Robot Navigation with MPEPC in Dynamic and Uncertain Environments: From Theory to Practice

Jong Jin Park, Collin Johnson and Benjamin Kuipers

Abstract— **True shared control of robotic vehicle is only possible when the robot attains sufficient level of autonomy. To be genuinely useful for human users, the robot must be able to carry out high-level navigational instructions on its own (e.g., go to my office desk), while allowing the user to take control whenever is necessary. However, in real life situations autonomous navigation is a difficult challenge since the environment is almost always dynamic and uncertain.**

In a companion paper [1], we have introduced the model predictive equilibrium point control (MPEPC) framework, which allows a robot wheelchair to navigate gracefully in dynamic, uncertain and structured environments in the presence of multiple pedestrians. The algorithm works well in simulations and in a physical robot: it works both in theory and in practice.

In this paper, we present the full technical description of the MPEPC-based navigation algorithm, and provide further discussions on theoretical aspects of the algorithm that allows the robot to navigate in dynamic and uncertain environments.

I. INTRODUCTION

To allow a human user to truly share control with a robot vehicle, the system must be intuitive and trustworthy: the user must be able to instruct the robot in a natural and intuitive way (e.g. go to my office desk), and the robot must be able to carry out the instruction safely and reliably while complying to user preferences. The robot must attain sufficient level of autonomy in order for the control to be granted. In this paper, we address the important problem of reliable autonomous navigation, when the (wheelchair) robot is given a destination pose as a high-level instruction.

The wheelchair system is a particularly good platform for autonomous navigation research. The platform is large enough to mount various sensors and carry a passenger, but small enough to run experiments indoors without requiring special infrastructures. An intelligent wheelchair capable of safe and autonomous navigation can provide necessary level of motor assistance to individuals with physical and mental disabilities.

The wheelchair system also poses special challenges. With pedestrians, a typical indoor environment is dynamic and uncertain. Pedestrians are more agile than a typical powered wheelchair, as they are omni-directional and can run faster. Like a car, the wheelchair is underactuated with nonholonomic constraints (cannot move sideways). The motion of the wheelchair needs to be *graceful* [2], avoiding all collisions and limiting the velocity, acceleration and jerk, to ensure the safety and comfort of potentially physically fragile users.

There has been impressive progress in the motion planning and autonomous navigation in the past few decades, but existing algorithms typically focus on solving a subset of the full problem. Path planning algorithms such as RRT [3], RRT* [4], A* and its variants [5] are very popular in static or quasi-static environments, but may not work well in highly dynamic environments. Reactive and control-oriented methods such as potential field [6], navigation function [7], gradient methods [8], VFH [9], DWA [10] and VO [11] are often more suitable in dynamic environments, but with reactive methods it may be difficult to generate more sophisticated robot behaviors and to limit accelerations and jerks, as the motion is determined based on observation on a single time step. The notion of graceful motion has been introduced and explored by Gulati [2], [12], but in general there has been limited work on planning comfortable motion. Planning in uncertain environment is also an open area, but important results are published recently [13], [14], [15].

In a companion paper [1], we present the model predictive equilibrium point control (MPEPC) framework for robot navigation in uncertain and dynamic environments. We show that with the proposed algorithm, we can formulate local navigation as a continuous, low-dimensional, and unconstrained optimization problem, which is easy to solve. The proposed algorithm can generate trajectories on-line that are safe, smooth and comfortable in structured, dynamic and uncertain environments.

In this paper, we identify critical features of the MPEPC-based navigation algorithm and provide further theoretical discussions. The proposed algorithm depends on: (1) decomposition of the navigation task; (2) integrated planning and control; (3) a pose-stabilizing feedback; and (4) decision-theoretic trajectory evaluation. The technical detail of the proposed algorithm is also described (as in [1]).

II. MPEPC NAVIGATION

A. Decomposition of the Navigation Task

In dynamic environments (e.g. a hall with a large number of pedestrians), we need trajectory planning in configuration-time space. Path planning alone is not sufficient, since a path is defined in configuration space and does not include

This work has taken place in the Intelligent Robotics Lab in the Computer Science and Engineering Division of the University of Michigan. Research of the Intelligent Robotics lab is supported in part by grants from the National Science Foundation (CPS-0931474 and IIS-1111494), and from the TEMA-Toyota Technical Center.

J. Park is with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA jongjinp@umich.edu

C. Johnson is with the Computer Science and Engineering Division, Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA collinej@umich.edu

B. Kuipers is with Faculty of Computer Science and Engineering Division, Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA kuipers@umich.edu

important timing information that is required to efficiently avoid collision with dynamic objects. But real environments are always uncertain, making trajectory planning in large time horizon impractical. It is difficult to accurately predict motion of a pedestrian for more than a few seconds.

For efficient on-line trajectory planning, we decompose the navigation task into two parts. We explicitly differentiate between the static and the dynamic part of the environment, and first plan a metric path to a destination in the static environment. We use navigation function (NF) [7] as our representation of the static plan, as it encodes a set of shortest-distance paths and is easy to compute (Fig. 1).[1] Then we use the NF as an instruction for the local on-line trajectory planner in the dynamic environment, treating it as a cost-to-go to the final destination in the model predictive control (MPC) setting. We will formally introduce the framework in the next subsection.

Detection and tracking of the pedestrians becomes considerably easier with the decomposition. We generate the static map via SLAM, and then identify the portions of laser scans that are not explained by the static map [16]. We cluster those dynamic scans and track them via Kalman filters for position and linear velocity (Fig. 2).

B. MPEPC and Integrated Planning and Control

For the on-line, local trajectory planning, we build upon a receding horizon model predictive control (MPC) architecture (e.g., see [17], [18], [19] for UAV examples, and [20], [21], [22], [23] for ground vehicle examples). MPC is useful in dynamic and uncertain environments as it provides a form of information feedback with constant re-planning, by continuously incorporating new information with the receding time horizon and optimizing the prediction of robot behavior within the time horizon. In the optimization framework, tradeoffs between multiple objectives can be expressed explicitly in the objective function.

Unlike most other MPCs, we start with a stabilizing feedback control policy

$$u = \pi_\zeta(x, x_*) \tag{1}$$

which always steers the system toward $x \to x_*$, where x_* is a control goal, or a reference. ζ represents manipulatable parameters in π_ζ (e.g., control gains).

We formulate the model predictive equilibrium point control (MPEPC) by defining the optimization space for the MPC with the pose-stabilizing controller (1). The optimization problem is cast as:

$$\underset{z_*=(x_*,\zeta)}{\text{minimize}} \quad J(x, x_*, \zeta, T) \tag{2}$$

$$\text{subject to} \quad \dot{x} = f_{\pi_\zeta}(x, \pi_\zeta(x, x_*)) \tag{3}$$

$$x(0) = x_I \tag{4}$$

$$x(t) \in \mathrm{X} \tag{5}$$

[1]Although cases exists when the navigation function can fail for non-circular robots (e.g. a long narrow corridor where the robot cannot rotate, with a destination behind the robot), for the wheelchair robot those cases are very uncommon in typical indoor environments.

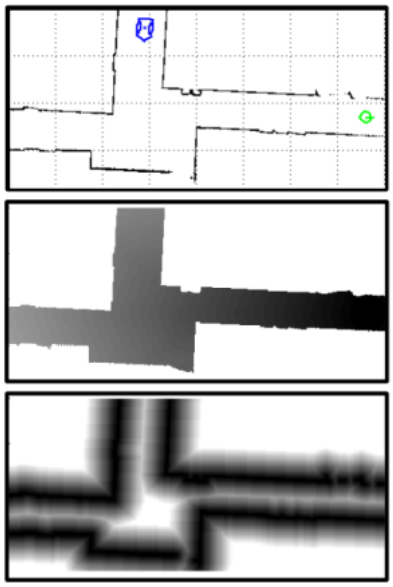

Fig. 1. The information about the static world available to the robot. **Top**: Occupancy grid map (20×10 m) with initial robot pose (blue) and the final goal (green circle). **Middle**: Navigation function. **Bottom**: Proximity to walls.

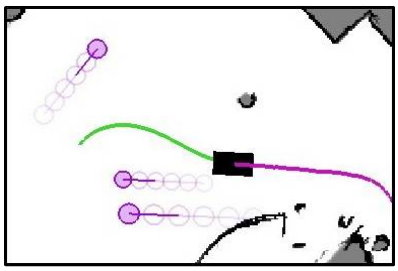

Fig. 2. (Best viewed in color) A snapshot of a physical wheelchair (black rectangle) navigating in a dynamic environment. Circles denote tracked pedestrians, with protruding lines indicating estimated linear velocity. Green trajectory represent a planned time-optimal trajectory.

where (2) is the navigation cost (which include the cost-to-go) that needs to be minimized, (3) is the system dynamics under the stabilizing policy π_ζ, (4) is the boundary condition, and (5) is the constraint on states with space of admissible states X.[2]

Observe that in (3), x_* essentially acts as a control input to the system, as the trajectory of the stable system is determined by the initial condition and the location of the equilibrium point. That is, the system can be controlled by changing the reference, or the target equilibrium point, x_*, over time. Such method is often referred to as equilibrium point control (EPC) (e.g., [24], [25]).[3]

For robot navigation with MPEPC, x_* is a target pose, or a *motion target*, in the neighborhood of the robot. The robot

[2]For navigation, (5) basically represents collision constrains. In following subsections, we introduce probabilistic weights and absorb (5) to the cost definition, formulating local navigation as a continuous, low dimensional and unconstrained optimization problem.

[3]The EPC framework is often inspired by equilibrium point hypothesis (EPH) [26], which a well-known hypothesis in biomechanics that locomotion is controlled by adapting the equilibrium point of a biomechanical system over time.

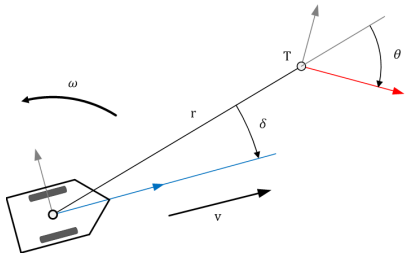

Fig. 3. Egocentric polar coordinate system with respect to the observer. Here, both θ and δ have negative values.

moves toward the motion target under the stabilizing control policy π_ζ. The robot navigates by continuously updating and moving toward a time-optimal motion target, so that it can reduce the cost-to-go to the final destination while satisfying all the constraints. The robot will converge to a motion target if the target stays stationary.[4]

The main benefit of the approach is that with a careful construction of the stabilizing control policy, the equilibrium point x_* can provide a compact, rich, and semantically meaningful parametrization of local trajectories in continuous domain. Note that constraints on control inputs u do not appear because it can be incorporated into design of $\pi_\zeta(x, x_*)$.

In the next subsection, we build upon a feedback control law (6) developed in [27] to construct the pose-stabilizing feedback control policy (1) suitable for MPEPC navigation.

C. Pose Stabilizing Control Policy

Suppose an observer is situated on a vehicle and fixating at a target T at a distance r away from the vehicle. Let θ be the orientation of T with respect to the line of sight from the observer to the target, and let δ be the orientation of the vehicle heading with respect to the line of sight. The egocentric polar coordinates $(r, \theta, \delta)^T$ completely describe the relation between the robot pose and the target pose, as shown in Fig. 3.

In [27], it is shown that linear velocity v and angular velocity ω satisfying the following control law (6) will globally drive the robot toward a target at $(r, \theta, \delta)^T$ by singular perturbation [28].

$$\omega = -\frac{v}{r}\left[k_2(\delta - \arctan(-k_1\theta)) + (1 + \frac{k_1}{1+(k_1\theta)^2})\sin\delta\right] \tag{6}$$

(6) describes a generalized pose-following control law with shape and gain parameters k_1 and k_2.[5] The path curvature κ resulting from the control law is

$$\kappa = -\frac{1}{r}\left[k_2(\delta - \arctan(-k_1\theta)) + (1 + \frac{k_1}{1+(k_1\theta)^2})\sin\delta\right] \tag{7}$$

[4]The equilibrium point (the motion target) is not to be confused with the final destination. Figuratively speaking, MPEPC-based navigation is similar to controlling a horse with a carrot and a stick.

[5]k_1 determines how fast the orientation error is reduced relative to distance error. $k_1 = 0$ reduces the controller to pure way-point following. k_2 is a gain value.

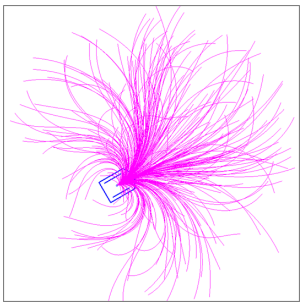

Fig. 4. Randomly selected 300 samples from the continuous space of smooth and realizable *closed-loop* trajectories, parametrized by the four dimensional vector $z_* = (r, \theta, \delta, v_{\max})^T$ over a time horizon $[0, T]$. The trajectories are generated from dynamically simulated vehicle (with non-holonomic constraints and actuator saturations) under the stabilizing kinematic controller.

with a relation $\omega = \kappa v$ which holds for planar motion.

One of the key features of the control law is that the convergence property does not depend on the choice of positive linear velocity v. We choose a curvature-dependent choice of linear velocity to make motion comfortable:

$$v(\kappa) = v(r, \theta, \delta) = \frac{v_{\max}}{1 + \beta|\kappa(r, \theta, \delta)|^\lambda} \tag{8}$$

where the parameters β and λ determine how the vehicle slows down at a high curvature point, and $v_{\max}$ is a user-imposed maximum linear velocity, with $v \to v_{\max}$ as $\kappa \to 0$. It can be shown that all velocities, acceleration and jerks are bounded under (6) and (8) so that the motion of the robot is smooth and comfortable. For our experiments, we use $k_1 = 1.5$, $k_2 = 3$, $\beta = 0.4$ and $\lambda = 2$.

By adding a slowdown rule near the target pose with

$$v = \min(\frac{v_{\max}}{r_{\text{thresh}}} r,\, v(\kappa)) \tag{9}$$

with some user-selected distance threshold r_{thresh}, the target T becomes a non-linear attractor that the vehicle will exponentially converge to,[6] where the maximum linear velocity parameter $v_{\max}$ can be understood as a gain value which determines the strength of the non-linear attractor. For our experiments, we use $r_{\text{thresh}} = 1.2\,\text{m}$.

Now, we have a 4-dimensional vector

$$\begin{aligned} z_* &= (\text{T}, v_{\max}) \\ &= (r, \theta, \delta, v_{\max})^T \end{aligned} \tag{10}$$

which completely describes the target of the vehicle system in the configuration-time space, and the trajectory of the vehicle converging to that target. Thus z_* also parametrizes *a space of smooth trajectories generated by the feedback control*, (6) - (9). As the configuration-time space for the robot is 4-dimensional, we have the most compact degree of freedom with our parameterization. We emphasize that the closed-loop trajectories parametrized by z_* (Fig. 4) are smooth and realizable (by the controller) by construction; i.e.

[6]With (9), $v = v(\kappa)$ when $r > r_{\text{thresh}}$ since $v(\kappa) \le v_{\max}$. Also, $v \sim r$ near $r = 0$ canceling $1/r$ in (6) and rendering the system exponentially stable. Proof shown in [27].

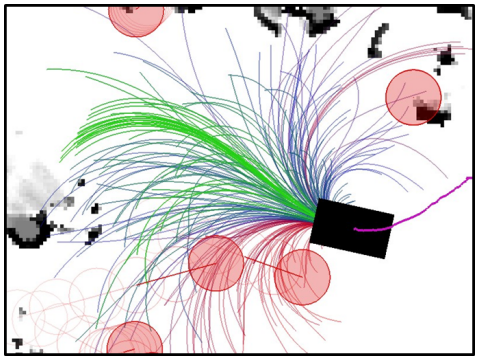

Fig. 5. (Best viewed in color) The wheelchair (a black rectangle) evaluating trajectory candidates in a crowded hall. Green to blue to red color gradient indicates low to high expected navigational cost (12). See text for details.

the constraints on control input are satisfied for any motion target x_* at the control level.

Compared to traditional MPC-based approaches [29], [30] which typically search over an open-loop solution over a discretized sequence of control inputs u, our representation is much more compact and we retain the benefit of feedback. It ensures more accurate prediction of future robot trajectory and robustness to fast disturbances. With the MPEPC, it is also possible to benefit from desirable properties in the control law, which in our case is smoothness and comfort in motion [27].

D. Decision-Theoretic Trajectory Evaluation

In our setting, the problem of navigation for the robot is to find the motion target and velocity gain which generates the most desirable trajectory over the lookahead horizon at each planning cycle, so that the robot can make the most progress to the final destination while minimizing discomfort and the chance of collision. Formally, let

$$q_{z_*} : [0,T] \to C \tag{11}$$

be a trajectory of the robot parametrized by z_* within a finite time horizon T, where $C \simeq \mathrm{R}^2 \times \mathrm{S}^1$ is the configuration space of the robot. The optimization problem for the predictive planner is to select z_* which minimizes the following sum of *expected* costs over the trajectory $q_{z_*}([0,T])$:

$$\begin{aligned} J(x, z_*, T) &= E[\phi(q_{z_*})] \\ &= E[\phi_{\text{progress}} + \phi_{\text{collision}} + \phi_{\text{action}}] \end{aligned} \tag{12}$$

where $\phi_{\text{collision}}$ is the cost of collision, ϕ_{action} is the cost of action, and ϕ_{progress} is the negative progress (Fig. 5). We use the optimal control framework to deal with the tradeoffs between the objectives, rather than to achieve the absolute optimality.

We put the problem into a standard decision-theoretic framework by using the *expected* values in our cost definition.[7] The use of expected values lets us properly deal with model and observation uncertainties.

[7]While it is difficult to find an example of MPC-based planning which uses expected values in the cost, it seems very natural to use expected values as MPC, by definition, is optimizing the *prediction* of robot behaviors.

1) Probability of Collision: Due to uncertainty in the motion of the robot and dynamic objects, it is highly desirable to incorporate the notion of probability into the cost function. Accurate representation of robot and object motion uncertainties is an interesting and important problem [14] [13], but we do not attempt to solve it here. We use a simplified model for the probability of collision.

Suppose a robot is expected to travel along a trajectory q_{z_*} at time t. We model the probability p_c^i of collision of the robot with the i-th object in the environment over a small time segment $[t, t+\Delta t]$ as a bell-shaped function:

$$p_c^i(d_i(t), \sigma_i) = exp(-d_i(t)^2/{\sigma_i}^2) \tag{13}$$

where $d_i(t)$ is the minimum distance from any part of the robot body to any part of the i-th object at time t, which is computed numerically. σ_i is an uncertainty parameter. We treat the static structure as a single object in the environment.

With (13), we can define the *survivability* $p_s(t)$ of the robot over $q_{z_*}([t, t+\Delta t])$:

$$p_s(t) \equiv \prod_i^m (1 - p_c^i(t)) \tag{14}$$

where m is the number of obstacles including the static structure. Thus $p_s(t)$ represents the probability that $q_{z_*}([t, t+\Delta t])$ will be collision-free with all obstacles. In our experiments, we use empirically chosen values for the uncertainty parameters: $\sigma = 0.1$ for the static structure, which represents robot position/simulation uncertainties; and $\sigma = 0.2$ for dynamic objects, which represents combined uncertainties of the robot and dynamic objects.

2) Expected Cost of a Trajectory: Given a 2D navigation function $NF(\cdot)$ and its gradient, which encodes shortest distance plans from all starting points to the destination in the static environment, the negative progress $\phi_{\text{progress}}(t)$ over $q_{z_*}([t, t+\Delta t])$ can be written as[8]

$$\phi_{\text{progress}}(t) = NF(q_{z_*}(t+\Delta t)) - NF(q_{z_*}(t)) \tag{15}$$

So that with (14) and (15), we can write *expected negative progress* $E[\phi_{\text{progress}}]$ over the static plan $NF(\cdot)$ which needs to be minimized as

$$E[\phi_{\text{progress}}] \equiv \sum_{j=1}^{N} p_s(j) \cdot \phi_{\text{progress}}(j) + c_1 \Delta\theta(q_{x_*}(T)) \tag{16}$$

where j is a shorthand notation for the j-th sample pose at time t_j along the trajectory, with $t_N = T$. The last term $\Delta\theta(q_{x_*}(T))$ is a weighted angular difference between the final robot pose and the gradient of the navigation function, which is an added heuristic to motivate the robot to align with the gradient of the navigation function. To guarantee the robot will never voluntarily select a trajectory leading to collision, we set $p_c^i(j \geq k) = 1$ (thus $p_s(j \geq k) = 0$) after any sample with $p_c^i(k) = 1$.

Note that without p_s and $\Delta\theta(\cdot)$, the negative progress reduces to $\phi_{\text{progress}} = NF(q_{x_*}(T)) - NF(q_{x_*}(0))$, where $NF(q_{x_*}(0))$ is a fixed initial condition which acts as a

[8]The 2D $NF(\cdot)$ only uses the position information from the pose $q_{x_*}(t)$.

normalizer and does not affect the optimization process. $NF((q_{x_*}(T))$ is the terminal cost we want to minimize, which is the underestimated cost-to-go to the final destination encoded by $NF(\cdot)$.

Similarly, with (13), we can write the *expected cost of collision* as

$$E[\phi_{\text{collision}}] \equiv \sum_{j=1}^{N} \sum_{i}^{m} p_c^i(j) \cdot c_2 \phi_{\text{collision}}^i(j) \tag{17}$$

where c_2 is a weight and $\phi_{\text{collision}}^i(j)$ is the agent's idea for the cost of collision with the i-th object at the j-th sample. In this paper, we treat all collisions to be equally bad and use $\phi_{\text{collision}}^i(j) = 0.1$ for all i, j.

For the action cost, we use a usual quadratic cost on velocities:

$$\phi_{\text{action}} = \sum_{j=1}^{N} (c_3 v^2(j) + c_4 \omega^2(j)) \Delta t \tag{18}$$

where c_3 and c_4 are weights for the linear and angular velocity costs, respectively. For the *expected cost of action*, we assume $E[\phi_{\text{action}}] = \phi_{\text{action}}$.

The overall *expected* cost of a trajectory is the sum of (16), (17), and (18), as given in (12). The probability weights p_c^i and p_s (13) - (14) can be understood as mixing parameters which properly incorporates the collision constraint (5) into the cost definition. The probability weights renders the collision term completely dominant when the robot is near an object, while letting the robot focus on progress when the chance of collision is low. With (12), the optimization problem (2) is now in continuous, low-dimensional and unconstrained form.

III. RESULTS

In this section, we briefly show some example results. In our lab, we have a differentially-driven wheelchair robot equipped with a laser range finder that builds local metrical and global topological maps [31] (Fig. 6).

A. Implementation with Pre-sampling for Initial Condition

We have used off-the-shelf optimization packages[9] for implementation. For our experiments, the receding horizon was set at $T = 5\,\text{s}$, the plan was updated at 1 Hz, and the underlying feedback controller ran at 20 Hz.

In practice, we found the performance of the optimizer (for both speed and convergence to global minimum) depends greatly on the choice of the initial conditions. Thus the optimization process is implemented in two phases. First, we coarsely sample the search space[10] to find a good initial condition. The best candidate form the pre-sampling phase is passed to a graient-based local optimizer as an initial condition for the final output.

With our problem formulation the computational cost for the numerical optimization is very small, achieving real-time performance. A typical numerical optimization sequence in each planning cycle converged in $< 200\,\text{ms}$ on a 2.66-GHz laptop, with ~ 220 trajectory evaluations on average in C++ (data not shown).

[9] *fmincon* in MATLAB and *NLopt* [33] in C++.

[10] by a dozen pre-selected samples which include the optimal target from the previous time step, stopping, typical soft/hard turns in MATLAB, and by a coarse global optimization algorithm in C++.

Fig. 6. The wheelchair robot ($0.76 \times 1.2\,\text{m}$). The wheelchair is differentially driven, such that linear and angular velocity are controlled independently. The robot is underactuated that the linear velocity is always aligned with the orientation of the robot (i.e. the robot cannot move sideways). The motor becomes saturated at linear and angular accelerations of $0.4\,\text{m/s}^2$ and $1.0\,\text{rad/s}^2$. The robot model for the forward prediction fully reflects the constraints.

Fig. 7. Snapshots of the wheelchair moving through a crowd [32].

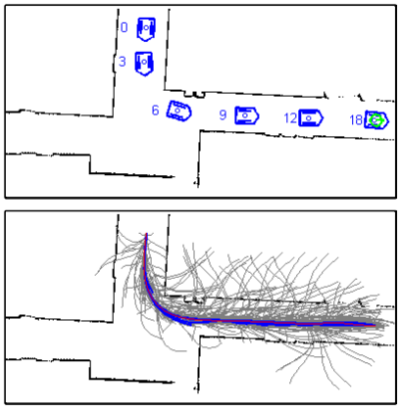

Fig. 8. (Best viewed in color) **Top**: Time-stamped snapshots of the motion of the robot ($0.76 \times 1.2\,\text{m}$) in a L-shaped corridor with a small opening (2 m wide, smallest constriction at the beginning 1.62 m). The robot moves quickly and smoothly in a free corridor. **Bottom**: Trajectories sampled by the planner (gray), time-optimal plans at each planning cycle (blue), and the actual path taken (red).

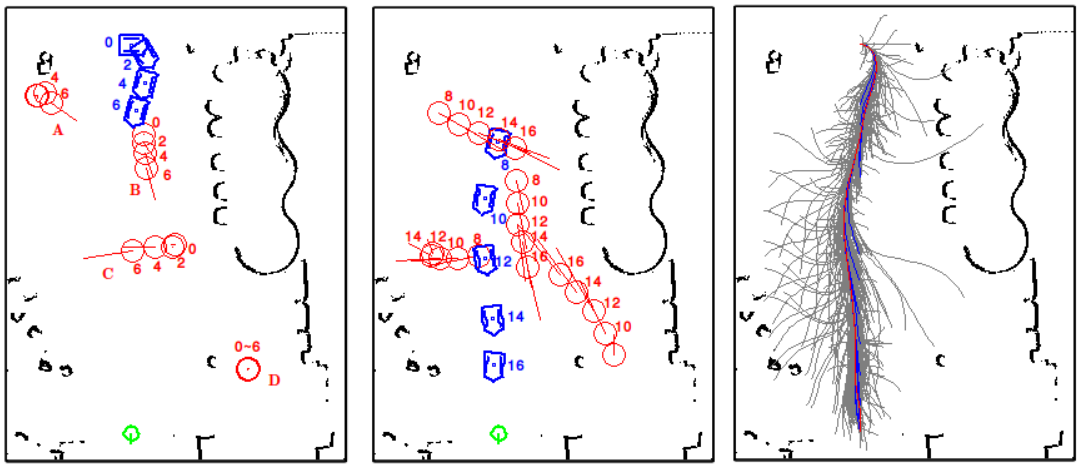

Fig. 9. (Best viewed in color) In an open hall (13.5×18 m) with multiple pedestrians (A-D, red circles), the robot (blue) estimates the trajectories of dynamic objects over the planning horizon (red lines) and navigates toward the goal (green circle) without collision. The robot begin traveling slowly in the first few seconds, then begin speeding up to avoid collision with pedestrians, and then slows down near the end. **Left and Center**: Time-stamped snapshots of the robot and pedestrian. **Right**: Trajectories sampled by the planner (gray), time-optimal plans selected at each planning cycle (blue) and the actual path taken (red).

B. Experiments

The algorithm works well both in simulations (Fig. 8 - 9) and in physical experiments (Fig. 7). In this paper, we only show data from two simulated example runs for illustration: one in a tight L-shaped corridor and another in a hall with multiple pedestrians.[11] All sensor data are from actual data traces obtained by the wheelchair robot. Robot motion is dynamically simulated within the environments generated from the sensor data.

A navigation results in a tight L-shaped corridor are shown in Fig. 8. The planner searches for the optimal trajectory parameterized by z_* in the neighborhood of the robot bounded by $0 \le r \le 8$, $-1 \le \theta \le 1$, $-1.8 \le \delta \le 1.8$ and $0 \le v_{\max} \le 1.2$. The weights in the cost function are set to $c_1 = 0.2$, $c_2 = 1$, $c_3 = 0.2$, and $c_4 = 0.1$.

Fig. 8 shows a sequence of time-stamped snapshots of the robot motion as it makes a right turn into a narrow corridor (Top), the trajectories sampled by the planner (Bottom, gray), the time-optimal plans selected at each planning cycle (blue), and the actual path taken (red). The robot moves swiftly through an empty corridor along the gradient of the navigation function toward the goal pose. Once the robot moves within 2 m of the goal pose the robot switches to a docking mode and fixes the motion target at the goal. In this example, the robot safely converges to the goal pose in about 18 s.

In Fig. 9, robot motion in a hall with multiple dynamic objects is shown. Here, we have $0 \le r \le 8$, $-1 \le \theta \le 1$, $-1.8 \le \delta \le 1.8$ and $0 \le v_{\max} \le 1.9$. The weights in the cost function are set to $c_1 = 0.2$, $c_2 = 1$, $c_3 = 0.4$, and $c_4 = 0.2$. The robot prefers to move slowly with the higher weights for the action cost ($c_3 = 0.4$ and $c_4 = 0.2$). The robot is able to move among multiple pedestrians, slowing down and speeding up as needed.

[11] See [1] for more results in simulations. A publication on the physical experiments is under review.

IV. DISCUSSION: FROM THEORY TO PRACTICE

The proposed MPEPC-navigation algorithm allows the robot to navigate in dynamic and uncertain environment, exhibiting very reasonable behaviors both in simulations and in the physical robot. As stated in the introduction, the algorithm depends on several important features: (1) decomposition of the navigation task; (2) integrated planning and control; (3) a pose-stabilizing feedback; and (4) the decision-theoretic trajectory evaluation.

We start by decomposing the navigation task and the relevant information. We explicitly differentiate the static and dynamic part of the environment, and decompose the task of navigation into static path planning and dynamic trajectory generation. This way, the trajectoy planner can focus on the small-scale neighborhood of the robot in the configuration-time space. The decompostion sets the stage for real-time operation required in dynamic environments.

Planning and control is truely integrated in the MPEPC framework. The search space of for the local trajectory planner (MPC) is defined by the pose-stabilizing controller (EPC). In our work, the pose-stabilizing controller (6) - (9) allows the robot to reach any pose in space while satisfying control constraints. It also provides a compact, rich and semantically meaningful parameterization of the smooth and realizable *closed-loop* trajectories in continuous domain (Fig. 4). The EPC-based compact and efficient parameterization is the key to the real-time trajectory generation.

We observe that many of the existing path-planning methods are control-decoupled, where the control problem of following the planned path is not considered during the planning phase. In such control-decopled approaches, a solution form the planner often requires nontrivial and expensive post-processing to make it admissible for the controller. In the MPEPC, the planner searches the space of realizable trajectories by construction, thus eliminating

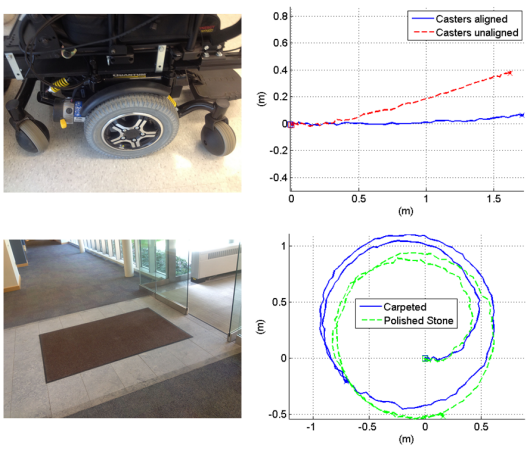

Fig. 10. Effects of castor wheels and surface conditions to the wheelchair motion, which are difficult to model. **Top**: Picture of the main and the castor wheels, and trajectories of the wheelchair under constant open-loop command of $v_{\text{cmd}} = 0.8\,\text{m/s}$ with the castor wheels aligned/misaligned. The wheelchair can exhibit large orientational error when the castor wheels are unaligned. **Bottom**: A typical surface condition in indoor environments, and trajectories of the wheelchair under constant open-loop command of $v_{\text{cmd}} = 1\,\text{m/s}$ and $\omega_{\text{cmd}} = 1.3\,\text{rad/s}$ on carpeted and polished stone surfaces.

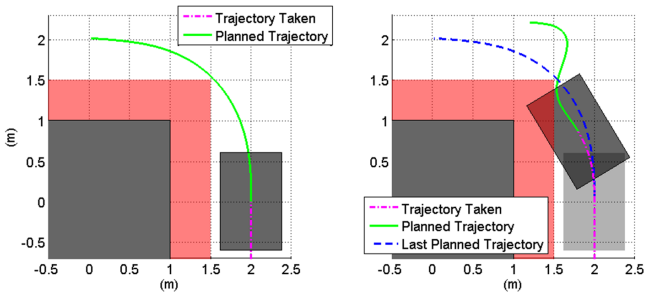

Fig. 11. (Best viewed in color) An example case of on-line trajectory planning solved as a constrained optimization problem. Static structures and the wheelchair robot are colored gray, and the hard bound on minimum clearance to obstacles (the collision constraint) is illustrated as pink-colored region. **Left**: To make a left turn, the trajectory planner found an optimal solution that satisfies the collision constraint. In general, optimal solutions are likely to be found on constraint boundaries, as shown in this example. **Right**: At the next planning cycle, the solution found in the previous time step (dashed line, blue) became invalid as the robot made a sharper turn than its original plan due to disturbances and/or an incorrect model. Since the vehicle is non-holonomic and the planner still needs to satisfy the hard collision constraint, new solution found (solid line, green) involves an oscillating s-curve, which is highly undesirable.

the need of the processing and minimizing the possiblity of actutor overload/failures.

The double feedback loop the in MPEPC adds to the robustness which can be critical in dynamic and uncertain environments. MPC introduces information feedback by constant replanning (at 1 Hz in simulations and at 3 Hz in physical experiments), making it very attractive for applications in dynamic environment. However, traditional MPCs usually search for open-loop solutions, which makes the system susceptible to fast disturbances and unmodeled dynamics. For example, we have found that castor wheels, floor finishing, small bumps on the floor, etc., can have significant effect on the wheelchair motion (Fig. 10), but are difficult to observe or model. The low-level feedback provided by EPC (at 20 Hz) is very important for fast disturbance rejection.

In structured, dynamic and uncertain environments, the probabilistic model for collision and the decision-theoretic trajectory evaluations was critical to the success of our algorithm. In typical path planning scenarios, it is common to enforce collision avoidance by introducing a hard-bound constraint on the minimum clearance to obstacles or on some function of the clearance. But we have observed that when combined with on-line replanning and model uncertainties, such hard-bound constraint can make the navigation algorithm brittle in highly structured or very crowded areas. An example case is shown in Fig. 11. Performance of our algorithm have greatly improved after we have introduced the (very simple) stochastic model of collision p_c^i (13) and p_s (14) which represents model and estimation uncertainties, evaluating trajectories for their *expected* values.

The probability weights can be viewed as a mixing parameter, which lets the planner to smoothly transition its focus between the progress and the collision based on the proximity to nearby hazards, allowing proper incorporation of the collision constraint into the cost definition. With EPC-based trajectory parameterization and the use of expected values (12), the optimization problem (2) is in continuous, low-dimensional and unconstrained form, which is easy to solve.

Customization of the robot behavior is also an important topic for human-robot interaction. For an autonomous robot to be trustworthy to humans, the robot must be able to modify its behaviors so that it can comply with user preferences. In our formulation, the weights in the cost definition provide an easy way to handle the trade-offs between competing sub-objectives, and a way to shape the behavior of the robot.

We believe the overall system could be improved by improving each module, e.g. better representation of motion uncertainties and better approximation of the cost-to-go.

V. CONCLUSION

In this paper, we view the problem of navigation as a continuous decision making process, where an agent with short-term predictive capability generates and selects a locally optimal trajectory at each planning cycle in a receding horizon control setting. By introducing a compact parameterization of a rich set of closed-loop trajectories given by a stabilizing control policy (6) - (9) and the use of expected values in the cost definition (12) - (18) for trajectory optimization in the MPEPC framework (2) - (5), the proposed algorithm achieves real time performance and generates very reasonable robot behaviors.

With MPEPC, we try to combine the optimality and the predictive capability of MPC (real-time information feedback by replanning with the receding horizon) with the simplicity and robustness of EPC (compact parameterization and low-level actuator feedback by the stabilizing controller). In effect, the MPEPC framework allows the planner to search simultaneously in the configuration space and in the con-

trol space, as it operates within the space of trajectories parametrized by a pose-stabilizing feedback control law.

We believe true shared control of robotic vehicle is only possible when the robot attains sufficient level of autonomy, so that a human user can transfer authority when desired. The proposed algorithm is important as it addresses the difficult problem of navigating in an uncertain and dynamic environment safely and comfortably while avoiding hazards, which is a necessary task for autonomous passenger vehicles.

REFERENCES

[1] J. Park, C. Johnson, and B. Kuipers, "Robot navigation with model predictive equilibrium point control," in *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2012.

[2] S. Gulati and B. Kuipers, "High performance control for graceful motion of an intelligent wheelchair," in *2008 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2008, pp. 3932–3938.

[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, Oct 1998.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[5] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Rob. Res.*, vol. 29, no. 5, pp. 485–501, Apr 2010.

[6] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *1991 IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, Apr 1991, pp. 1398–1404.

[7] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[8] K. Konolige, "A gradient method for realtime robot control," in *2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, 2000, pp. 639–646.

[9] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transaction on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, Jun 1991.

[10] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[11] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Rob. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[12] S. Gulati, "A framework for characterization and planning of safe, comfortable, and customizable motion of assistive mobile robots," Ph.D. dissertation, The University of Texas at Austine, Aug 2011.

[13] N. Du Toit and J. Burdick, "Probabilistic collision checking with chance constraints," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 809–815, Aug 2011.

[14] A. Lambert, D. Gruyer, and G. St. Pierre, "A fast monte carlo algorithm for collision probability estimation," in *10th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, Dec 2008, pp. 406–411.

[15] N. Du Toit and J. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transaction on Robotics*, vol. 28, no. 1, pp. 101–115, Feb 2012.

[16] J. Modayil and B. Kuipers, "The initial development of object knowledge by a learning robot," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 879–890, nov 2008.

[17] L. Singh and J. Fuller, "Trajectory generation for a uav in urban terrain, using nonlinear mpc," in *American Control Conf.*, vol. 3, 2001, pp. 2301–2308.

[18] E. Frew, "Receding horizon control using random search for uav navigation with passive, non-cooperative sensing," in *2005 AIAA Guidance, Navigation, and Control Conf. and Exhibit*, Aug 2005, pp. 1–13.

[19] S. Bertrand, T. Hamel, and H. Piet-Lahanier, "Performance improvement of an adaptive controller using model predictive control : Application to an uav model," in *4th IFAC Symposium on Mechatronic Systems*, vol. 4, 2006, pp. 770–775.

[20] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *American Control Conf.*, vol. 6, Jul 2004, pp. 5576–5581.

[21] P. Ogren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, Apr 2005.

[22] T. Howard, C. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Proceedings of the 7th Int. Conf. on Field and Service Robotics*, Jul 2009.

[23] G. Droge and M. Egerstedt, "Adaptive look-ahead for robotic navigation in unknown environments," in *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sep 2011, pp. 1134–1139.

[24] A. Jain and C. Kemp, "Pulling open novel doors and drawers with equilibrium point control," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS Int. Conf. on*, Dec 2009, pp. 498 –505.

[25] T. Geng and J. Gan, "Planar biped walking with an equilibrium point controller and state machines," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 2, pp. 253–260, Apr 2010.

[26] A. G. Feldman and M. F. Levin, "The equilibrium-point hypothesis past, present and future," in *Progress in Motor Control*, ser. Advances in Experimental Medicine and Biology, D. Sternad, Ed. Springer US, 2009, vol. 629, pp. 699–726.

[27] J. Park and B. Kuipers, "A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment," in *2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 4896–4902.

[28] H. K. Khalil, *Nonlinear Systems (3rd Edition)*, 3rd ed. New York, NY, U.S.: Prentice Hall, 2002.

[29] J. Rawlings, "Tutorial overview of model predictive control," *Control Systems, IEEE*, vol. 20, no. 3, pp. 38–52, Jun 2000.

[30] A. Grancharova and T. A. Johansen, "Nonlinear model predictive control," in *Explicit Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 2012, vol. 429, pp. 39–69.

[31] P. Beeson, J. Modayil, and B. Kuipers, "Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy," *Int. J. Rob. Res.*, vol. 29, pp. 428–459, Apr 2010.

[32] J. Park, C. Johnson, and B. Kuipers, "Graceful navigation via Model Predictive Equilibrium Point Control (MPEPC) in dynamic and uncertain environments," *Under Review*.

[33] S. G. Johnson. The NLopt nonlinear-optimization package. [Online]. Available: http://ab-initio.mit.edu/nlopt

[34] G. Indiveri, A. Nuchter, and K. Lingemann, "High speed differential drive mobile robot path following control with bounded wheel speed commands," in *2007 IEEE Int. Conf. on Robotics and Automation (ICRA)*, Apr 2007, pp. 2202–2207.

[35] E. Prassler, J. Scholz, and P. Fiorini, "Navigating a robotic wheelchair in a railway station during rush hour," *Int. J. Rob. Res.*, vol. 18, pp. 760–772, 1999.

[36] H. Tanner and J. Piovesan, "Randomized receding horizon navigation," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2640–2644, Nov 2010.