ and $\partial\mathrm{H}/\mathbf{u}_j$ are computed using (11), (12) and (14) respectively, and the updated controls $\mathbf{u}_j^{i+1}$ are computed by (15). Finally, i is incremented and the iterations continue, until a convergence termination criterion is satisfied.

V. Reactive Tracking in the Presence of Obstacles

Typically, the desired motion trajectory is computed by some motion planning algorithm based on a map of the environment and is assumed to be collision-free. During the actual motion execution it is possible that obstacles appear in the vehicle's path, which had not been present in the planning phase (e.g., animals, humans, machines). This may also happen because of imprecision in the field map, or vehicle localization errors. If the robot is equipped with range sensors, e.g., a laser scanner, then at any time the distances of a number of obstacle points, which had not been present in the original environment map from the respective sensor are known. These distances can be incorporated in the cost function, so that the computed control tracks the desired trajectory, while staying away from the obstacles.

Let T_{sr} be the fixed transformation matrix of the rth sensor frame with respect to the vehicle frame, and $T(\mathbf{x})$ be the transformation matrix of the vehicle's frame with respect to the fixed navigation frame. Let ${}^{S_r}\mathbf{r}_m(\mathbf{x})$ be the mth range measurement of the rth sensor in the sensor frame, at robot state $\mathbf{x}$. The corresponding obstacle-point position expressed in the navigation frame is $\mathbf{p}_{rm} = T(\mathbf{x})T_{s_r}\,{}^{S_r}\mathbf{r}_m(\mathbf{x})$ and is independent of the robot state. The rth sensor's frame origin expressed in the navigation frame is $\mathbf{s}_r(\mathbf{x}) = T(\mathbf{x})T_{s_r}\,{}^{S_r}\mathbf{s}$, where ${}^{S_r}\mathbf{s} = [0\,0\,1]^T$. Hence, their relative distance vector, expressed in the navigation frame is:

$$\mathbf{a}_{rm}(\mathbf{x}) = \mathbf{p}_{rm} - \mathbf{s}_r(\mathbf{x}) \tag{16}$$

and the scalar distance is $D_{rm}(\mathbf{x}) = (\mathbf{a}_{rm}^T\mathbf{a}_{rm})^{1/2}$. The obstacle points which do not correspond to obstacles in the map contribute to the cost functions θ and ψ with a term which penalizes states which result in proximity with the obstacles:

$$V(\mathbf{x}) = \sum_{r=1}^{R}\sum_{m=1}^{M_r}\left(\frac{C}{D_{rm}(\mathbf{x}) + \varepsilon}\right)^{\rho} \tag{17}$$

where R is the number of range sensors, M_r is the number of readings of the rth sensor, ε is a small positive real number, $C>0$ and $\rho>1$. In order to solve the system of (11), (12) the terms $\partial\psi/\partial\mathbf{x}_j$, $\partial\theta/\partial\mathbf{x}_j$ are needed. Thus, we need to compute $dV_{rm}/d\mathbf{x}_j$, which can be shown to be

$$\frac{dV_{rm}}{d\mathbf{x}_j} = \frac{-\rho C^{\rho}}{(D_{rm}(\mathbf{x}_j) + \varepsilon)^{\rho+1}}\frac{dD_{rm}}{d\mathbf{x}_j}, \tag{18}$$

where

$$\frac{dD_{rm}}{d\mathbf{x}_j} = \left(\frac{d\mathbf{a}_{rm}}{d\mathbf{x}_j}\right)\frac{dD_{rm}}{d\mathbf{a}_{rm}} = \frac{1}{D_{rm}}\left(\frac{d\mathbf{a}_{rm}}{d\mathbf{x}_j}\right)\mathbf{a}_{rm} \tag{19}$$

Using (16) and noticing that the obstacle-point $\mathbf{p}_{rm}$ is independent of the robot state, the term $d\mathbf{a}_{rm}/d\mathbf{x}_j$ is found to be equal to $-d\mathbf{s}_r/d\mathbf{x}_j$.

VI. Simulation Results

The NMPT was implemented in C++ and numerous simulations were performed. The vehicle's wheelbase L was taken equal to 2 m, with φ_{max}=60° and v_{max}= 1 m/s and the sampling period dt=100 ms. The steering system time-lag was set equal to $\tau_\phi = 0.15\,s$, and the velocity time-lag to $\tau_v = 1\,s$. All the elements of all cost matrices were set to zero, except for $q_e(1,1)= q_e(2,2)$=500 (distance error costs) and $q_e(3,3)$=100 (orientation error cost). The velocity errors were penalized less, with $q_e(5,5)$=50, because path accuracy was considered more important. The NMPT gradient descent algorithm was allowed to execute for 1000 iterations; larger iteration-limits did not improve the solutions.

In a first experiment, the tracking controller's step response was tested on a 300-point square path with sharp 90° orientation discontinuities. An initial horizontal position error of 0.25 m was introduced in the robot's position. At the corners, the robot's desired orientation switches from π/2 to 0 and then to 3π/2. The nominal velocity was equal to 0.5 m/s. The response of the NMPT was compared against that of a Pure Pursuit Control (PPC) type tracking algorithm, appropriately modified for non-holonomic vehicles. The PPC was assumed again to be a high-level controller, issuing desired velocity and steering angle commands to the low-level controllers. If the position errors between the current tractor position and the look-ahead point expressed in the tractor frame are e_x and e_y, the orientation error e_θ, and the velocity error e_v the pure-pursuit controller is given by:

$$\begin{aligned} u_v &= K_v e_v + K_x e_x \\ u_\phi &= \mathrm{sgn}(u_v)\tan^{-1}\left(\frac{(K_\theta e_\theta + K_y e_y)L}{v}\right) \end{aligned} \tag{20}$$

The term $K_v e_v$ in u_v is for velocity tracking and the term $K_x e_x$ makes sure that the look-ahead point will be reached even if the desired velocity at that point is zero. The term $K_\theta e_\theta + K_y e_y$ is a desired rotational velocity term $\dot{\theta}_d$ and the desired steering angle is solved for it from (7). The error e_y is contained in the desired rotational velocity because non-holonomic vehicles have to turn in order to move laterally. The term $\mathrm{sgn}(u_v)$ corrects the steering during reverse motion. Finally, the K parameters are the gains of the controller. The PPC was assumed to be executing at the same sampling rate as the NMPT and the look-ahead distance it used was optimal.

In order to select the optimal PPC look-ahead point for this path, different values for the number of look-ahead points (M) on the path were tried for the PPC-based motion

simulations. The minimum path tracking error was achieved at a distance of 1.25*m* (*M*=25). This result can be explained by the minimum turning radius, which is 1.15 m, and which translates to 23 path points, at a moving speed of about 0.5*m/s*. Hence, distances significantly different from 1.15*m* would result in large overshoots, or corner-cutting. The PPC gains were selected to minimize overshoot, via trial and error. The PPC gains were tuned to provide critical damping with a look-ahead distance of 1.25m (*M*=25).

The PPC simulation resulted in average and maximum path tracking errors of 4.62cm and 34.94cm respectively. The total area between the desired and executed paths was approximately 0.73m^2. The NMPT simulation was executed with the optimization horizon parameter *M* ranging from 10 to 80 points. The paths for *M*=20 and 60 points are shown in Fig. 3.

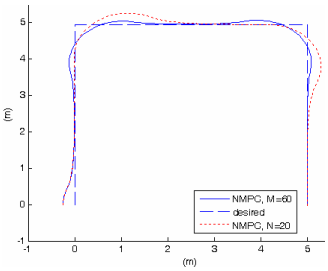

Fig. 3 NMPT tracking of a square path (M=20 and M=60 points)

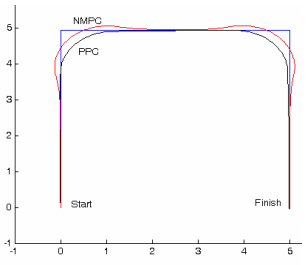

Fig. 4 Tracking with pure-pursuit and NMPT controllers (M=60)

Each NMPT computation required 0.07s on a Pentium 2.6 GHz single-CPU system. When NMPT was executed with *M*=25 it performed worse than the PPC. The reason was that NMPT didn't turn well in advance before the sharp turns. This is expected, since NMPT minimizes the tracking error over its entire horizon, whereas PPC turns *M* points in advance. However, when M was increased to *M*=60 (5 meters), NMPT performed much better (Fig. 4), with average and maximum tracking errors equal to 4.2cm and 19.78cm respectively. The total error-area was approximately 0.65m^2. The NMPT accomplished this by performing an anticipatory turning motion which increased the instantaneous tracking error, but lead to smaller total error. Such motions cannot be performed by PID-type controllers. The NMPT convergence time increased to 0.4s, but speed-problems could be overcome with dedicated hardware. It should be mentioned that part of the tracking errors for both tracking controllers are due to the first order approximation for the steering and velocity systems. A pure kinematics model would result in smaller errors.

In a second simulation experiment the robot followed a key-hole shaped path at a constant speed of 0.5 m/s. The desired and executed paths for two different starting positions are shown in Fig. 5. Both starting positions contained a ±0.5 m horizontal initial error, whereas the second position had also a 20^o orientation error. Both trajectories converged to the desired one despite the relatively large initial errors. The average and maximum trajectory tracking errors after convergence were 0.96cm and 1.36cm respectively, i.e., the NMPT followed the trajectory very closely. The average and maximum velocity tracking errors after convergence were 0.48% and 2.02% respectively. Each new control computation required 13.1ms on a Pentium 2.6 GHz single-CPU system, which is adequate for real-time implementation.

Next, a rectangular obstacle was introduced in a position which prohibited exact path following and the simulation was executed again. As it is shown in Fig. 6, the NMPT deviated from the desired trajectory in order to safely avoid the obstacle. The vehicle passed from the right side of the obstacle because of its limited turning radius; passing it from the left would have required a turning angle of more that 60^o.

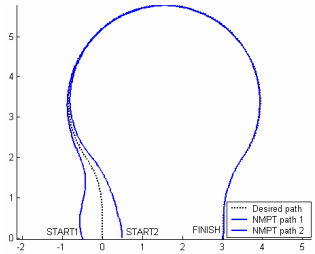

Fig. 5 Key-hole maneuver

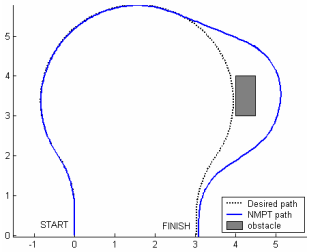

Fig. 6 Deformation due to obstacle

In a third simulation experiment a fish-tail maneuver was executed, which contained large discontinuities in both orientation and velocity. Fig. 7 shows three different executed paths for two different starting positions. The first path resulted from an initial horizontal position error of +0.5 m. The second and third paths resulted from an initial error of -0.5m and an orientation error of 20^o. The second path converged more slowly to the desired path than the third path, because the orientation penalty term in the $\mathbf{Q}_e$ matrix was $q_e(3,3)$=1500, whereas for the third path a smaller value of $q_e(3,3)$=100 was used. Hence, a position vs. orientation error trade-off existed. In Fig. 8 the deviation of the NMPT from the desired trajectory is shown in the presence of an unexpected obstacle.

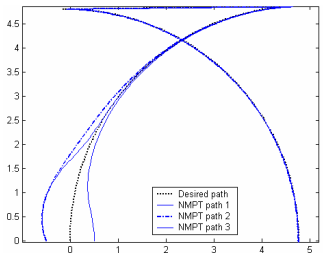

Fig. 7 Fish-tail maneuver

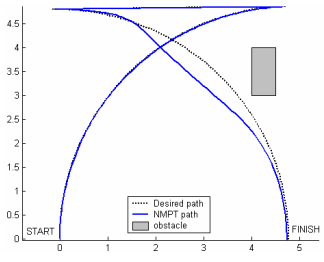

Fig. 8 Deformation due to obstacle

VII. Conclusions and Future Work

The NMPT consistently converged to the desired trajectories and followed them accurately, despite large

initial errors and discontinuities in the desired velocity and orientation.

Overall, the computational cost does not seem to be prohibitive for real-time control. The optimization horizon M is an important parameter, which regulates a trade-off between timely obstacle avoidance and tracking quality (large M) vs. fast solution time (small M). Also, in the presence of obstacles the convergence rate of the gradient descent numerical procedure varied along the path.

The delayed computation of "fresh" NMPT controls, even if it happens intermittently, can create problems during real-time control. If the NMPT solution time is λ times longer than dt, then at the next λ steps the optimal control computed at step k must be used. This is equivalent to open-loop control and the tracker's stability for large λ needs to be investigated further. A partial solution to the speed problem could be the use of more advanced numerical techniques for the NMPT optimization. For example, a direct approach, such as *direct transcription* could be used, which casts the optimal control problem into a constrained Nonlinear Programming Problem (NLP). Such problems can be solved very efficiently using sparse Sequential Quadratic Programming [16].

Another issue is the choice of the cost matrix elements. From our simulations it was clear that they affect tracking quality as well as the shape of the path. This is due to trade-offs among position, orientation, and velocity errors, and appropriate values for different types of motions and missions need to be found.

A characteristic of trajectory tracking, which came up during simulations is that obstacle avoidance may lead to significantly longer paths. Given that the desired trajectory is defined by N points and that the NMPT sampling interval dt is fixed, longer paths can be traversed with the same N and dt parameters, only if the vehicle's velocity is increased; this may lead to increased velocity tracking errors. More importantly, if the path is too long (e.g., N_p points, with $N_p >> N$), the maximum velocity cannot be exceeded and the executed trajectory will not terminate at the goal. A solution to this problem could be the asynchronous advancement of the executed and desired trajectory indices, so that at the end of the motion (after N_p points) the vehicle's state is $\mathbf{x}_{N_p} = \mathbf{x}_N^d$. Overall, in theory, NMPT offers tracking capabilities for wheeled vehicles, which linear controllers cannot match [4]. Hence, it seems to offer a promising approach for mobile robot path tracking applications, and it deserves further investigation.

REFERENCES

[1] O. Amidi, "Integrated Mobile Robot Control", M.S. thesis, Dept. of Electrical and Computer Engineering, CMU, Pittsburgh, PA, 1990.

[2] A. Balluchi,, A. Bicchi,, A. Balestrino, and G. Casalino, "Path Tracking Control for Dubin's Car", *Proc. IEEE Int. Conf. Robot. & Autom.*, Minneapolis, MN, pp. 3123-3128, 1996.

[3] J.-M. Yang, I.-H. Choi, and J.-H. Kim, "Sliding Mode Control of a Nonholonomic Wheeled Mobile Robot for Trajectory Tracking", *Proc. IEEE Int. Conf. Robot. & Autom.*, Leuven, Belgium, pp. 2983-2988, 1998.

[4] Y. Zhang, S. Velinsky, and X. Feng, "On the Tracking Control of Differentially Steered Wheeled Mobile Robots", *Journal of Dynamic Systems, Measurement, and Control*, pp. 455-466, 1997.

[5] J. Witt, C.D.III Crane, and D. Armstrong, "Autonomous Ground Vehicle Path Tracking", *Journal of Robotic Systems*, Vol.21, No. 8, pp. 439–449, 2004.

[6] R. Brockett, "Asymptotic stability and feedback stabilization", In *Differential Geometric Control Theory*, pp. 181-208, Birkhauser, 1983.

[7] A. Divelbiss, and J. Wen, "Trajectory Tracking Control of a Car-Trailer System", *IEEE Transactions on Control Systems Technology*, Vol.5, No. 3, pp. 269-278, 1997.

[8] J. Wen, and J. Sooyong, "Nonlinear Model Predictive Control based on Predicted State Error Convergence", Proceeding of the 2004 American Control Conference, Boston, MA, pp. 2227-2232.

[9] O. Brock and O. Khatib, "Elastic strips: Real-time path modification for mobile manipulation". *Robotics Research*, pp. 5–13, Springer- Verlag, 1998.

[10] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive Path Deformation for Nonholonomic Mobile Robots", *IEEE Transactions on Robotics*, Vol.20, No.6, pp. 967-977, 2004.

[11] B. Kouvaritakis, and M. Cannon, *Non-Linear Predictive Control: Theory and Practice*, IEE Publishing, London, 2001.

[12] H. Chen, and F. Allgoewer, "Nonlinear model predictive control schemes with guaranteed stability", *Nonlinear Model Based Process Control*, R. Berber and C. Kravaris, Eds. Dodrecht: Kluwer Academic Publishers, 1998, pp. 465–494.

[13] D.E. Kirk. *Optimal Control Theory: An Introduction*, Prentice Hall, 1970.

[14] A.P. Sage and C.C. White, III, "Optimum Systems Control", 2nd ed., Prentice Hall, 1977.

[15] D. Wu, Q. Zhang, J. Reid, H. Qiu, and E. Benson, "Model Recognition and Simulation of an E:H Steering Controller on Off-Road Equipment", *Fluid Power Systems and Technology*, S. Nair, S. Mistry, Eds. New York: ASME, 1998, pp. 55–60.

[16] J. Betts, "Survey of Numerical Methods for Trajectory Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, pp. 193-207, 1998.