Robot Navigation with Model Predictive Equilibrium Point Control

Jong Jin Park, Collin Johnson and Benjamin Kuipers

Abstract— **An autonomous vehicle intended to carry passengers must be able to generate trajectories on-line that are safe, smooth and comfortable. Here, we propose a strategy for robot navigation in a structured, dynamic indoor environment, where the robot reasons about the near future and makes a locally optimal decision at each time step.**

We propose the model predictive equilibrium point control (MPEPC) framework, in which the ideas of time-optimality and equilibrium point control are combined in order to generate solutions quickly and responsively.

Our approach depends on a carefully designed compact parameterization of a rich set of closed-loop trajectories, and the use of expected values in cost definitions. This allows us to formulate local navigation as a continuous, low-dimensional, and unconstrained optimization problem, which is easy to solve. We present navigation examples generated from real data traces.

I. INTRODUCTION

The ability to navigate within an environment is crucial for any autonomous mobile agent to survive, to interact with the world, and to achieve its goals. Indeed, people are required to perform navigation tasks routinely across a wide range of settings: moving up to a desk in a structured office environment; walking through a crowded railway station; and even using an external vehicle such as driving a car to commute.

The fundamental goal of autonomous navigation is simple: getting from place A to place B, with little or no human input. However, solving the problem may require solving a number of difficult sub-problems: (P1) finding a navigable route through space from A to B; (P2) determining whether it is possible for the robot to move along the path without any part of the extended robot shape colliding with any hazards in the static environment; (P3) making decisions in real time to (P4) avoid collisions with dynamic obstacles, such as pedestrians or vehicles whose motion are inherently uncertain; (P5) determining the motion commands so the robot will move as planned; (P6) to satisfy human users, by ensuring that the motion is efficient, safe, and comfortable (even "graceful"). Note that some of the sub-problems may have conflicting objectives, such as constraining the motion command vs. quickly avoiding dynamic obstacles. Due to conflicting sub-objectives and high complexity, the full problem of navigation remains a difficult challenge. Thus, a typical approach has been to focus on solving one or two of the subproblems.

In a static or quasi-static environment, planning algorithms have seen impressive improvements in the past few decades. Sampling based methods, such as rapidly exploring random trees (RRT) [1] can work in a high-dimensional continuous configuration space and finds a feasible path if it exists, given sufficient time. A recent variant, RRT* [2], can guarantee asymptotic optimality as the number of samples grows. Heuristic search like A^* is also very popular, which typically performs greedy search on grid-based maps. Good heuristics often lead to very efficient search, and majority of participants in the DARPA Urban Grand Challenge used A^* and its variants [3]. Most of these methods can be described as control-decoupled since the control problem of following the planned path is solved elsewhere. The paths found by the algorithms require non-trivial and often expensive post processing to make the path smooth and admissible to the controller. In a dynamic environment, the plan often needs to be recomputed entirely when dynamic obstacles block the path or when the target moves.

In highly dynamic environments, local and immediate motions become more important to avoid collisions, so reactive, control-oriented methods are often more desirable. Some early work includes potential field methods [4] and VFH [5], where a robot is reactively pushed away from obstacles and pulled toward the goal. With the potential field, robot motion can be computed everywhere in the map very quickly, as the field is a property of the map that is computed independently. Navigation function based methods [6] and gradient methods [7] can eliminate local minima associated with naive potential fields, but it may be expensive and difficult to compute the field in the full configuration space of the robot. Other reactive methods, such as DWA [8] or VO [9], search the velocity space directly and choose velocities that are guaranteed to avoid collision. VO can guarantee the motions to be collision-free given perfect knowledge of dynamic obstacles. With reactive methods in general, the motion commands are computed based on a single time step, so the resulting acceleration and jerk can be arbitrarily large which is not suitable for passenger-oriented applications.

Only limited attention has been given to planning comfortable motion. Gulati [10] provides the most comprehensive result to date, solving a full optimization problem over the entire trajectory space to minimize a discomfort metric,

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

which is computationally expensive. Other works [11] [12] are typically modifications of existing control laws to render the trajectory more comfortable.

In this paper, we view the problem of navigation as a continuous decision making process, where an agent with short-term predictive capability reasons about its situation and makes an informed decision at each time step. We introduce the model predictive equilibrium point control (MPEPC) framework which can be solved very efficiently utilizing a compact parametrization of trajectory by an analytic feedback control law [13]. We build upon a model predictive control architecture where trade-offs between progress toward the goal, quality of motion, and probability of collision along a trajectory are fully considered. The algorithm provides real-time control of a physical robot, while achieving comfortable motion, static-goal-reaching and dynamic obstacle avoidance in a unified framework.

II. MODEL PREDICTIVE EQUILIBRIUM POINT CONTROL

For our navigation algorithm, we use a receding horizon model predictive control (MPC) architecture (e.g., see [14], [15], [16] for UAV examples, and [17], [18], [19], [20] for ground vehicle examples). MPC is useful in dynamic and uncertain environments as it provides a form of information feedback, 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.

The time horizon in MPC implicitly represents the predictive capability of the planner. If the planner believes the world is static and well-known, the horizon can be extended into infinity and the plan can be generated over the entire distance, as in the case of many planners with a quasi-static world assumption. If the anticipation is very difficult then the horizon can be reduced to zero and the planner becomes reactive.

A. Traditional MPC

Typical MPC starts with a cost metric:

$$J = \int_0^T l_1(x(t), u(t), t)\,\mathrm{d}t + l_2(x_f) \tag{1}$$

where l_1 is a state-dependent cost function and l_2 is a terminal cost the estimating cost-to-go from terminal state $x_f = x(T)$ to the final destination. In MPC, the control input $u(t)$ is generally solved for as a constrained optimization problem [21]:

$$\begin{aligned}
\underset{u}{\text{minimize}} \quad & J(x,u,T) && (2)\\
\text{subject to} \quad & \dot{x} = f(x,u) && (3)\\
& x(0) = x_I && (4)\\
& u(t) \in \mathrm{U} && (5)\\
& x(t) \in \mathrm{X} && (6)
\end{aligned}$$

where (3) is the system dynamics, (4) is the boundary condition, and (5) and (6) are constraints on control and states with space of admissible controls U and space of admissible states X. For general non-linear systems, the optimization problem can be a very difficult to solve, as the cost function is not convex and the dimensionality of the problem can grow very quickly.

For example, to directly optimize a trajectory over a 5 s horizon for a non-holonomic vehicle in a structured environment with a digital controller commanding linear and angular velocities ($u = (v, \omega)^T$) at 20 Hz, the optimizer needs to search over a 200-dimensional space while satisfying complex kinematic and dynamic constraints (5) and collision constraints (6). Thus, many existing methods rely on coarse discretization of control inputs [22] to make the problem tractable, which can lead to good results (e.g. [18]) but at the expense of motion capability.

With above formulation (2), the optimizer is searching for an open-loop solution, leaving the system susceptible to fast disturbances between planning cycles. There are other sophisticated methods that try to optimize over the space of control policies $u = \pi(x)$ in order to retain the benefit of feedback [23] [24], which is also very difficult in general since the space of non-linear control laws can be very large and difficult to parametrize well.

B. Model Predictive Equilibrium Point Control

The dynamics of a stable system f_* with an equilibrium point x_*, i.e., $x \to x_*$ as $t \to \infty$, can be written as

$$\dot{x} = f_*(x, x_*) \tag{7}$$

and the trajectory of the system is determined by its initial condition and the location of the equilibrium. One way to enforce such condition is to impose a stabilizing feedback control policy

$$u = \pi(x, x_*) \tag{8}$$

which always steers the system toward $x \to x_*$, so that given the the control goal x_*,

$$\begin{aligned}
\dot{x} &= f(x,u) \\
&= f_\pi(x, \pi(x, x_*)) && (9)\\
&= f_*(x, x_*) && (10)
\end{aligned}$$

where f_π denotes the dynamics of the system under the control policy π. Note that in (10), the equilibrium essentially acts as a control input to the system, i.e., 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., [25], [26]).[1]

MPC and EPC can complement each other very nicely. We formulate the model predictive equilibrium point control

[1]The EPC framework is often inspired by equilibrium point hypothesis (EPH) [27], which a well-known hypothesis in biomechanics that locomotion is controlled by adapting the equilibrium point of a biomechanical system over time.

(MPEPC) framework by augmenting MPC with EPC, where the optimization problem can be cast as:

$$\underset{z_*=(x_*,\zeta)}{\text{minimize}} \quad J(x,x_*,\zeta,T) \tag{11}$$

$$\text{subject to} \quad \dot{x} = f_{\pi_\zeta}(x,\pi_\zeta(x,x_*)) \tag{12}$$

$$x(0) = x_I \tag{13}$$

$$x(t) \in \mathrm{X} \tag{14}$$

where x_* is the target equilibrium point of the system under a stabilizing control policy π_ζ where ζ represents manipulatable parameters in π_ζ (e.g., control gains).

For robot navigation with MPEPC, x_* is a target pose, or a *motion target*, in the neighborhood of the robot. The robot 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.[2]

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, semantically meaningful, and continuous parametrization of good local trajectories. Therefore, we can search over the space of z_*, or equivalently, the *space of trajectories parametrized by x_* under the control policy π_ζ*, as opposed to searching for the entire sequence of inputs u. We retain the benefit of feedback, which ensures more accurate prediction and robustness to disturbances. Control constraints (5) do not appear because it can be incorporated into design of $\pi_\zeta(x,x_*)$.

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 [13] we developed a feedback control law (15) which guides a robot from an arbitrary initial pose to an arbitrary target pose in free space, which allows us to formulate the navigation problem within the MPEPC framework. We introduce the robot system and the control policy in section III. The detailed description of the objective function is given in section IV, where we introduce probabilistic weights and absorb collision constraints (14) to the cost definition, formulating local navigation as a continuous, low dimensional and unconstrained optimization problem which is easy to solve. Results are shown in section V.

III. SYSTEM OVERVIEW

A. The Wheelchair Robot

Our primary target application is an assistive wheelchair. We want to design an intelligent wheelchair capable of safe and autonomous navigation, which can provide necessary level of motor assistance to individuals suffering from physical and mental disabilities. We require the motion to be safe and comfortable, avoiding all collisions and limiting the velocity, acceleration and jerk. In our lab, we have a differentially-driven wheelchair robot equipped with a laser range finder that builds local metrical and global topological maps [28] (Fig. 2, Left).

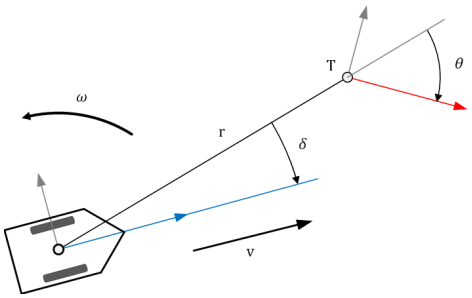

Fig. 1. Egocentric polar coordinate system with respect to the observer. Here, both θ and δ have negative values.

B. Feedback Control Policy and Trajectory Parametrization

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. 1.

In [13], it is shown that linear velocity v and angular velocity ω satisfying the following control law (15) will globally drive the robot toward a target at $(r,\theta,\delta)^T$ by singular perturbation [29].

$$\omega = -\frac{v}{r}[k_2(\delta - \arctan(-k_1\theta)) + (1+\frac{k_1}{1+(k_1\theta)^2})\sin\delta] \tag{15}$$

(15) describes a generalized pose-following control law with shape and gain parameters k_1 and k_2.[3] The path curvature κ resulting from the control law is

$$\kappa = -\frac{1}{r}[k_2(\delta - \arctan(-k_1\theta)) + (1+\frac{k_1}{1+(k_1\theta)^2})\sin\delta] \tag{16}$$

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:

$$v(\kappa) = v(r,\theta,\delta) = \frac{v_{\max}}{1+\beta|\kappa(r,\theta,\delta)|^\lambda} \tag{17}$$

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 $\kappa \to 0$.

[2]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.

[3]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.

It can be shown that all velocities, acceleration and jerks are bounded under (15) and (17) 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

$$v = \min(\frac{v_{\max}}{r_{\text{thresh}}} r, v(\kappa)) \tag{18}$$

with some user-selected distance threshold r_{thresh}, the target T becomes a non-linear attractor that the vehicle will exponentially converge to,[4] 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

$$\begin{aligned} z_* &= (x_*, v_{\max}) \\ &= (r, \theta, \delta, v_{\max})^T \end{aligned} \tag{19}$$

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*, (15) - (18). 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 trajectories parametrized by z_* are smooth and realizable (by the controller) by construction; e.g., the constraint (5) is satisfied for any motion target x_* at the control level.

IV. MPEPC NAVIGATION

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 satisfying comfort and collision constraints. Formally, let

$$q_{z_*} : [0, T] \to C \tag{20}$$

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 $q_{z_*}([0,T])$:

$$E[\phi(q_{z_*})] = E[\phi_{\text{collision}}] + E[\phi_{\text{action}}] + E[\phi_{\text{progress}}] \tag{21}$$

where $\phi_{\text{collision}}$ is the cost of collision, ϕ_{action} is the cost of action, and ϕ_{progress} is the negative progress.[5] We introduce the cost definition in the following subsections.

[4]With (18), $v = v(\kappa)$ when $r > r_{\text{thresh}}$ since $v(\kappa) \le v_{\max}$. Also, $v \sim r$ near $r = 0$ canceling $1/r$ in (15) and rendering the system exponentially stable. Proof shown in [13].

[5]We use the optimal control framework to deal with the tradeoffs between the objectives, rather than to achieve the absolute optimality.

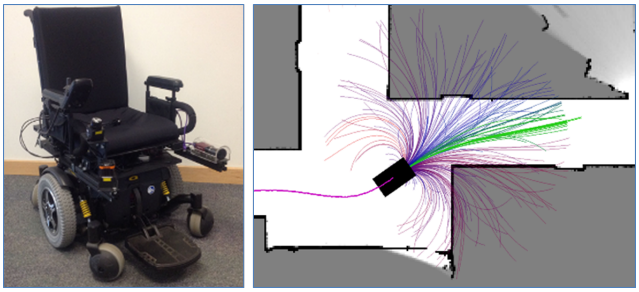

Fig. 2. (Best viewed in color) **Left**: Differentially driven wheelchair robot (0.76m x 1.2m), equipped with laser range finder and SLAM capability. **Right**: Sampled trajectories parametrized by z_* (19), generated by a dynamically simulated vehicle operating under the stabilizing control policy (15) - (18) in a SLAM-generated map. The wheelchair is depicted as a black rectangle. Green to red color gradient indicates low to high expected navigational cost (21). See text for details.

A. 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 [30] [31], 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:

$$p_c^i(d_i(t), \sigma_i) = exp(-d_i(t)^2/{\sigma_i}^2) \tag{22}$$

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, and σ_i is an uncertainty parameter. We treat the static structure as a single object in the environment.

With (22), we can define the *survivability* $p_s(t)$ of the robot over $q_{z_*}([t, t+\Delta t])$:

$$p_s(t) \equiv \prod_i^m (1 - p_c^i(t)) \tag{23}$$

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.

B. 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[6]

$$\phi_{\text{progress}}(t) = NF(q_{z_*}(t+\Delta t)) - NF(q_{z_*}(t)) \tag{24}$$

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

So that with (23) and (24), we can write *expected negative progress* $E[\phi_{\text{progress}}]$ over the static plan $NF(\cdot)$ which needs to be minimized as

$$E[\phi_{\text{progress}}] \equiv \sum_{j=1}^{N} p_s(j) \cdot \phi_{\text{progress}}(j) + c_1 \Delta\theta(q_{x_*}(T)) \quad (25)$$

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 normalizer and does not affect the optimization process. $NF((q_{x_*}(T))$ is the terminal cost (which corresponds to $l_2(x_f)$ in (1)) we want to minimize, which is the underestimated cost-to-go to the final destination encoded by $NF(\cdot)$.

Similarly, with (22), we can write the *expected cost of collision* as

$$E[\phi_{\text{collision}}] \equiv \sum_{j=1}^{N} \sum_{i}^{m} p_c^i(j) \cdot c_2 \phi_{\text{collision}}^i(j) \quad (26)$$

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:

$$\phi_{\text{action}} = \sum_{j=1}^{N} (c_3 v^2(j) + c_4 \omega^2(j)) \Delta t \quad (27)$$

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 (25), (26), and (27), as given in (21). In light of (11) and (19), we have

$$\begin{aligned} J(x, z_*, T) &= E[\phi(q_{z_*})] \\ &= E[\phi_{\text{collision}} + \phi_{\text{progress}} + \phi_{\text{action}}] \end{aligned} \quad (28)$$

The probability weights p_c^i and p_s (22) - (23) can be understood as a mixing parameter which properly incorporates the collision constraint (14) into the cost definition. The use of expected values 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 (28), the optimization problem (11) is now in continuous, low-dimensional and unconstrained form.

V. RESULTS AND DISCUSSION

A. Implementation with Pre-sampling for Initial Condition

We have used off-the-shelf optimization packages[7] 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. We pre-sample the search space with a dozen selected samples, including the optimal target from the previous time step, stopping, and typical soft/hard turns.[8] The best candidate from the pre-sampling phase is passed to a gradient-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).

B. Experiments

We tested the algorithm in two typical indoor environments: a tight L-shaped corridor, and an open hall with multiple dynamic objects (pedestrians). In the L-shaped corridor, the proposed algorithm allows the robot to exhibit wide range of reasonable motions in different situations, e.g. to move quickly and smoothly in an empty corridor (Fig. 4), and to stop, start and trail a pedestrian in order to progress toward a goal without collision (Fig. 5). In the hall environment, we show that the algorithm can deal with multiple dynamic objects, and that changing a weight in the cost definition can result in qualitatively different robot behavior in the same environment (Fig. 7 - 8).

All sensor data are from actual data traces obtained by the wheelchair robot. The position and velocity of dynamic objects are tracked from traces of laser point clusters, and the planner estimates the motion of the dynamic objects using a constant velocity model over the estimation horizon $[0, T]$. Robot motion is dynamically simulated within the environments generated from the sensor data.

Simulated navigation results in a tight L-shaped corridor are shown in Fig. 4 - 6. The robot has the occupancy grid map, the navigation function, and the proximity to static obstacles as the information about the static world (Fig. 3). The planner searches for the optimal trajectory parameterized by z_* in the neighborhood of the robot bounded by $0 \leq r \leq 8$, $-1 \leq \theta \leq 1$, $-1.8 \leq \delta \leq 1.8$ and $0 \leq v_{\max} \leq 1.2$. 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. 4 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),

[7] *fmincon* in MATLAB and *NLopt* in C++.

[8] Pre-sampling with likely movements seems to be more effective than random sampling.

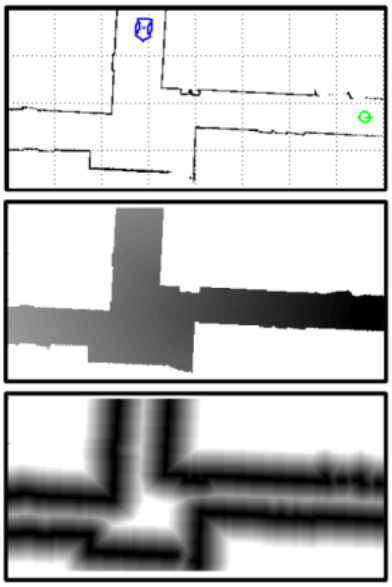

Fig. 3. The information about the static world available to the robot. **Top**: Occupancy grid map (20×10 m) with initial robot pose (blue) and the final goal (green circle). **Middle**: Navigation function. **Bottom**: Proximity to walls.

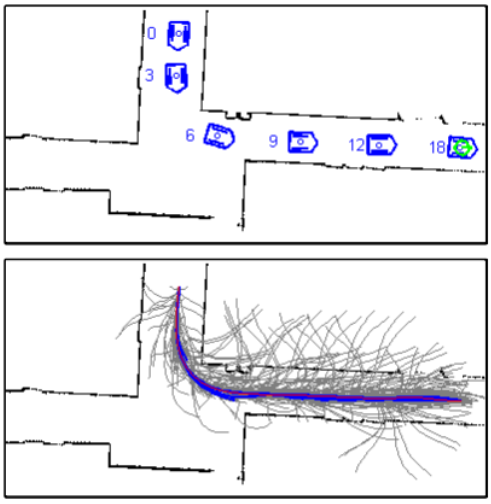

Fig. 4. (Best viewed in color) **Top**: Time-stamped snapshots of the motion of the robot (0.76×1.2 m) in a L-shaped corridor with a small opening (2 m wide, smallest constriction at the beginning 1.62 m). The robot moves quickly and smoothly in a free corridor. See Fig. 6 for velocities and accelerations along the trajectory. **Bottom**: Trajectories sampled by the planner (gray), time-optimal plans at each planning cycle (blue), and the actual path taken (red).

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 near the allowed maximum speed (1.2 m/s) (Fig. 6, Top) 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. 5, the robot runs with the same weights in the cost function but with a slow moving (~ 0.5 m/s) pedestrian in the map. The tracked location (red circle) and the estimated

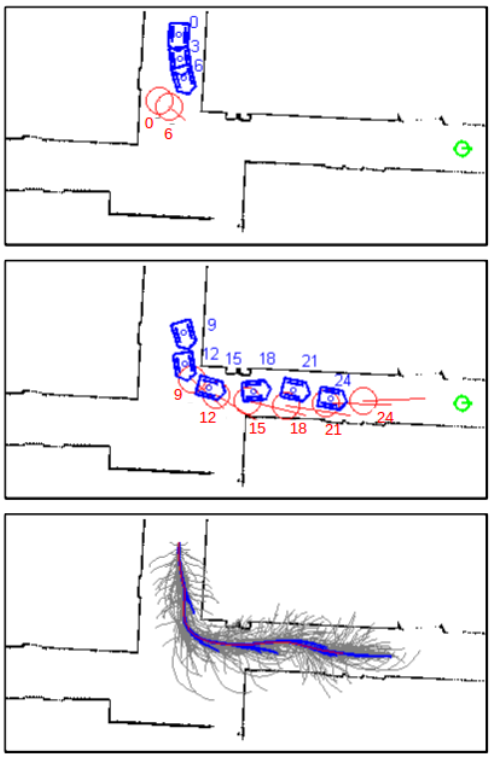

Fig. 5. (Best viewed in color) **Top**: The robot motion in 0 - 6 s. As a pedestrian (red circle) begins to move, the robot stops briefly to avoid collision (at $t \sim 6$ s). Velocities and accelerations shown in Fig. 6. Estimated trajectory of the dynamic object is shown as red lines. **Middle**: As the pedestrian moves into the corridor, the robot starts trailing the pedestrian at about the same average speed, progressing toward the goal while avoiding collision. **Bottom**: Trajectories sampled by the planner (gray), time-optimal plans at each planning cycle (blue), and the actual path taken (red).

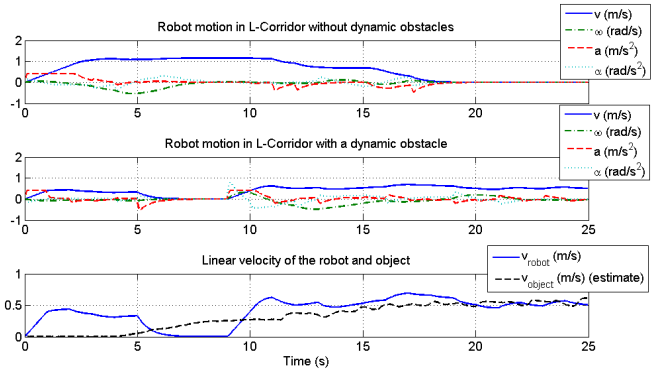

Fig. 6. (Best viewed in color) **Top**: Velocity and acceleration profiles for the robot in the free corridor. The vehicle is given a very small weight on linear velocity cost and moves close to the maximum allowed speed. **Middle**: In a corridor with a pedestrian, the robot is able to stop, start again and trail the pedestrian with the proposed algorithm, without changing any parameters. **Bottom**: Linear velocity profile of the robot and the pedestrian (dynamic object). The robot stops briefly at $t \sim 6$ s as the pedestrian blocks the way, and trails the pedestrian nearly at the same speed as it moves.

trajectory (red line) of the dynamic object is shown. In order to progress toward the goal without collision, the robot effectively trails the pedestrian, stopping and restarting as needed. A comparison of the robot speed and the pedestrian speed is shown in Fig. 6, Bottom.

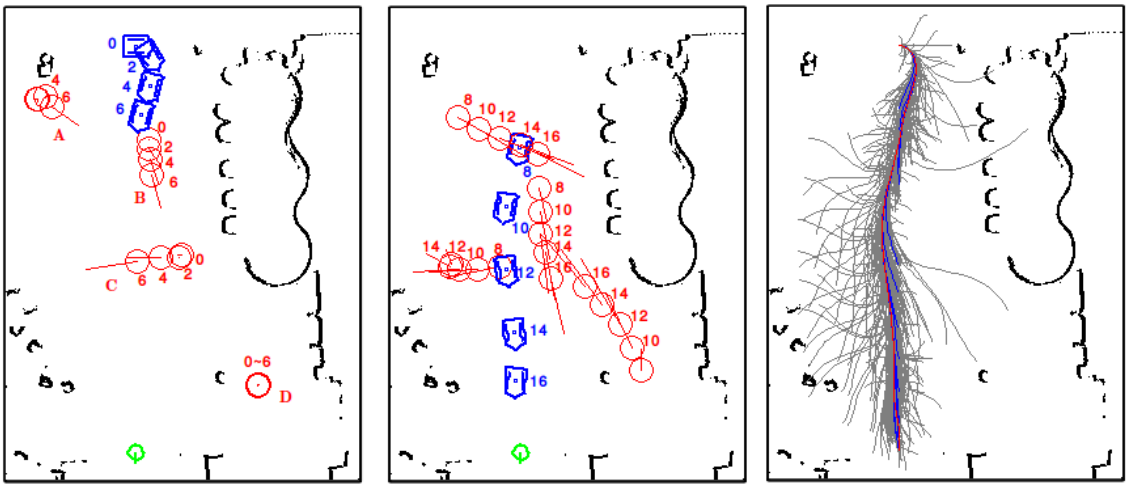

Fig. 7. (Best viewed in color) In an open hall (13.5×18 m) with multiple pedestrians (A-D, red circles), the robot (blue) estimates the trajectories of dynamic objects over the planning horizon (red lines) and navigates toward the goal (green circle) without collision. Compared to Fig. 8 the robot moves slower due to larger weights on the action cost ($c_3 = 0.4$ and $c_4 = 0.2$). **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).

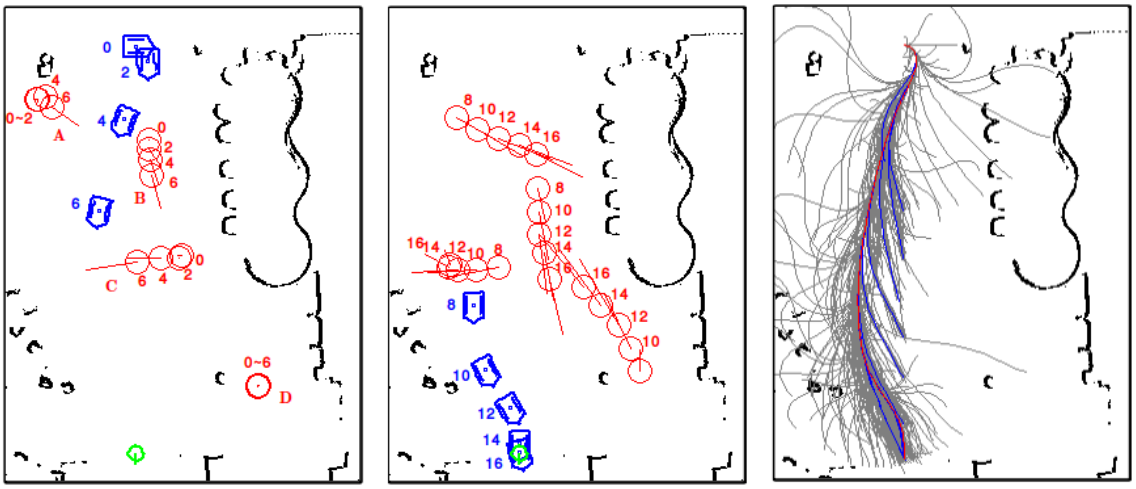

Fig. 8. (Best viewed in color) Robot motion in the same environment as Fig. 7, but with low weights on the action cost ($c_3 = 0.04$ and $c_4 = 0.02$). The robot moves more aggressively and cuts in front of the dynamic object marked C, which it passed behind in Fig. 7. **Left and Center**: Time-stamped 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).

In Fig. 7 - 8, robot motion in an open hall with multiple dynamic objects is shown. For these examples, the bound on linear velocity gain is increased to $v_{max} \leq 1.9$ m/s. In Fig. 7, the weights for the linear and angular velocity cost is set higher ($c_3 = 0.4$ and $c_4 = 0.2$) so the robot prefers to move slower, moving behind the pedestrian C as it passes and then speeding up. In Fig. 8, with a small weight on the linear velocity cost ($c_3 = 0.04$ and $c_4 = 0.02$) the robot is more aggressive and does not slow down, cutting in front of the slow-moving pedestrian C. The trajectories sampled by the planner (gray), time-optimal plans at each planning cycle (blue), and the actual path taken (red) are shown on the right.

As can be seen in the examples, the algorithm can generate suitable trajectories across a range of situations exhibiting very reasonable behavior. We note that in realistic setting, for an agent with limited actuation capability (non-holonomic and dynamic constraints) and without perfect knowledge of the environment, it is not possible to guarantee absolute collision avoidance. However, the algorithm still tries to maximize expected progress while minimizing the effort and probability of failure with the information and predictive capability available.

The overall algorithm is very efficient and achieves real-time performance (P3) since the search space is small and does not require any post-processing to compute the control inputs. In effect, the MPEPC framework allows the planner to search simultaneously in the configuration space and in the control space, as it operates within the space of trajectories parametrized by a feedback control law; thus (P5) is trivially satisfied. With MPEPC, it is possible to benefit from desirable properties in the control law, which in our case is smoothness and comfort in motion [13]. The weights

in the cost definition provide a way to handle the trade-offs between competing sub-objectives, and a way to shape the behavior of the robot (P6).

We have a hybrid model of the environment in the sense that we explicitly differentiate between the static and dynamic parts of the environment. This factoring allows us to quickly identify a navigable route in the static environment in the form of navigation function (P1), which provides an approximate cost-to-go to the goal from any part of the navigable space. Collision with static and dynamic obstacles are checked in real time within the local neighborhood of the configuration-time space in a unified framework (P2), (P4). 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.

VI. 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 (15) - (18) and a probabilistic cost definition (21) for trajectory optimization in the MPEPC framework (11) - (14), the proposed algorithm achieves real time performance and generates very reasonable robot behaviors.

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.

VII. ACKNOWLEDGMENT

The authors would like to thank Grace Tsai for helpful discussions and valuable suggestions.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.