Nonlinear Model Predictive Control for Omnidirectional Robot Motion Planning and Tracking With Avoidance of Moving Obstacles

Model de commandes prédictives non linéaire pour la planification et le suivi de mouvements d’un robot omnidirectionnel pour éviter des obstacles mobiles

Timothy A. V. Teatro, *Student Member, IEEE*, J. Mikael Eklund, *Senior Member, IEEE*, and Ruth Milman, *Member, IEEE*

***Abstract*— This paper presents a nonlinear model predictive control algorithm for online motion planning and tracking of an omnidirectional autonomous robot. The formalism is based on a Hamiltonian minimization to optimize a control path as evaluated by a cost function. This minimization is constrained by a nonlinear plant model, which confines the solution space to those paths which are physically feasible. The cost function penalizes tracking error, control amplitude, and the presence in a potential field cast by moving obstacles and Boards. An experiment is presented demonstrating the successful navigation of a field of stationary obstacles. Simulations are presented demonstrating that the algorithm enables the robot to react dynamically to moving obstacles.**

***Résumé*— Cet article présente un algorithme de commande prédictive de modèle non linéaire pour la planification de mouvement et le suivi en ligne d’un robot autonome omnidirectionnel. Le formalisme est basé sur une minimisation Hamiltonienne pour optimiser un trajet de contrôle tel qu’évalué par une fonction de coût. Cette minimisation est limitee par un modèle des installations non linéaire limitant ainsi l’espace de solution à des chemins physiquement réalisables. La fonction de coût pénalise les erreurs de suivi, l’amplitude de contrôle, et la présence d’un objet dans un champ potentiel en déplaçant les obstacles et les bords. Les résultats expérimentaux montrent une navigation réussie d’un regroupement d’obstacles stationnaires. Les simulation démontrent que l’algorithme permet au robot de réagir de façon dynamique à des obstacles mobiles.**

***Index Terms*—Mobile robot motion-planning, predictive control.**

I. Introduction

IN THE past four decades, many methods have been developed for planning and controlling robot motion. Constrained by the technology of their time, many of these techniques were suited toward robots with basic sensors and limited computing power. As technology has progressed, so has the methodologies. Now, there are numerous planning routines which incorporate rich sensor data with elegant and computationally demanding procedures.

Manuscript received November 30, 2013; revised April 26, 2014; accepted May 7, 2014. Date of current version December 4, 2014. This work was supported by the NSERCs Engage Program.

The authors are with the Department of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada (e-mail: timothy.teatro@uoit.ca; mikael.eklund@uoit.ca; ruth.milman@uoit.ca).

Associate Editor managing this paper’s review: Sidney Givigi.
Digital Object Identifier 10.1109/CJECE.2014.2328973

Model predictive control (MPC) was originally developed for process control [1], [2]. The method involves using a mathematical model to predict the consequences of a control plan over a finite horizon. That model is used in conjunction with optimal control methods to find an optimal control plan for the preview horizon. These plans are generated as part of a seemingly open loop strategy. The loop is closed when information from the plant’s state estimator is used to seed the forecasting model.

The optimization is performed online, over short time segments yielding short control plans. The global state trajectory is formed by executing subsets of each small plan. The method can be applied to nonlinear models directly, and is then referred to as nonlinear model predictive control (NMPC) [1], [3], [4].

This paper follows similar applications of NMPC/MPC to tracking and planning of other vehicles. Fahimi [5] applied

NMPC to formation control for groups of autonomous surface vehicles. Eklund *et al.* [6] applied NMPC to aircraft in pursuit/evasion games. Klančar and Škrjanc [7] applied MPC to a two wheeled robot using linearized tracking-error dynamics. More to the point of this paper, it has been applied to several sorts of wheeled terrestrial robots [8], [9]. The problem of tracking a path with NMPC is studied in [10] and sufficient conditions for recursive feasibility and asymptotic stability are presented. Hsieh and Liu [11] used a scalar potential field cost terms to avoid obstacles.

This paper outlines the development of a simple NMPC path planner and tracker as it is applied to an omnidirectional vehicle (robot) with steering rate control. The underlying mathematical framework is based on the solution of the Euler–Lagrange equations for the system to find the stationary points of a Hamiltonian. Algorithms for both the minimization and a governing NMPC controller are explicitly described and are shown to successfully route a robot through a field of stationary obstacles. In simulations, it is shown that the controller can also handle moving obstacles.

II. Vehicle Dynamics

For this paper, a simple robot is modeled wherein only position, speed, steering angle, and steering rate are considered. The steering angle always represents the direction of travel (precluding the possibility of slipping) and the speed is taken to be constant at v. The direction of travel is controlled in time by varying the steering rate. The discrete-time state vector ($\boldsymbol{q}_k$) and control input (u_k) are

$$\boldsymbol{q}_k = [x_k \quad \dot{x}_k \quad y_k \quad \dot{y}_k \quad \theta_k]^\mathsf{T} \quad \text{and} \quad u_k = \dot{\theta}_k \tag{1}$$

which express the rectangular x and y coordinates with corresponding speed components $\dot{x}$ and $\dot{y}$, along with steering angle θ and steering rate $\dot{\theta}$ at time-step k. The time-step index k is related to real time t as $t = kT$ with interval T.

Integrating the equations of motion with a first-order Euler scheme yields the discrete equations approximating the dynamics of the vehicle as

$$\boldsymbol{q}_{k+1} = \boldsymbol{f}(\boldsymbol{q}_k, u_k) = \begin{bmatrix} x_k + \dot{x}_k T \\ v\cos\theta_{k+1} \\ y_k + \dot{y}_k T \\ v\sin\theta_{k+1} \\ \theta_k + u_k T \end{bmatrix}. \tag{2}$$

III. NMPC Controller Design

The NMPC controller problem is to repetitively solve the discrete finite horizon open loop optimization

$$u^* = \underset{\{u_1, u_2, \ldots u_N\}}{\arg\min} \; J(\boldsymbol{q}, u) \tag{3}$$

wherein, the designed cost function $J(\boldsymbol{q}, u)$ is minimized with respect to the control input $u = \{u_1, u_2, \ldots, u_N\}$, over the N-step NMPC horizon, yielding the optimal control plan u^*. The control algorithm employs the model in (2) to obtain a forecast of the state trajectory from a given control plan. The forecast is continually seeded with the robot's internal reckoning of state, which has the effect of closing the loop of the open loop control problem, (3).

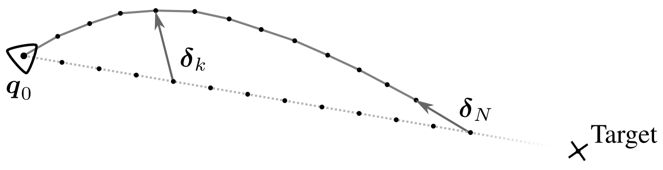

Fig. 1. Illustration of the airline tracking scheme. Tracking errors are the difference between the evenly sampled airline path (light gray dotted line) and the trajectory forecast from u (dark gray line). Tracking error vectors are indicated as dark gray arrows.

A quadratic cost function is chosen

$$\begin{aligned} J(\boldsymbol{q}, u) &= \boldsymbol{\delta}_N^\mathsf{T} \boldsymbol{Q}_0 \boldsymbol{\delta}_N + \sum_{k=1}^{N-1} \Big(R u_k^2 + \boldsymbol{\delta}_k^\mathsf{T} \boldsymbol{Q} \boldsymbol{\delta}_k + \Phi(x_k, y_k) \Big) \\ &= \phi(\boldsymbol{\delta}_N) + \sum_{k=1}^{N-1} L\big(\boldsymbol{q}_k, u_k\big). \end{aligned} \tag{4}$$

The tracking error $\boldsymbol{\delta}_k$, is penalized along with control effort. The running cost L contains the potential field Φ, which enforces our soft constraints associated with obstacle avoidance (to be discussed more in Section IV). The terminal cost $\phi(\boldsymbol{\delta}_N)$ encourages the end of the path ($\boldsymbol{q}_N$) to be close to $\boldsymbol{q}_N^{\text{ref}}$. The weighting matrices $\boldsymbol{Q}_0$, $\boldsymbol{Q}$, scalar R are chosen to reflect the desired balance of their respective cost terms.

The tracking error $\boldsymbol{\delta}_k = \boldsymbol{q}_k^{\text{ref}} - \boldsymbol{q}_k$ is the difference between the state vector and a time varying reference. The time reference is generated at each step by drawing a line in Euclidean space from $\boldsymbol{q}_0$ to the target (the airline path). That line is discretized into $N-1$ points separated by length vT. The corresponding points are used in sequence as $\boldsymbol{q}_k^{\text{ref}}$. This arrangement is shown in Fig. 1.

The weighting scalar R deserves a brief but special consideration in the present context, since it controls the level of holonomicity expressed in the control path. If this parameter is set to zero, the robot can make instantaneous changes in direction, since nothing else limits the turn radius of the vehicle. However, sudden changes in direction may pose practical problems, so setting R to a positive value will, in effect, add the turn radius into the cost balance. The value of R can even be a function of velocity so that narrower turns can be executed at lower speeds.

Following the path of similar research [5], [6], [12], we solve the minimization problem 3 by solving the Euler–Lagrange differential equations, using a variational expansion to construct a gradient descent scheme. The cost measure in (4) is augmented with a set of Lagrange multipliers $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$ enforcing (2) as a constraint

$$J_\Lambda = \phi(\boldsymbol{\delta}_N) + \sum_{k=1}^{N-1} \Big(L(\boldsymbol{q}_k, u_k) + \boldsymbol{p}_{k+1}^\mathsf{T} \left(\boldsymbol{f}(\boldsymbol{q}_k, u_k) - \boldsymbol{q}_{k+1} \right) \Big). \tag{5}$$

The state trajectory is computed from (2), so it will always be the case that $\boldsymbol{q}_{k+1} - \boldsymbol{f}(\boldsymbol{q}_k, u_k) = 0$. We may, therefore, freely choose the Lagrange multipliers without affecting the balance of (5).

Following Pontryagin's formalism [13, Ch. 5], we define the control Hamiltonian as:

$$\mathcal{H}_k = L(\boldsymbol{q}_k, u_k) + \boldsymbol{p}_{k+1}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{q}_k, u_k). \tag{6}$$

Combining (5) and (6), the augmented cost measure becomes

$$J_\Lambda = \phi(\boldsymbol{\delta}_N) + \boldsymbol{p}_N^{\mathsf{T}} + \mathcal{H}_0 + \sum_{k=1}^{N-1} \left(\mathcal{H}_k - \boldsymbol{p}_k^{\mathsf{T}} \boldsymbol{q}_k\right). \tag{7}$$

In order to minimize J_Λ, we compute the differential change in J_Λ with respect to its dependants

$$\mathrm{d}J_\Lambda = \left(\frac{\partial \phi}{\partial \boldsymbol{q}_N} - \boldsymbol{p}_N^{\mathsf{T}}\right) \mathrm{d}\boldsymbol{q}_N + \frac{\partial \mathcal{H}_0}{\partial \boldsymbol{q}_0} \mathrm{d}\boldsymbol{q}_0 + \frac{\partial \mathcal{H}_0}{\partial u_0} \mathrm{d}u_0 + \sum_{k=1}^{N-1} \left[\left(\frac{\partial \mathcal{H}_k}{\partial \boldsymbol{q}_k}\right) \mathrm{d}\boldsymbol{q}_k + \frac{\partial \mathcal{H}_k}{\partial u_k} \mathrm{d}u_k\right]. \tag{8}$$

Now in a position to make choices regarding our Lagrange multipliers so as to simplify (8) as much as possible, we choose

$$\boldsymbol{p}_N^{\mathsf{T}} = \frac{\partial \phi}{\partial \boldsymbol{q}_N} \quad \text{and} \quad \boldsymbol{p}_k^{\mathsf{T}} = \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{q}_k} + \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{\delta}_k} \frac{\partial \boldsymbol{\delta}_k}{\partial \boldsymbol{q}_k}$$

or more explicitly

$$\boldsymbol{p}_N^{\mathsf{T}} = \boldsymbol{\delta}_N^{\mathsf{T}} \, \boldsymbol{Q}_0 \, \frac{\partial \boldsymbol{\delta}_N}{\partial \boldsymbol{q}_N} \tag{9}$$

and

$$\boldsymbol{p}_k^{\mathsf{T}} = \boldsymbol{\delta}_k^{\mathsf{T}} \, \boldsymbol{Q} \, \frac{\mathrm{d}\boldsymbol{\delta}_k}{\mathrm{d}\boldsymbol{q}_k} + \boldsymbol{p}_{k+1}^{\mathsf{T}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}_k} + \frac{\partial \Phi_k}{\partial \boldsymbol{q}_k} \tag{10}$$

with Φ_k as the scalar potential at step k. The constant factors from the differentiation are absorbed into the weighting parameters. The choice of Lagrange multipliers consequently reduces (8) to

$$\mathrm{d}J_\Lambda = \sum_{k=1}^{N-1} \frac{\partial \mathcal{H}_k}{\partial u_k} \mathrm{d}u_k + \boldsymbol{p}_0^{\mathsf{T}} \mathrm{d}\boldsymbol{q}_0. \tag{11}$$

The right-most term of the above equation is zero, since $\boldsymbol{q}_0$ is constant and $\mathrm{d}\boldsymbol{q}_0$ is therefore also zero. The final effort is in evaluating $\partial\mathcal{H}_k/\partial u_k$ by expanding and differentiating (6), which reveals that

$$\frac{\partial \mathcal{H}_k}{\partial u_k} = R u_k + \boldsymbol{p}_{k+1}^{\mathsf{T}} \frac{\partial \boldsymbol{f}}{\partial u_k} \tag{12}$$

giving us the tools to compute the gradient elements of $\mathcal{H}_k$, and therefore to iteratively minimize the cost function with respect to each u_k.

A basic gradient descent algorithm implementing this formalism is listed in Algorithm 1.

Starting from some initial condition $\boldsymbol{q}_0$ and an initial guess for the optimal control history u, one can use the derived formalism to iteratively improve u to approach u^*. The relationships developed in this section are used to find the gradient elements $\partial\mathcal{H}_k/\partial u_k$, which are then used to step u toward u^*. Convergence rate and stability are balanced (in part) by the step factor Δ. The iteration loop is broken when the Euclidean norm of the vector formed by the gradient elements is sufficiently small. This algorithm is by no means the best way to implement the minimization, but it may be the most straightforward. Compared with more sophisticated methods, it sacrifices elegance and performance for intuition as it illustrates the basic relationships that may lead to an optimal control history.

Algorithm 1 General Algorithm for Iteratively Optimizing a Control History Against a Cost Function

```
$u \leftarrow$ initial guess
$\boldsymbol{q}_0 \leftarrow$ starting position
**while** norm$([\partial\mathcal{H}_1/\partial u_1, \cdots, \partial\mathcal{H}_N/\partial u_N])$ > *tolerance* **do**
  **for** $k = 1, \ldots, (N-1)$ **do**
    $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{f}(\boldsymbol{q}_k, u_k)$
  **end for**
  Compute $\boldsymbol{p}_N$ from Eq. (9)
  **for** $k = (N-1), \ldots, 1$ **do**
    Compute $\boldsymbol{p}_k$ from Eq. (10)
  **end for**
  **for** $k = 1, \ldots, (N-1)$ **do**
    Compute gradient element $\partial\mathcal{H}_k/\partial u_k$ from Eq. (12)
  **end for**
  **for** k=1, …, N **do**
    $u \leftarrow u - \Delta\left(\partial\mathcal{H}_k/\partial u_k\right)$
  **end for**
**end while**
```

Algorithm 2 General Statement of the MPC Algorithm

```
$u \leftarrow$ initial guess
$\boldsymbol{q}_0 \leftarrow$ starting position
**while** not sufficiently close to target **do**
  Get $u^*$ and $\boldsymbol{q}^*$ over $N$ using Algorithm 1
  Execute the first $C$ steps of $u^*$
  **for** $k = C, \ldots, N$ **do**
    $\boldsymbol{q}_{k-C} \leftarrow \boldsymbol{q}_k^*$
    $u_{k-C} \leftarrow u_k^*$
  **end for**
**end while**
```

An NMPC algorithm based on Algorithm 1 is listed in Algorithm 2. Speaking generally, NMPC limits the scope of the optimization problem to N time-steps, the prediction horizon. The optimization is carried out for a segment of the path to the target that is N steps long. A subset of that optimal path, the first C steps are executed, and the optimization is carried out again for the next set of N steps. This process is repeated until the target is reached. The integer C is the control horizon. The general configuration of an NMPC controller is illustrated in the block diagram in Fig. 2.

In that figure, the global path planner takes the target and plans a route to it. That part of the diagram is typically offline. The replanner, however, runs online and adapts the path dynamically to new information. The desired path is passed into the MPC block which consists of an optimizer and system model which pass information back and forth

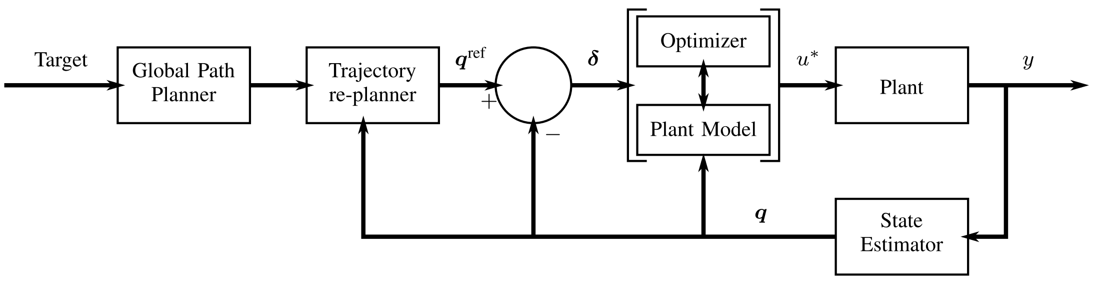

Fig. 2. Block diagram for a system with NMPC controller.

until it has computed the optimal path u^*. A subset of u^* is executed by the plant (the mobile robot) and the resulting pose of the system and other output information is available in y. Whatever routines and devices are needed to convert y into the state vector $\boldsymbol{q}$ is encompassed in the state estimator. The state estimator passes the state vector to the blocks that require it.

IV. Avoidance of (Moving) Obstacles

In this paper, avoidance of obstacles and walls is achieved by modulating the potential field $\Phi(x_k, y_k)$. The potential field is accumulated in the running cost terms of the cost function. Minimization of the cost will yield paths which tend to minimize exposure to positive potential fields. Conversely, negative potential fields may be used as attractors.

This potential model affords one a great deal of flexibility in constructing the potential fields—the only requirement being that the potential field is differentiable. In this paper, potential fields are constructed as a sum of inverse-square point potentials over the set of obstacles $\mathcal{O}$ (which may include walls)

$$\Phi(x, y) = \sum_{i \in \mathcal{O}} \frac{1}{(x - X_i)^2 + (y - Y_i)^2 + \varepsilon}. \tag{13}$$

The term ε avoids the singularity and allows prescription of the maximum potential as $1/\varepsilon$.

The potential field of each wall is computed by finding the point on each wall closest to (x, y), and placing the point obstacle there. This is done by first checking to see if either of the end points are the closest points. Otherwise, the vector connecting the first endpoint to the the coordinate (x, y) is projected onto the vector between the two endpoints of the wall yielding the point on the wall closest to the coordinate (x, y). In the results, the reader may notice the superposition of potential fields where two line segments meet to form a corner. This can be avoided, but posed no significant problem in the present research.

A. Moving Obstacles

A primitive moving obstacle avoidance can be managed simply by updating the positions of obstacles between optimization (NMPC) steps. If the speed of the obstacles is sufficiently small compared with the speed of the robot, the motion of the two can be safely decoupled in the optimization step. If the speed of the obstacle is not considerably slower than the speed of the robot, then the robot may be hit, if it enters the path of a moving obstacle. There is also a risk to the stability of the optimization step if an obstacle moves suddenly to a point close to the robot in its path of motion.

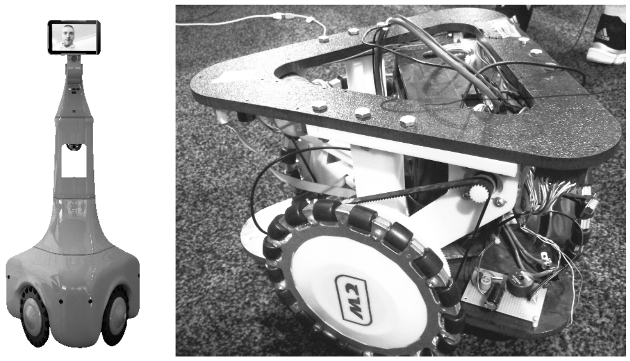

Fig. 3. VirtualME robot by CrossWing [14]. The left panel is a profile of the robot with coverings. The right panel shows the exposed mechanical and electronic components of the three omniwheeled base.

If an obstacle moves a significant distance in the interval $N \cdot T$ (the time interval of the NMPC prediction horizon), then the motion of the obstacle should be considered within the optimization. Doing so allows the robot to react earlier to the presence of a moving obstacle (possibly avoiding a collision) and leads to a lower global minimum of cost. It also reduces the risk of the potential field destabilizing the convergence of the optimization algorithm.

If the future trajectory of moving obstacles is not known *ex ante*, it must be approximated for the controller.

The known or approximated trajectory of obstacles is used to update the potential field during execution of Algorithm 1, so that each $\boldsymbol{p}_k$ is computed with a potential field which reflects the arrangement of obstacles for that time-step.

V. Results and Discussion

A. Experiment With Robot in Stationary Obstacle Field

Before considering moving obstacles or walls, earlier experiments had the robot navigating between waypoints through simulated fields of point obstacles. The robot used for the experiments was the virtualME platform from CrossWing [14], shown in Fig. 3.

VirtualME is a three-omni wheeled robot with an onboard Android tablet and a single-board computer, (a Raspberry Pi in the model used), running GNU/Linux. A command-line interface is served from the single-board computer that accepts various motion commands. The tablet provided a WiFi connection and forwarded a TCP/IP socket to the command-line server on the Raspberry Pi.

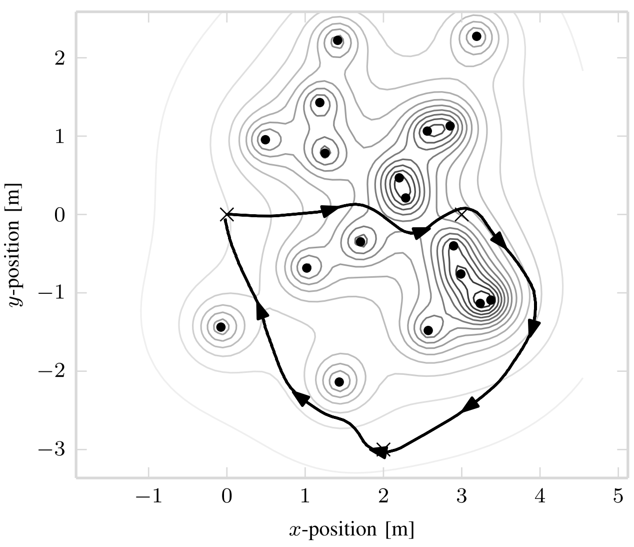

Fig. 4. Path (black line with arrows) taken by the robot, as reported by internal odometry, through a simulated potential field (gray isolines) cast by stationary obstacles (black dots). The waypoints targeted by the robot are marked with crosses.

Commands could be passed to the robot over the WiFi connection, or the NMPC controller could be run directly on the Raspberry Pi.

The NMPC controller generated commands for the robot which specified speed, direction and turn-rate. The direction was fixed to the forward heading and the speed fixed at 0.4 m/s. The turn rate was computed by the NMPC controller. The robot's internal odometrical position estimate was fed back to the NMPC controller.

To design the simulated environment, an amorphous arrangement of obstacles was created by shifting a 2-D periodic lattice of points by normally distributed pseudorandom vectors. Starting from $(0, 0)$ m, the robot was given three waypoints that took it through the obstacle field. A waypoint was successfully reached if the robot's reported position was closer than 10 cm from it. The obstacle potential field and waypoints are shown in Fig. 4, along with a path measured by running the controller on the robot's Raspberry Pi and polling the robot's internal odometry estimates.

The length of the robot's trajectory in Fig. 4 is 11.91 m and the error drift of the odometry readings accumulated to approximately 60 cm over that trajectory. Although one should consider that error significant, it does not diminish the success of the NMPC controller in navigating the field.

B. Simulation With Moving Obstacles

In Section IV-A, two methods for handling moving obstacles are described. The first is the most basic, merely updating obstacle positions between NMPC steps. The second is more comprehensive and couples the trajectories (known or approximated) of the obstacles to the path optimization. Obstacles in the former method will be referred to as noncoupling and the later as coupling. The term coupled is meant in the sense that the path computed in the optimization step is explicitly dependent on the motion of the obstacles.

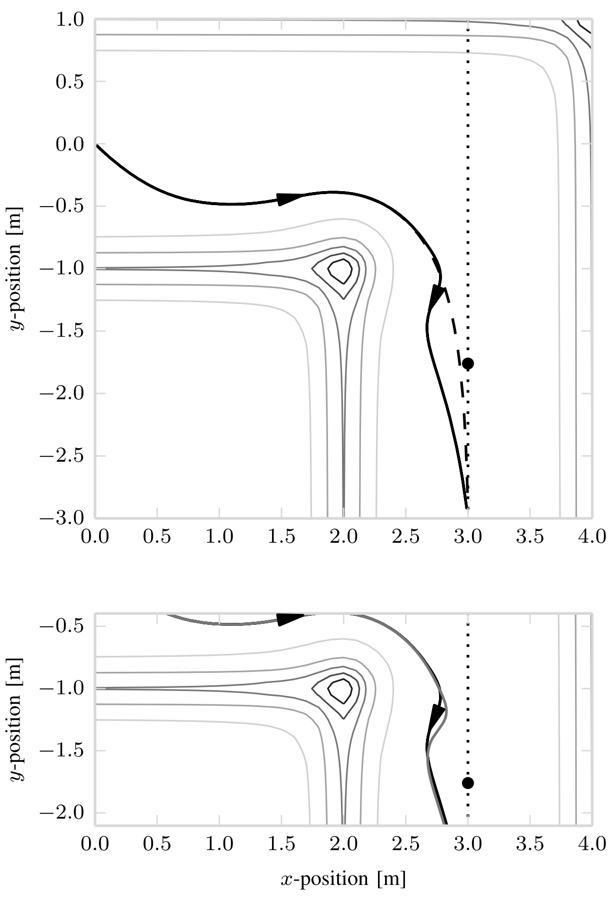

Fig. 5. Top panel: comparison of simulated paths, one taken by an unobstructed robot (dashed line) and one taken by a robot avoiding a moving coupled obstacle (solid line) navigating through a 2-m wide hallway. The robot starts at $(0, 0)$, and is moving toward $(3, -3)$. The moving obstacle moves along the dotted line at 0.4 m/s. The black circle on the dotted line marks the position of the moving obstacle at the moment of apparent divergence of the dashed and dotted lines [roughly $(2.75, -1.0)$]. The isolines illustrate the contour of the potential field cast by the walls of the hallway. Bottom panel: similar to the top panel, this panel contrasts the difference in paths when the obstacle motion is coupled (black line) and noncoupled (gray line).

A simple scenario of an obstacle moving up a 2-m wide hallway is constructed. The hallway has a 90° bend and the robot is to navigate from one end to the other. The moving obstacle starts from the opposite end of the hallway traveling at 0.4 m/s in the y-direction. The path of the obstacle is chosen so that the robot will have to maneuver evasively soon after it comes around the corner. Fig. 5 (top panel) illustrates that scenario, and includes the path taken by the robot to avoid the coupled moving obstacle, as well as the path it would have taken if it were unobstructed.

Fig. 5 (bottom panel) shows the difference in paths taken by the robot upon meeting coupled and non-coupled obstacles. The difference is subtle because the obstacle moves slowly. However, it is apparent from the diagram that when the obstacle motion is not coupled, the robot comes closer to the obstacle, and steers more aggressively to avoid it. The cost of that path is therefore higher.

VI. Conclusion

We noted that the robot was easily trapped by constellations of obstacles or walls. This clearly identifies the need for a higher level path planner. We have created several schemes which successfully integrate globally planned path information with the NMPC controller to remove or shrink the state space neighborhoods which lead to these limit cycles.

We are continuing our research with integrated global planners, and are currently working to minimize the global information needed for stability. It is desirable to create a controller which locally adjusts for obstacles and environmental dynamics, while making the best possible progress toward a global target. This relationship allows NMPC to extend the viable lifetime of a global plan, and minimizes the amount of detail it must include. The NMPC control and rapid global path planning can both be computationally expensive. It is natural to seek a compromise between the two, making the most of available data while safeguarding against a rapidly changing environment.

The next major endeavor will involve the development of a heuristic supervisor to track the stability of the NMPC controller and to dynamically adjust the weighting parameters, time step interval, and other parameters effecting the stability of the controller. This effort must of course proceed from a thorough analysis of controller stability.

The current scheme does not control velocity and it does not take advantage of the ability of the robot to make instantaneous direction changes. In future research endeavors, the cost function will be modified to allow the robot to safely travel at speeds appropriate to the local obstacle field and to safely take advantage of the fact that the direction of travel and the orientation of the robot are independent.

Acknowledgment

The authors would like to thank CrossWing Inc. for the use of their virtualME platform and their ongoing hardware and software support.

References

[1] S. Qin and T. A. Badgwell. (2003 Jul.). A survey of industrial model predictive control technology. *Control Eng. Pract.* [Online]. *11(7)*, pp. 733–764. Available: http://linkinghub.elsevier.com/retrieve/pii/S0967066102001867

[2] J. Richalet, A. Rault, J. L. Testud, and J. Papon. (1978, Sep.). Model predictive heuristic control. *Automatica* [Online]. *14(5)*, pp. 413–428. Available: http://linkinghub.elsevier.com/retrieve/pii/0005109878900018

[3] R. Findeisen, L. Imsland, F. Allgower, and B. A. Foss. (2003, Jan.). State and output feedback nonlinear model predictive control: An overview. *Eur. J. Control* [Online]. *9(2–3)*, pp. 190–206. Available: http://www.sciencedirect.com/science/article/pii/S0947358003702751

[4] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. (2000). Constrained model predictive control: Stability and optimality. *Automatica* [Online]. *36(6)*, pp. 789–814. Available: http://www.sciencedirect.com/science/article/pii/S0005109899002149

[5] F. Fahimi. (2007, Aug.). Non-linear model predictive formation control for groups of autonomous surface vessels. *Int. J. Control* [Online]. *80(8)*, pp. 1248–1259. Available: http://www.tandfonline.com/doi/abs/10.1080/00207170701280911

[6] J. M. Eklund, J. Sprinkle, and S. S. Sastry. (2012, May). Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft. *IEEE Trans. Control Syst. Technol.* [Online]. *20(3)*, pp. 604–620. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5756676

[7] G. Klančar and I. Škrjanc. (2007, Jun.). Tracking-error model-based predictive control for mobile robots in real time. *Robot. Auto. Syst.* [Online]. *55(6)*, pp. 460–469. Available: http://linkinghub.elsevier.com/retrieve/pii/S0921889007000140

[8] S. G. Vougioukas, "Reactive trajectory tracking for mobile robots based on non linear model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3074–3079.

[9] L. Lapierre, R. Zapata, and P. Lepinay. (2007, Apr.). Combined path-following and obstacle avoidance control of a wheeled robot. *Int. J. Robot. Res.* [Online]. *26(4)*, pp. 361–375. Available: http://ijr.sagepub.com/content/26/4/361.short

[10] S. Yu, X. Li, H. Chen, and F. Allgöwer, "Nonlinear model predictive control for path following problems," *Int. J. Robust Nonlinear Control*, 2014, doi: 10.1002/rnc.3133.

[11] C.-H. Hsieh and J.-S. Liu, "Nonlinear model predictive control for wheeled mobile robot in dynamic environment," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM*, Jul. 2012, pp. 363–368.

[12] G. J. Sutton and R. R. Bitmead, "Performance and computational implementation of nonlinear model predictive control on a submarine," in *Nonlinear Model Predictive Control* (Progress in Systems and Control Theory), vol. 26, F. Allgöwer, A. Zheng, and C. I. Byrnes, Eds. Basel, Switzerland: Birkhäuser Basel, 2000, pp. 461–472.

[13] D. E. Kirk, *Optimal Control Theory: An Introduction*. New York, NY, USA: Dover, 2004.

[14] (2011). *Crosswing's Corporate Website*. CrossWing's, Markham, ON, Canada [Online]. Available: http://www.crosswing.com/virtualme/

Timothy A. V. Teatro (S'13) received the B.Sc. (Hons.) degree in physics and the M.Sc. degree in materials science, in 2007 and 2009 respectively, from the University of Ontario Institute of Technology, Oshawa, ON, Canada, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research background has focused in computational physics, with particular interest in *ab initio* quantum mechanical calculations of solid state systems. He has also conducted research in forensic blood-flight and spatter pattern analysis. In his doctoral studies, his research interests have expanded to include autonomous systems, optimal control, and model predictive control.

J. Mikael Eklund (S'98–M'03–SM'11) received the B.Sc. and M.Sc. degrees in engineering and the Ph.D. degree from Queens University, Kingston, ON, Canada, in 1989, 1997, and 2003, respectively.

He was a Simulator Flight Control Systems Engineer with CAE Electronics, Montreal, QC, Canada, between his bachelor's and master's studies. He was a Visiting Postdoctoral Scholar with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA, until 2006. He is an Associate Professor with the Department of Electrical and Software Engineering, University of Ontario Institute of Technology, Oshawa, ON, Canada. His current research includes autonomous systems, including robotic vehicles and smart medical systems.

Ruth Milman (S'97–M'04) received the B.A.Sc. (Hons.) degree in computer engineering from the University of Toronto, Toronto, ON, Canada, in 1995, and the M.A.Sc. and Ph.D. degrees from the Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, in 1997 and 2004, respectively.

She was a Post-Doctoral Researcher with the University of Toronto from 2005 to 2007, and joined the University of Ontario Institute of Technology, Oshawa, ON, in 2007. She is an Assistant Professor with the Department of Electrical and Software Engineering, Faculty of Engineering and Applied Science, University of Ontario Institute of Technology. Her current research interests include optimization and computationally efficient algorithms for model predictive control, and the application of both linear and nonlinear MPC to autonomous systems.