Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Receding Horizon Trajectory Planning with an Environment-Based Cost-to-go Function

Bernard Mettler and Olivier Toupet
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts, MA 02139
Email: bmettler@mit.edu

Abstract— **This paper presents a general framework for trajectory planning in three dimensions for vehicles with broad maneuvering capabilities operating in continuously evolving environments. The approach combines an online receding horizon trajectory optimization and a cost-to-go function computed offline that approximates the performance of the vehicle in the global environment and provides the terminal cost for the receding horizon optimization. This breakdown permits a computationally tractable implementation. To enable an efficient computation of the cost-to-go function, a finite dimensional decomposition of the global environment is used. The paper describes how this decomposition is set up to compute the cost-to-go function, as well as its integration with the online receding horizon trajectory optimization. An example is used to demonstrate the system's basic capabilities. This approach combines key results from robotics motion planning and vehicle trajectory optimization.**

I. Introduction

Small-scale autonomous vehicles are expected to operate in intricate 3D environments that often will only be known to a limited level of details. The knowledge about the environment, and even about the task is expected to evolve as the mission unfolds. Trajectory planning for dynamical systems is typically formulated as an optimal control problem since the dynamics of the vehicle need to be accounted for explicitly. These problem however are computationally too complex to be solved in real time. Solving for the entire trajectory online in its full details, accounting for the vehicle dynamics, and the environment is often not meaningful since a large portion of the trajectory depends on details that are subject to large uncertainties. However, the global mission and environment still need to be accounted for so that the trajectory in the immediate environment can be generated.

Receding horizon optimization represents an attractive framework for trajectory planning. In receding horizon trajectory optimization, the online computational resources are used to solve a finite horizon trajectory optimization problem based on the current vehicle state and immediate environment knowledge. Beyond the optimization horizon, the planning problem is approximated by a cost-to-go (CTG) function. The CTG function, can be computed offline and updated when needed (when task or global environment knowledge changes). In this way, receding horizon trajectory planning provides a way to manage the computational complexity by breaking down the large problem into two problems. For many situations this break down is natural due to difference in time-scale between the immediate trajectory requirements (order of seconds) and the long-term ones (minutes). It is also attractive from a computational standpoint since it provides a way to leverage both the on- and offline computational resources.

Receding horizon control became popular in process control because it provides a way to explicitly handle hard control state constraints as well as nonlinear and time-varying plants [8]. Also, re-solving for the control action online, instead of using a pre-computed control law, helps compensate for uncertainties and disturbances. More recently, in part due to the steady increase in computational power, the domain of application is expanding to include mechanical systems and even vehicles. A key questions for a successful implementation is how to properly account for the discarded "tail" of the optimization. Several techniques have been proposed to handle the stability issues arising with finite optimization horizons [8]. The most general one is to approximate the discarded tail by an appropriate cost-to-go function. Jadbabaie *et al.* [5] have shown that stability of the control scheme can be guaranteed by using a control Lyapunov function (CLF) which is an "appropriate" upper bound on the CTG. They demonstrated their approach on an experimental thrust-vectored flying wing test-bed [4].

Like for the control of nonlinear systems, the performance and stability of the trajectory planning system depends on an appropriate choice of CTG function. Existing techniques to compute CTG functions for trajectory planning are mostly based on the visibility graph approach introduced by Bellingham *et al.* [1]. In this approach, the 2D environment is represented using polygonal obstacles and the path is approximated by the edges of the visibility graph constructed for a particular goal location and obstacle configuration. The minimum-time CTG function is then computed using a shortest path algorithm. To account for the vehicle performance, the path length is divided by the maximal vehicle speed; a penalty is used when successive edges involve a change in direction. This particular technique has recently been extended to three dimensions [7]. The main shortcoming of the visibility graph approach is that it is based on environments described by polygons and that the path can only take into account limited vehicle maneuvering capabilities.

In [9], we introduced an approach to compute a cost-to-go function for trajectory planning from a decomposition of the environment based on the vehicle maneuvering capabilities. The decomposed environment provides a way to incorporate various elements of the environment and mission, such as terrain features and threats, as well as path visibility considerations. The goal of our work is to find a way of representing the environment which is conducive to effective trajectory planning algorithms. In the following we describe the use of multi-resolution decomposition of the environment that enable to account for a broader vehicle performance range in the CTG computation, as well as a more consistent approximation of the CTG with respect to the finite horizon objective.

We first provide a short background on the formulation of the trajectory planning problem using receding horizon optimization. Then we describe the technique used to compute the CTG function, first we explain how we perform the environment decomposition and second how we use it to compute the CTG function. Then we describe the receding horizon trajectory planner and its integration with the CTG function. To demonstrate the capabilities of the trajectory planning system we present simulation results. Finally, we provide conclusions and an outline of our ongoing work.

II. Background: Receding Horizon Trajectory Planning

The trajectory planning can be formulated as a constrained nonlinear optimization problem. The cost function $J(\mathbf{x})$ typically captures desired behavior such as minimizing time or fuel. The constraints typically include the following: the vehicle dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$; maneuvering limitations $\mathbf{u} \in U$; constraints specifying the vehicle destination; and the requirement that the trajectory stays inside the obstacle- or terrain-free space $x_{pos} \in \Theta_{free}$.

This type of nonlinear optimization problems can be solved using numerical optimization methods derived from nonlinear programming (NLP). For trajectory planning applications, however, the problem is often too large to be solved in real time. Receding horizon optimization allows to dramatically cut down the computational burden by truncating the full horizon optimization to some finite length H. To account for the discarded "tail" of the optimization a terminal cost $V(\mathbf{x(t)})$ can be used. This cost captures the cost-to-go from the state reached at the end of the horizon ($\mathbf{x}(t = T)$) can be used. With this terminal cost, the optimization minimizes the composite cost which has $\mathbf{x}(t = T)$ as parameter. [IMPROVE: Accounting for the tail of the optimization is critical in trajectory planning since the terrain-free space Θ_{free} is not convex [1].]

In analogy to the control problem, the ideal cost-to-go function is the one that gives the exact cost of the discarded tail. In theory, it could be obtained by solving offline the infinite horizon optimization problem for a large number of positions covering the region of operation. This however is computationally demanding. Moreover an accurate solution may not be required for the CTG. In the following we describe our approach to efficiently computing an approximate CTG function which takes into account the key performance tradeoffs that govern how effectively a vehicle can negotiate an environment.

III. Environment Decomposition and CTG Computation

Environment decomposition is common to many robot motion planning techniques [3], [6], [13]. The basic working principle of these approaches is to decompose the environment into obstacle free cells and then search the graph representing the connectivity of the cells for the shortest sequence of edges that link the cell containing the current vehicle position to that containing the destination. Our CTG computation technique is inspired from this general principle but attempts to incorporate sufficient information about the vehicle motion so that the resulting CTG function is an appropriate terminal cost for a receding horizon trajectory optimization.

A. Multi-scale environment decomposition

In order to be able to compute a CTG function we need to establish a link between geometrical environment characteristics and some measure of performance. For many ground and aerial vehicles the main tradeoff that determines how quickly it can travel through terrain is how its ability to maneuver (e.g. turn) is affected by speed; typically, the faster the speed the more space the vehicle needs to maneuver. This principle comes from the fact that inertial forces scale with the speed squared, and that vehicles can only sustain a limited load. Therefore, traveling through regions with obstacles or intricate spaces requires more time (and vice versa). This information can then be used for the computation of the value function. In this paper we use a vehicle that behaves according to a fixed-wing aircraft. For this type of vehicle, the main maneuvering tradeoffs are: 1) the minimum turn radius vs. airspeed (R_{turn} vs. V); and 2) maximum flight path angle vs. airspeed (γ vs. V).

The key principles on which the decomposition is based are: (i) if the decomposition yields cells in an area then there must exist a feasible trajectory through this area; (ii) the decomposition should give an indication of the performance of the vehicle through that area with respect to the mission objective. We use 1) to determine the horizontal cell size d_{xy}, and 2) to determine the vertical cell size d_z. If we set d_{xy} to the turn radius R_{turn} then (i) will be satisfied since trajectories at the nominal performance level will fit in any decomposition based on d_{xy}. To be able to represent different levels of performance we use different levels of resolution. The multi-resolution representation also reduces the amount of data needed to model the environment.

The first step in the decomposition is to sample the terrain with a regular grid that corresponds to the highest resolution (smallest cells). The sampled data is then used to generate the level 1 obstacle-free cells. The level 2 cells are generated by clustering four adjacent, obstacle-free level 1 cells. Level 1 cells directly adjacent to terrain and obstacles cannot

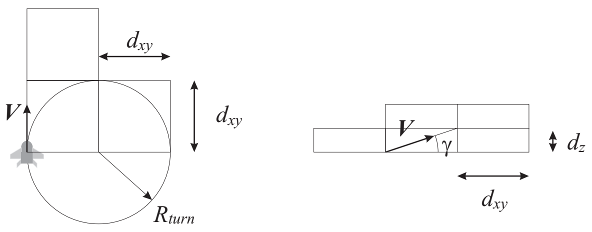


Fig. 1. Horizontal d_{xy} and vertical d_z cell dimensions based on vehicle maneuvering limits (turn radius: R_{turn} and flight path angle γ) for a characteristic speed V.

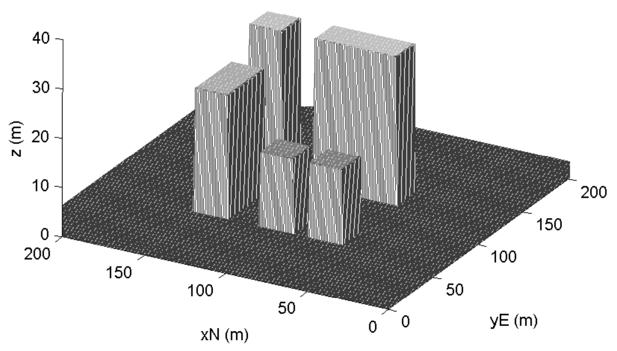


Fig. 2. Sample environment of a city block with five buildings covering an area of 200 by 200 meters.

be clustered, this way we create a boundary layer around environment features.

The smallest achievable turn radius R_{turn} at airspeed V is determined by the maximum load factor n_{max} [12]. Therefore horizontal cell size is

$$d_{xy} = R_{turn} = V^2/(g\sqrt{n_{max}^2 - 1}), \tag{1}$$

and the vertical spacing d_z is based on the flight path angle γ and the horizontal spacing d_{xy}:

$$d_z = d_{xy} * \tan\gamma. \tag{2}$$

Figure 1 shows the horizontal and vertical cell sizing based on the vehicle turn radius R_{turn} and flight path angle for a given speed V.

Figure 2 shows a synthetic 3D environment representing a city block, which we use to illustrate our algorithm. In our example, we use two planar resolution levels of d_{xy} =10 and 20 meter edges, and layers of d_z =5 meter, respectively. These are sufficient to resolve the city block environment. For a maximum load factor of $n_{max} = \sqrt{3}$ (based on typical small-scale helicopter capabilities [10]), according to Eq. III-A, the characteristic speeds for level 1 and level 2 cells are V_1 = 14m/sec and V_2 = 20m/sec respectively.

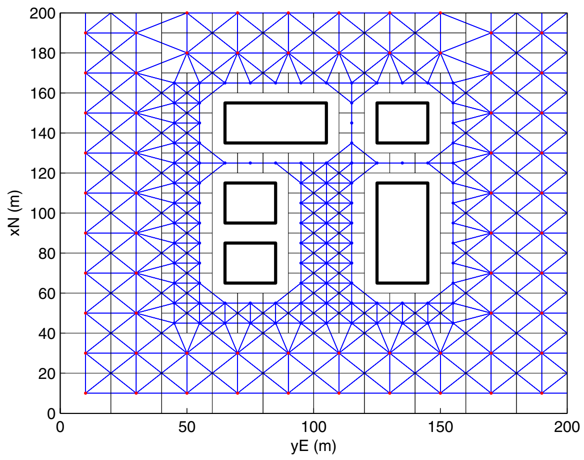


Fig. 3. Graph showing connectivity of cells for the first layer (z = 2.5 m) for the sample environment.

Figure 3 shows the graph corresponding to the decomposition for the first layer (cell centers elevation z = 2.5 meters). The red dots represent the level two cells and the blue dots the level 1 cells. All node connections are shown by the blue vertices. Notice how the area surrounding the buildings is populated by small cells and how the wider free space by large cells. A path in the cell environment is represented by a sequence of edges.

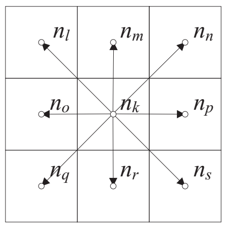
B. CTG Function Computation

The CTG function $f(n_k)$ provides the cost-to-go from every node point n_k in the environment graph to an *a priori* designated destination node n_t. For example, for a minimum-time performance objective, the cost-to-go function gives the travel time to the destination node.

For the purpose of computing the CTG function, the vehicle motion in the environment graph is described through node transitions. The vehicle can transition from each node n_k to all its directly adjacent nodes . For each node, the set of adjacent nodes $S_A(n_k)$ is provided by the adjacency matrix of the environment graph. Its configuration is function of the local environment (e.g. see Figure 3). Figure 4 shows the nodes (and cells) that are adjacent and level to n_k for a single resolution without terrain or obstacle encroachment. It also shows the corresponding transition graph. The transition between the node n_k and a node $n_l \in S_A(n_k)$ carries an incremental cost $g_{k,l}$. For the minimum time objective, $g_{k,l}$ is the travel time between the two nodes. In our example, $g_{k,l}$ is determined based on the relative configuration of n_k and n_l.

With the node transition model, we can compute the CTG function $f(n_k)$ using a shortest-path algorithm. In our example we use dynamic programming. The process involves performing a value-iteration with the following recurrence relation, based on Bellman's optimality principle [2]:

$$f(n_k) = \min_{n_l \in S_A(n_k)} [g_{k,l} + f(n_l)]. \tag{3}$$



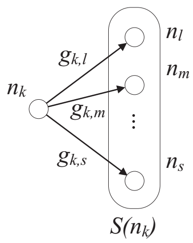


Fig. 4. Illustration of cell transitions used to model the vehicle motion in the cell environment: possible planar transitions from n_k in a free environment (left) and the corresponding node transition graph (right) showing the set of adjacent nodes $S_A(n_k)$ with the corresponding transition costs $g_{k,\cdot}$ used in the value iteration. Note that $S_A(n_k)$ is function of the environment.

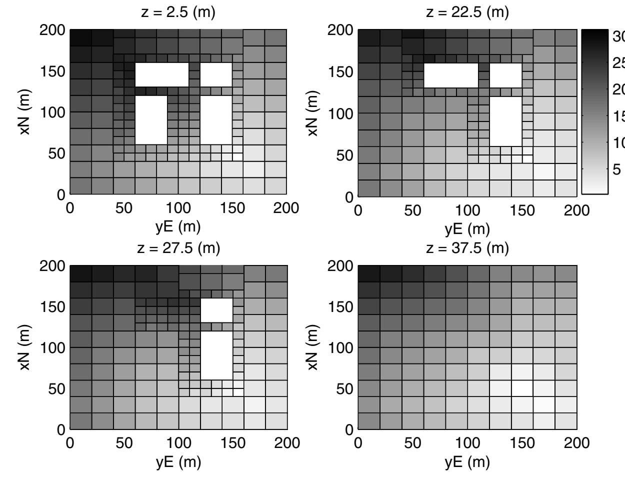


Fig. 5. Cost-to-go function for the environment shown in Fig. 2. The gray levels represent time to go in sec from cells in different layers (z) to a destination cell located at [40 160 0]

Before proceeding with the value iteration, the value function is first initialized: the value of the designated destination node n_{dest} is set to zero and all other nodes are set to infinite. The value iteration proceeds by implementing the recurrence relation of Eq. 3 across all node points. The iteration is stopped when a desired level of convergence is attained.

Figures 5 and 6 show the computed value function for the terrain example shown in Figure 2. The lighter the gray shading the smaller the time to go.

IV. Online Trajectory Planning

The online trajectory planning is performed using a receding horizon optimization, which uses the CTG function as terminal cost. The optimization is formulated as a mixed-integer-linear-program (MILP) which explicitly accounts for the vehicle dynamics and maneuvering limits. Mixed-integer-linear-programming (MILP) has been used in several application of receding horizon optimization to trajectory planning [1], [7]. The formulation is restricted to piecewise linear objectives and constraints but the use of integer

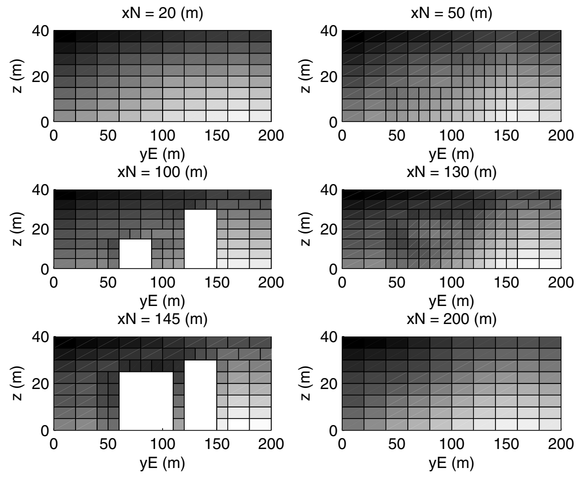


Fig. 6. Cost-to-go function for the environment shown in Fig. 2. The gray levels represent time to go in sec from cells in different cross section (xN) to a destination cell located at [40 160 0]

variables makes for versatile modeling capabilities.

A. Receding Horizon Planner with Terminal Cost

The online trajectory planner generates a continuous trajectory to one of the CTG points. The point that is selected, and the trajectory that leads to it from the current vehicle location and state, is the one that minimizes the cost to the destination. The cost to reach the destination has two components: the cost-to-go from the current location to one of the CTG points; and the cost-to-go from that point to the designated destination. The former is formulated as a linear objective, the latter is obtained directly from the CTG function. At regular time intervals T_s, the optimal command is implemented, and the optimization process is repeated as the vehicle advances along the planned trajectory. As the vehicle evolves through the environment, the RHP is generating a continuous trajectory to the destination by planning a local trajectory that successively uses new terminal CTG points

Note that the “best” CTG point may not be the one with the lowest cost-to-go value, but the one that minimizes the composite cost combining the time to go from the current vehicle position and state, and satisfies the constraints. Also, as the vehicle evolves through the environment, it may actually never fly through the CTG points used to plan the trajectory. Namely, as soon as a CTG point that provides a better composite cost is reachable, the trajectory will branch toward that new point.

B. MILP Formulation

The linear objective function $J(x)$ describes the composite minimum time objective over a planning horizon of length H. The problem is formulated in discrete time with time steps k

$$J = \sum_{k=0}^{H} [b_{arv}[k] k T_s + T_{ctg}]. \tag{4}$$

The two components of the objective are: the trajectory time from the current vehicle location to the CTG point $\mathbf{x}_{ctg}$ selected by the planner; and the time to go from that point to the final destination (given by the cost-to-go value T_{ctg} at that point). The trajectory time is modeled using the minimum time logic formulation, with the vector of binary variables $\mathbf{b}_{arv}[k]$, described in [11]. The entry $\mathbf{b}_{arv}[k] = 1$ at the time step k corresponding to the trajectory time kT_s.

The CTG data are provided to the planner in the form of a set of CTG points $S_N(\mathbf{x})$ in some neighborhood of the current vehicle position $\mathbf{x}$. The data includes the coordinates of the CTG points $\mathbf{X}_{ctg}$, the corresponding cost-to-go values T_{ctg}, and the gradient of the CTG function $\mathbf{V}_{ctg}$. A vector of binary variables $\mathbf{b}_{ctg}[i]$ is used to specify the active CTG point. The length of the vector corresponds to the number of points N in the CTG set $S_N(\mathbf{x})$. The active CTG point coordinates $\mathbf{x}_{ctg}$ is designated by

$$\begin{aligned} \mathbf{x}_{ctg} - \mathbf{X}_{ctg}[i] &\leq M(1 - b_{ctg}[i]) \\ -\mathbf{x}_{ctg} + \mathbf{X}_{ctg}[i] &\leq M(1 - b_{ctg}[i]). \end{aligned} \tag{5}$$

To ensure that only one cell is used, the following constraint is enforced:

$$\sum_{i=1}^{n} b_{ctg}[i] = 1. \tag{6}$$

The same logic is used for the active CTG value T_{ctg}

$$\begin{aligned} T_{ctg} - \mathbf{T}_{ctg}[i] &\leq M(1 - b_{ctg}[i]) \\ -T_{ctg} + \mathbf{T}_{ctg}[i] &\leq M(1 - b_{ctg}[i]). \end{aligned} \tag{7}$$

At the selected CTG point, the vehicle has to satisfy the optimal state of the CTG function. Our CTG function does not explicitly account for the vehicle state, however, the heading and the velocity can be approximated from the gradient of the CTG function. Note that this is possible here since the CTG function is defined over a grid of discrete points. This latter condition is ensured by constraining the cross product of the vehicle speed and the gradient of the CTG function to be zero at the selected cell (collinearity), and by constraining the scalar product of these two vectors to be nonnegative (same direction). The corresponding set of constraints formulated in the inertial frame are:

$$\begin{aligned} \mathbf{v}(k) \times \mathbf{v}_{ctg}[i] &\leq M(1 - b_{ctg}[i]) + M(1 - b_{arv}[k]) \\ -\mathbf{v}(k) \times \mathbf{v}_{ctg}[i] &\leq M(1 - b_{ctg}[i]) + M(1 - b_{arv}[k]), \end{aligned} \tag{8}$$

and

$$-\mathbf{v}(k) \cdot \mathbf{v}_{ctg}[i] \leq M(1 - b_{ctg}[i]) + M(1 - b_{arv}[k]), \tag{9}$$

where $\mathbf{v}_{ctg}[i] = \mathbf{V}_{ctg}[i]$. Since the gradient is a parameter of the optimization problem, both constraints are linear.

The remaining constraints used to formulate the trajectory optimization are the ones enforcing the vehicle dynamics and maneuvering limitations. For the demonstration example a relatively simple inertial frame model is used. It accounts for the dynamics as a first order equation of motion with airspeed bounds. Maneuvering limitations are captured by acceleration limits (see [11] for details).

C. Stability and Performance of the RH Planner

The stability and performance of the trajectory planning system depends on several factors. First, the cost-to-go function should exhibit no local minima, otherwise the planner may get trapped. Second, the cost-to-go points should be sufficiently dense, respectively, the reachable region of the RHP should be sufficiently large, so that multiple CTG points are reachable. In our implementation, we extract a nominal set of CTG points which we prune using heuristics. The nominal set has 5×5point in the horizontal plane and has 5 points in the vertical direction, for a total of $N = 125$. Pruning the set of CTG points is key to the computational performance of the online planner since each point requires a binary variable. Our horizon was set to $H = 8$ steps. Also, the cost-to-go function should not "outperform" the online trajectory planner otherwise the local trajectory may not find a competitive CTG point needed to converge to the goal.

The composite cost at a given state x, with respect to the selected CTG point x_{ctg} and corresponding cost $C_{ctg}(x_{ctg})$ has the form:

$$J(x, x_{ctg}) = C(x, x_{ctg}) + C_{ctg}(x_{ctg}), \tag{10}$$

where $C(x, x_{ctg})$ is the cost of the continuous trajectory generated by the planner from the current position x to one of the CTG points x_{ctg}; and $C_{ctg}(x_{ctg})$ is the cost-to-go value at that point.

The CTG point x_{ctg} that is selected is the one that minimizes the composite cost for the current vehicle state and location x

$$\min_{x_{ctg} \in S_N(x)} J(x, x_{ctg}) \tag{11}$$

At some location $\bar{x}$ along the continuous trajectory $(x, x_{ctg,n})$, a new CTG point $x_{ctg,n+1}$ will provide a lower composite cost, i.e., $J(\bar{x}, x_{ctg,n+1}) < J(\bar{x}, x_{ctg,n})$. At this point the RHP trajectory branches off to $x_{ctg,n+1}$ until the next branching occurs.

The resulting trajectory is leading to the destination if the composite cost is monotonously decreasing along the trajectory. Between each CTG point switchings this is guaranteed by the RHP. At switchings we must have

$$C(\bar{x}, x_{ctg,n+1}) + C_{ctg}(x_{ctg,n+1}) < C(\bar{x}, x_n) + C_{ctg}(x_n) \tag{12}$$

At the limit, when $\bar{x} = x_{ctg,n}$, $C(\bar{x}, x_{ctg,n}) = 0$, and we have

$$C(x_{ctg,n}, x_{ctg,n+1}) < C_{ctg}(x_{ctg,n}) - C_{ctg}(x_{ctg,n+1}). \tag{13}$$

Essentially, for the switching to happen at this limit point, the RH cost must be less than the difference in cost between the new and current CTG point. The CTG function therefore must be an upper bound on the infinite horizon cost.

V. Simulation Results

To demonstrate the capabilities of the planner, we simulated trajectories for a small-scale vehicle evolving in the 3D environment shown in Figure 2. The environment has several buildings of different heights arranged in a block covering an

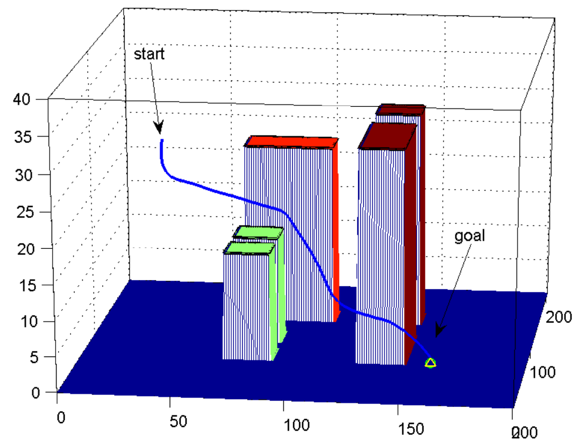


Fig. 7. Perspective view of trajectory for $\mathbf{x}_{goal} = [40, 160, 2.5]$ and $\mathbf{x}_{start} = [180, 20, 22]$.

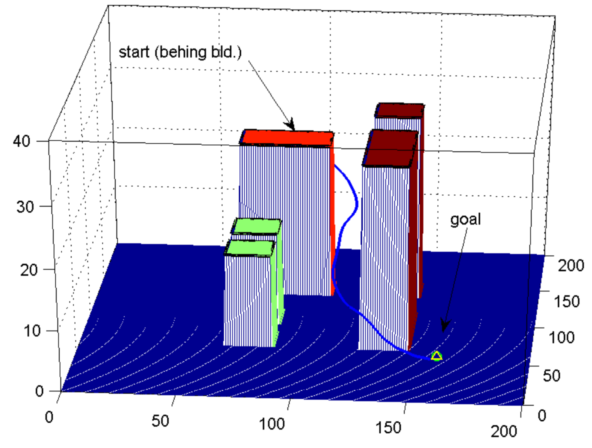


Fig. 8. Perspective view of simulated trajectory for $\mathbf{x}_{goal} = [40, 160, 2.5]$ and $\mathbf{x}_{start} = [170, 90, 20]$.

area of about 200×200 meters. For the simulations we used a Windows PC laptop. The offline environment decomposition and CTG computation takes of the order of 1 to 5seconds. The MILP trajectory optimization (using CPLEX) took on average about 5 seconds. Note that the implementation was not optimized for speed; based on our experience, these numbers are very encouraging.

Figure 7 and 8 show perspective views of the two trajectories obtained in our simulation. Both use the same goal located at $\mathbf{x}_{goal} = [40, 160, 2.5]$ with respective starting positions of $\mathbf{x}_{start} = [180, 20, 22]$ and $\mathbf{x}_{start} = [170, 90, 20]$. The CTG function for this goal was shown in Figures 5 and 6. The results, clearly demonstrate how the trajectory planner can tackle complex environments.

VI. Conclusions

The trajectory planning method described in this paper enables real-time tractable trajectory planning in complex environments for vehicle with a broad range of maneuvering capabilities.

Ongoing work focuses on refining the terrain decomposition process, value function computation, and the extraction of a minimal set of CTG points to make the receding horizon planner computationally more efficient. We are currently developing an approach that combines the terrain decomposition and value function computation into a single process. This approach uses a more detailed model of the vehicle dynamics to better capture the tradeoff between maneuvering and performance. It also explicitly takes into account the state of the vehicle. The combined process is computationally more efficient, enabling near real-time performance for the environment decomposition and value function computation. We are also developing online techniques to generate a local CTG function that can be used with onboard sensors. Finally, we are working on the development of design guidelines to help achieve a rigorous integration between the components of our planning system based on the type of vehicle capabilities, environment and mission.

VII. Acknowledgments

This work was supported in part by the Office of Naval Research Grant N00014-03-1-0171 and Boeing contract MIT-BA-GTA-1.

References

[1] J. Bellingham, A. Richards, and J. How. Receding Horizon Control of Autonomous Aerial Vehicles. American Control Conference, 2002.
[2] R. Bellman and S.E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1962.
[3] Rodney A. Brooks and Tomas Lozano-Perez. “A subdivision algorithm in configuration space for Find- Path with rotation”. *IEEE Transaction on Systems Man Cybernetics*, SMC-15(2):224–233, March/April 1985.
[4] A. Jadbabaie and J. Hauser. “Control of a thrust-vectored flying wing: a receding horizon–LPV approach”. *International Journal of Robust and Nonlinear Control*, 12:869–896, 2002.
[5] A. Jadbabaie, J. Yu, and J. Hauser. “Unconstrained Receding-Horizon Control of Nonlinear Systems”. *IEEE Transactions on Automatic Control*, pages 776–783, May 2001.
[6] S. Kambhampati and L.S. Davis. “Multi-resolution path planning for mobile robots”. *IEEE Journal of Robotics and Automation*, RA-2(3), September 1986.
[7] Y. Kuwata and J. How. “Three Dimensional Receding Horizon Control for UAVs”. Number AIAA-2004-5144. American Institute of Aeronautics and Astronautics, 2004.
[8] D.Q. Mayne, J.B. Rawling, C.V. Rao, and P.O.M. Scokaert. “Constrained model predictive control: Stability and optimality”. *Automatica*, 36(6):789–814, 1987.
[9] B. Mettler and E. Bachelder. “Combining On- and Offline Optimization Techniques for Efficient Autonomous Vehicle’s Trajectory Planning”. San Francisco, CA, August 2005. AIAA Guidance, Navigation and Control Conference and Exhibit.
[10] B. Mettler, C. Dever, and E. Feron. “Scaling and Dynamic Characteristics of Miniature Rotorcraft”. *AIAA Journal of Guidance Navigation and Control*, 27(3), January 2004.
[11] T. Schouwenaars, B. Mettler, E. Feron, and J. How. “Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles”. *Journal of Aerospace Computing, Information, and Communication*, 1, December 2004.
[12] Robert F. Stengel. *Flight Dynamics*. Princeton University Press, Princeton, New Jersey, 2004.
[13] D. Zhu and J.C. Latombe. “New heuristic algorithms for efficient hierarchical path planning”. *IEEE Trans. on Robotics and Automation*, 7(1), 1991.