AIAA Guidance, Navigation, and Control Conference and Exhibit
11-14 August 2003, Austin, Texas

AIAA 2003-5635

Stable Receding Horizon Trajectory Control for Complex Environments

John Bellingham*, Yoshiaki Kuwata†, and Jonathan How‡
MIT Department of Aeronautics and Astronautics

ABSTRACT

This paper presents a stable receding horizon controller for the minimum time trajectory optimization problem with a vehicle flying in a complex environment with obstacles and no-fly zones. The trajectory optimization is done using mixed-integer linear programming (MILP), which can directly incorporate logical constraints such as obstacle avoidance and waypoint selection and provides an optimization framework that can account for basic dynamic constraints such as turn limitations. Previous work introduced a receding horizon control that significantly reduces the computational effort for solving MILP problems. A straight line approximation used beyond the planning horizon gives a good estimate of the cost-to-go, but is shown to fail when no *kinodynamically feasible* trajectory could be constructed. A new formulation in this paper solves this problem by using a modified form of Dijkstra's algorithm to construct a path approximation that is kinodynamically feasible from the start to the goal.

With this revised path approximation and the new terminal constraints in the MILP formulation, the receding horizon MILP optimization problem is proven to have a feasible solution, which guarantees that the vehicle can reach the goal in bounded time. The simulation results show this new formulation is computationally tractable.

INTRODUCTION

The capabilities and roles of Unmanned Aerial Vehicles (UAVs) are evolving, and require new concepts for their control. In particular, further progress will require methods in planning and execution to coordinate the achievement of goals between multiple aircraft. The problem of optimizing a kinematically and dynamically constrained path is a significant aspect of controlling autonomous vehicles, and has received attention in the fields of robotics, undersea vehicles, and aerial vehicles. However, planning trajectories that are both optimal and dynamically feasible is complicated by the fact that the space of possible control actions is extremely large and non-convex, and simplifications that reduce the dimensionality of the problem without losing feasibility and optimality are very difficult.

One alternative to overcome this computational burden is to use *receding horizon control* (RHC)[1] which uses a plant model and an optimization technique to design an input trajectory that optimizes the plant's output over a period of time, called the planning horizon. A portion of the input trajectory is then implemented over the shorter execution horizon, and the optimization is performed again starting from the state that is reached. If the control problem is not completed at the end of the planning horizon, the cost incurred past the planning horizon must be accounted for in the cost function. The selection of the terminal penalty in the RHC design is a crucial factor in obtaining reasonable performance, especially in the presence of obstacles and no-fly zones.

In general, the cost function of a receding horizon controller's optimization problem estimates the cost-to-go from the selected terminal state to the goal. For vehicle trajectory planning problems in a field with no-fly zones, Ref. [2] presents a receding horizon controller that uses the length of a *path to the goal* made up of straight line segments as its cost-to-go. This is a good approximation for minimum time of arrival problems since the true minimum distance path to the goal will typically touch corners of the obstacles that block the vehicle's path. This planner using the straight line approximation is called SLC in this paper. While the SLC provides good results in practice, the trajectory design problem can become infeasible when the positioning of nearby obstacles leaves no dynamically feasible tra-

* Research Assistant, MIT **john_b@alum.mit.edu**
† Research Assistant, MIT,**kuwata@mit.edu**
‡ Associate Professor, Senior Mbr. AIAA, **jhow@mit.edu**. Room 33-328, 77 Mass. Ave., Cambridge, MA 02139.

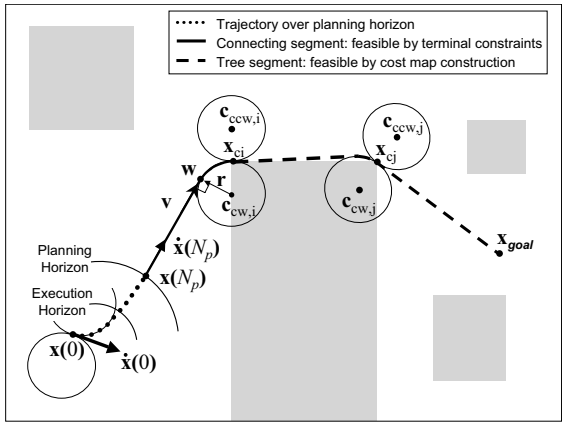

Fig. 1: Resolution Levels of the Stable Receding Horizon Trajectory Designer. Connecting and tree segments make up the path from $[\mathbf{x}(N_p)^T\ \dot{\mathbf{x}}(N_p)^T]^T$ and guarantee the feasibility of the trajectory optimization past the planning horizon to the goal. $\star$

jectory from the state that is reached by the vehicle. In this case, the vehicle cannot follow the heading discontinuities where the line segments join, since the path associated with the cost-to-go estimate is not dynamically feasible.

This paper modifies the trajectory planner to guarantee that the vehicle always reaches the goal in bounded time. It does so by evaluating the cost-to-go estimate along a *kinodynamically feasible* path to the goal. This new trajectory designer is shown to: (a) be capable of designing trajectories in highly constrained environments where the previous formulation $(\text{SLC})^2$ is incapable of reaching the goal; (b) result in minimal increase in path length over the trajectories found by that planner, and (c) be computationally tractable.

The modified trajectory designer and a method for computing a suitable cost map is given in the next section. This is followed by a *mixed-integer linear program* (MILP) form of the trajectory designer's optimization problem. Finally, the new approach is used to design trajectories in several simulations.

STABLE RECEDING HORIZON TRAJECTORY DESIGNER

In this section, the receding horizon controller stability analysis techniques are applied to prove that a modified version of the receding horizon controller is stable and reaches the goal in bounded time.

The operation of the stable receding horizon trajectory planner is shown schematically in Fig. 1. Before the trajectory optimization is performed, a tree of kinodynamically feasible paths and their lengths is found from a discrete set of N_C states $[X_c^T\ \dot{X}_c^T]^T$ to the goal, where X_c denotes a set of the positions whose costs are known, and $\dot{X}_c$ is the set of velocities consistent with the path to the goal from each position in X_c. The way to construct $\dot{X}_c$ is shown later. A state $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T \in [X_c^T\ \dot{X}_c^T]^T$ is called "a state of known cost".

In the trajectory design phase, a detailed trajectory segment over the planning horizon, from the current state to a chosen "terminal state", is optimized using a discrete-time model of the vehicle dynamics. Let N_p denote the time steps in the planning horizon, and express the terminal state on the planning horizon as $[\mathbf{x}(N_p)^T\ \dot{\mathbf{x}}(N_p)^T]^T$. The "connecting segment" connects with constant heading from $\mathbf{x}(N_p)$ to $\mathbf{w}$ along $\mathbf{v}$ (see Fig. 1), then turns at the vehicle's maximum rate over a path from $\mathbf{w}$ to $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T$. Note that this is a kinodynamically feasible path.

The allowed terminal set $\mathcal{X}_f$ includes all positions outside of obstacles, but constrains the velocity $\dot{\mathbf{x}}(N_p)$ with respect to the position $\mathbf{x}(N_p)$ such that the terminal velocity $\dot{\mathbf{x}}(N_p)$ is aligned with the constant heading of the connecting segment $\mathbf{v}$. The vehicle can then follow a "tree segment" from the precomputed set to the goal.

The connecting segment, from $\mathbf{x}(N_p)$ to a state of known cost, is discussed in the next section. This is followed by a discussion of the selection of the states of known cost and the construction of the tree of trajectories. The next step is to present the stability proof of the overall controller.

The Connecting Segment and The Terminal Constraint Set

The definition of the terminal constraint set makes use of two circles, centered at $\mathbf{c}_{\text{cw,i}}$ and $\mathbf{c}_{\text{ccw,i}}$, for every one of the N_C states of known cost $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T$, as shown in Fig 1. Two sets are defined as $\forall i, \mathbf{c}_{\text{cw,i}} \in C_{\text{cw}}$ and $\mathbf{c}_{\text{ccw,i}} \in C_{\text{ccw}}$, and $C = C_{\text{cw}} \cup C_{\text{ccw}}$ is a set of all the circle centers. These circles model the clockwise and counter-clockwise turning path of the vehicle up to $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T$. The circles have a radius of ρ, the vehicle's turning radius at maximum speed $v_{\max}$. The circle centers $\mathbf{c}_{\text{cw,i}}$ and $\mathbf{c}_{\text{ccw,i}}$ are positioned so that $\mathbf{x}_{\text{ci}}$ is on their circumference, and $\dot{\mathbf{x}}_{\text{ci}}$ is tangent to both.

$$\forall\ i \in \mathcal{C} = \{1 \dots N_C\}:$$

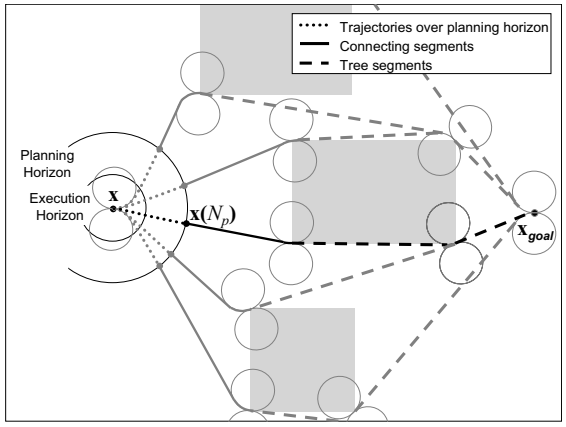

Fig. 2: Example of a tree of kinodynamically feasible trajectories to the goal. ⋆

$$\mathbf{c}_{\mathrm{cw,i}} = \mathbf{x}_{\mathrm{ci}} + \frac{\rho}{v_{\max}} \begin{bmatrix} \dot{x}_{\mathrm{ci2}} \\ -\dot{x}_{\mathrm{ci1}} \end{bmatrix} \quad (1)$$

$$\mathbf{c}_{\mathrm{ccw,i}} = \mathbf{x}_{\mathrm{ci}} + \frac{\rho}{v_{\max}} \begin{bmatrix} -\dot{x}_{\mathrm{ci2}} \\ \dot{x}_{\mathrm{ci1}} \end{bmatrix} \quad (2)$$

$\mathcal{X}_f$ is the set of states which have a velocity tangent to one of the turning circles. This property guarantees that a kinodynamically feasible connecting segment exists from all states in $\mathcal{X}_f$ to a state in $[X_c^T \ \dot{X}_c^T]^T$, and this property is expressed through constraints on the straight portion $\mathbf{v}$ of the connecting segment. The connecting segment joins the terminal state $[\mathbf{x}(N_p)^T \ \dot{\mathbf{x}}(N_p)^T]^T$ to the point $\mathbf{w} = \mathbf{c}_i + \mathbf{r}$ on the turning circle as shown in Fig. 1. Note that $\mathbf{c}$ is selected from $\mathbf{c}_{\mathrm{cw,i}}$ and $\mathbf{c}_{\mathrm{ccw,i}}$. The vector $\mathbf{v}$ is constrained to be in the direction $\dot{\mathbf{x}}(N_p)/\|\dot{\mathbf{x}}(N_p)\|$ of the terminal state's velocity and perpendicular to $\mathbf{r}$.

$$\begin{aligned} \forall (\mathbf{x}, \dot{\mathbf{x}}) \in \mathcal{X}_f, \exists\ \mathbf{c}_i \in C: & \\ \mathbf{x} + \mathbf{v} &= \mathbf{c}_i + \mathbf{r} \\ \mathbf{v} &= V \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} \quad (3) \\ \mathbf{v} \cdot \mathbf{r} &= 0 \quad (4) \\ \|\mathbf{r}\| &= \rho \end{aligned}$$

where V is the length of the straight portion $\mathbf{v}$. Note that all points on turning circles, including the states of known cost $[\mathbf{x}_{\mathrm{ci}}^T \ \dot{\mathbf{x}}_{\mathrm{ci}}^T]^T$, are in $\mathcal{X}_f$ and therefore on a feasible path towards the goal. Guaranteeing obstacle avoidance over the connecting segment is discussed later in the paper.

The Tree of Trajectory Segments

As described above, the receding horizon controller's optimization problem makes use of a tree of kinodynamically feasible trajectories from a limited set of states to the goal. An example of such a tree is shown in Fig. 2. This section presents a modified Dijkstra's algorithm that takes into account the minimum turn radius constraint to find the tree.

Minimum distance trajectories tend to connect the vehicle's starting position, vertices of obstacles, and the goal. These points are chosen as the positions X_c of the states of known cost. The velocities $\dot{X}_c$ of the states of known cost have magnitudes equal to the vehicle's maximum speed $v_{\max}$. Their direction is chosen, as the tree is being constructed, to follow the next segment in the tree, so that all states on the trajectory tree belong to $\mathcal{X}_f$. Once the velocity $\dot{\mathbf{x}}_{\mathrm{ci}}$ of a state of known cost is chosen, the circle centers $\mathbf{c}_{\mathrm{cw,i}}$ and $\mathbf{c}_{\mathrm{ccw,i}}$ are set as specified in Eqs. 1 and 2. The distance d_{i} from the i^{th} state of known cost $\mathbf{x}_{\mathrm{ci}}$ to the goal is also given by this algorithm.

Algorithm 1 is applied to grow the tree of kinodynamically feasible paths from all states in $[X_c^T \ \dot{X}_c^T]^T$ to the goal $\mathbf{x}_{\mathrm{goal}}$. When MAKE-PATH-TREE is called, the desired positions X_c for the states of known cost are passed in. MAKE-PATH-TREE divides X_c into two sets; $\mathcal{N}$, whose path to the goal has been fixed, and $\bar{\mathcal{N}}$, whose path to the goal could still be improved. At each iteration, the path of the state in $\bar{\mathcal{N}}$ with minimum distance to the goal cannot be improved, so $[\mathbf{x}_{\mathrm{cj}}^T \ \dot{\mathbf{x}}_{\mathrm{cj}}^T]^T$ is moved from $\bar{\mathcal{N}}$ to $\mathcal{N}$. This effectively fixes its path and adds it as a branch to the current tree of paths. Each position $\mathbf{x}_{\mathrm{cj}} \in \bar{\mathcal{N}}$ is then examined.

The routine FIND-CONNECTING-SEGMENT is used to find a trajectory $[X^T \ \dot{X}^T]^T$ from every $\mathbf{x}_{\mathrm{ci}}$ to state $[\mathbf{x}_{\mathrm{cj}}^T \ \dot{\mathbf{x}}_{\mathrm{cj}}^T]$. $\dot{\mathbf{x}}_0$ denotes the first velocity of the trajectory $\dot{X}^T$. As shown in Fig. 3, this trajectory has the properties of a connecting segment: it maintains a speed of $v_{\max}$ while following a tangent to the turning circle centered at $\mathbf{c}_{\mathrm{cw,j}}$ or $\mathbf{c}_{\mathrm{ccw,j}}$, then turns around this circle to reach $[\mathbf{x}_{\mathrm{cj}} \ \dot{\mathbf{x}}_{\mathrm{cj}}]$. If this path does not penetrate an obstacle, and passing through $\mathbf{x}_{\mathrm{cj}}$ shortens the path from $\mathbf{x}_{\mathrm{ci}}$ to the goal, then the distance, velocity, and circles associated with $\mathbf{x}_{\mathrm{ci}}$ are updated to correspond to this new path. The velocity $\dot{\mathbf{x}}_{\mathrm{ci}}$ is always chosen as the first velocity $\dot{\mathbf{x}}_0$ from the connecting segment.

By this construction, all states on the tree of trajectory segments are also in $\mathcal{X}_f$, because they either have a velocity tangent to a turning circle or are on

algorithm $(\mathrm{C_{cw}}, \mathrm{C_{ccw}}, \dot{\mathrm{X}}_\mathrm{c}, \mathbf{d}) = \textsc{Make-Path-Tree}(X_c, \mathcal{X}_\mathrm{obst}, \mathbf{x}_\mathrm{goal}, \dot{\mathbf{x}}_\mathrm{goal})$
  $d_1 := 0$; \\ initialize data for goal
  $\mathbf{x}_{\mathrm{c},1} := \mathbf{x}_\mathrm{goal}$; $\dot{\mathbf{x}}_{\mathrm{c},1} := \dot{\mathbf{x}}_\mathrm{goal}$;
  $\mathbf{c}_{\mathrm{cw},1} := \mathbf{x}_{\mathrm{c},1} + \rho[\dot{x}_{\mathrm{goal},2}, -\dot{x}_{\mathrm{goal},1}]^T/v_\mathrm{max}$; $\mathbf{c}_{\mathrm{ccw},1} := \mathbf{x}_{\mathrm{goal},1} - \rho[\dot{x}_{\mathrm{goal},2}, -\dot{x}_{\mathrm{goal},1}]^T/v_\mathrm{max}$;
  Set all other costs in $\mathbf{d}$ to ∞;
  $\bar{\mathcal{N}} := X_c$; $\mathcal{N} := \emptyset$;
  while $\bar{\mathcal{N}} \neq \emptyset$ **do**
    Choose the node $\mathbf{x}_\mathrm{j}$ in $\bar{\mathcal{N}}$ with minimum d_j
    Move $\mathbf{x}_\mathrm{cj}$ from $\bar{\mathcal{N}}$ to $\mathcal{N}$;
    $\textsc{Relax}(\mathrm{j}, \bar{\mathcal{N}}, \mathbf{d}, \dot{\mathrm{X}}_\mathrm{c}, \mathcal{X}_\mathrm{obst}, \mathrm{C_{cw}}, \mathrm{C_{ccw}}, \rho, v_\mathrm{max})$
  end while

procedure $\textsc{Relax}(\mathrm{j}, \bar{\mathcal{N}}, \mathbf{d}, \dot{\mathrm{X}}_\mathrm{c}, \mathcal{X}_\mathrm{obst}, \mathrm{C_{cw}}, \mathrm{C_{ccw}}, \rho, v_\mathrm{max})$
  for all $\mathbf{x}_\mathrm{ci} \in \bar{\mathcal{N}}$ **do**
    $(X, \dot{X}) := \textsc{Find-Connecting-Segment}(\mathbf{x}_\mathrm{ci}, \mathbf{c}_{\mathrm{cw,j}}, \mathbf{c}_{\mathrm{ccw,j}})$;
    $\Delta d_\mathrm{ij} := \textsc{Path-Length}(X)$;
    if $\Delta d_\mathrm{ij} + d_\mathrm{j} < d_\mathrm{i}$ **and** $\forall\, \mathbf{x} \in X : \mathbf{x} \notin \mathcal{X}_\mathrm{obst}$ **then**
      $d_\mathrm{i} := \Delta d_\mathrm{ij} + d_\mathrm{j}$; \\ shorten path by going through j
      $\dot{\mathbf{x}}_\mathrm{ci} := \dot{\mathbf{x}}_0$; \\ direct velocity to head down next segment in path
      $\mathbf{c}_{\mathrm{cw,i}} := \mathbf{x}_\mathrm{ci} + \rho[\dot{x}_{12}, -\dot{x}_{11}]^T/v_\mathrm{max}$; $\mathbf{c}_{\mathrm{ccw,i}} := \mathbf{x}_\mathrm{ci} + \rho[-\dot{x}_{12}, \dot{x}_{11}]^T/v_\mathrm{max}$;
    end if
  end for

Algorithm 1: Constructing the Feasible Path Tree: This is a modified form of Dijkstra's Algorithm, in which costs are found at the same time as paths are grown to the goal.

a turning circle. Also, by this construction the end of any connecting segment is continuous in position and velocity with the start of a tree segment. At the initial position, two circles that are tangent to the initial velocity are added so that the existence of feasible path from start to the goal is guaranteed. These properties will be exploited in the next section to prove the stability of the controller. Note that if the initial position cannot be added to the feasible path tree, the optimization problem is not solved and the vehicle chooses not to start.

Proof of Stability and Guaranteed Arrival

Suppose the current state is $\mathbf{x} = \mathbf{x}(k)$, then the stable receding horizon controller solves the optimization problem

$$\min_{\mathbf{u}=\{\mathbf{u}(k),\ldots,\mathbf{u}(k+N_p-1)\}} F(\mathbf{x}(k+N_p))$$

s.t.

$$\left.\begin{array}{rcl} \mathbf{x}(k+i+1) & = & f(\mathbf{x}(k+i), \mathbf{u}(k+i)) \\ \mathbf{u}(k+i) & \in & \mathcal{U} \\ \mathbf{x}(k+i) & \in & \mathcal{X} \end{array}\right\} \quad (i = 0, 1, \ldots, N_p - 1)$$

$$\mathbf{x}(k+N_p) \in \mathcal{X}_f \subset \mathcal{X}$$

where $N_p \geq 1$ is a number of steps in *the planning horizon*, $F(k+N_p) \geq 0$ is a terminal penalty and in this work is taken to be the cost to minimize, $\mathbf{x}(k+i+1) = f(\mathbf{x}(k+i), \mathbf{u}(k+i))$ corresponds to the system dynamics, $\mathcal{U}$ represents the set of admissible control inputs, $\mathcal{X}$ represents the set of admissible states, and $\mathcal{X}_f$ is a terminal constraint set. Note that the terminal penalty $F(\mathbf{x}(k+N_p))$ is a function of $\mathbf{x}(k)$ and $\mathbf{u} = \{\mathbf{u}(k), \ldots, \mathbf{u}(k+N_p-1)\}$.

The optimal solution of this program starting at state $\mathbf{x}(k)$ results in a state trajectory $X^\mathrm{o}(\mathbf{x}(k))$ and control trajectory $U^\mathrm{o}(\mathbf{x}(k))$ with associated minimum cost $F^O(\mathbf{x}(k+N_p))$. Let $\mathbf{x}^\mathrm{o}(i; \mathbf{x}(k))$ be the i^th $(i = 1, \ldots, N_p)$ state reached by applying $U^\mathrm{o}(\mathbf{x}(k))$ starting from $\mathbf{x}(k)$, and let $\mathbf{u}^\mathrm{o}(1\!:\!i; \mathbf{x}(k))$ be the first i control inputs applied. Assume that the first N_e control inputs $\mathbf{u}^\mathrm{o}(1\!:\!N_e; \mathbf{x}(k))$ from $U^\mathrm{o}(\mathbf{x}(k))$ are executed before the optimization is performed again, where N_e specifies a number of steps in *the execution horizon*. Then, the change of the terminal cost is

$$F^O(\mathbf{x}(k+N_e+N_p)) - F^O(\mathbf{x}(k+N_p))$$

$$= \min_{\substack{\mathbf{u}=\{u(k+N_e),\ldots,u(k+N_e+N_p-1)\} \\ \text{given } \mathbf{x}(k), u^\mathrm{o}(1:N_e;\mathbf{x}(k))}} F(\mathbf{x}(k+N_e+N_p))$$

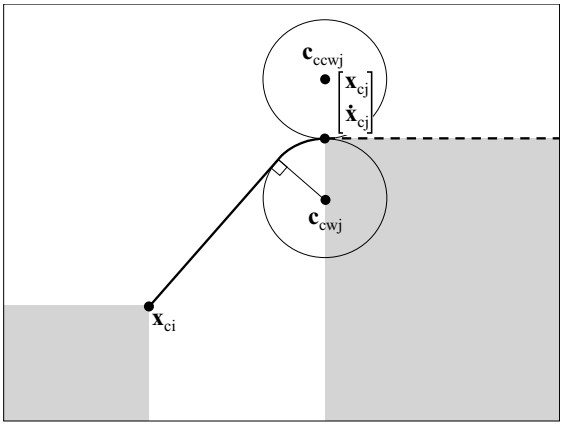

Fig. 3: The Trajectory Returned by FIND-CONNECTING-SEGMENT. This routine connects the $\mathbf{x}_{\text{ci}}$ to the state $[\mathbf{x}_{\text{cj}}^T\ \dot{\mathbf{x}}_{\text{cj}}^T]^T$. First, it finds the shortest line through $\mathbf{x}_{\text{ci}}$ that is tangent to the turning circle at $\mathbf{c}_{\text{cw,i}}$ or $\mathbf{c}_{\text{ccw,i}}$ and compatible with the turning direction of the circle. The turning path from the tangent to $[\mathbf{x}_{\text{j}}^T\ \dot{\mathbf{x}}_{\text{j}}^T]^T$ is then added. ⋆

$$- \min_{\substack{\mathbf{u}=\{u(k),\ldots,u(k+N_p-1)\} \\ \text{given } \mathbf{x}(k)}} F(\mathbf{x}(k+N_p)) \tag{5}$$

To proceed we must first consider the exact form of the terminal penalty $F(\mathbf{x})$, which is chosen in this paper to be the distance from $\mathbf{x}(k+N_p)$ to $\mathbf{x}_{\text{goal}}$ *along a kinodynamically feasible path*

$$F(\mathbf{x}) = V + \rho|\theta| + d_{\text{i}} \tag{6}$$

where V is defined in Eq. 3 as the length of the tangent from $\mathbf{x}(k+N_p)$ to the point $\mathbf{w}$ at which it intersects the circle, $|\theta|$ is the magnitude of the angle that the vehicle turns between $\mathbf{w}$ and the selected state of known cost $\mathbf{x}_{\text{ci}}$, and d_{i} is the length of the kinodynamically feasible path found from $\mathbf{x}_{\text{ci}}$ to the goal. The tree of trajectory segments is rooted at the goal and is connected to the start. This trajectory is made up of straight segments connecting arcs around turning circles, and the turning circles with radius $||\mathbf{r}|| = \rho$, the vehicle's minimum turning radius, so it is a dynamically feasible trajectory. Thus, there exists an admissible control input sequence $U^*(\mathbf{x}(k+N_p))$ that could force the vehicle to follow this trajectory past the planning horizon. Note that the terminal speed of the vehicle is $v_{\max}$, and along this feasible trajectory $F(\mathbf{x})$ decreases by $v_{\max}dt$ at each step. Thus, at $\mathbf{x} = \mathbf{x}(k+N_e)$, if we apply $u^{\text{o}}(N_e+1:N_p;\mathbf{x}(k))$ and then the first N_e steps of $U^*(\mathbf{x}(k+N_p))$, instead of the optimal (but

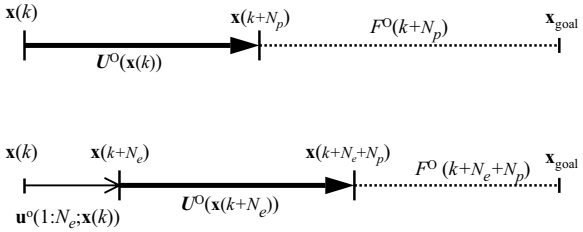

Fig. 4: Top figure shows first optimization problem solved starting at

$$\mathbf{x}(k) : F^O(\mathbf{x}(k+N_p)) = \min_{\substack{\mathbf{u} \\ \text{given}\mathbf{x}(k)}} F(\mathbf{x}(k+N_p)).$$

After executing the first N_e steps of the control input obtained as a solution of the first problem, the next optimization problem is solved as shown in the bottom figure. ⋆

unknown) $U^{\text{o}}(\mathbf{x}(k+N_e))$, Eq. 5 reduces to

$$\begin{aligned}
& F(\mathbf{x}(k+N_e+N_p))\Big|_{\mathbf{x}(k),u^{\text{o}}(1:N_e;\mathbf{x}(k)),U^{\text{o}}(\mathbf{x}(k+N_e))} \\
& \qquad - F(\mathbf{x}(k+N_p))\Big|_{\mathbf{x}(k),U^{\text{o}}(\mathbf{x}(k))} \\
\leq\ & F(\mathbf{x}(k+N_e+N_p))\Big|_{\mathbf{x}(k),U^{\text{o}}(\mathbf{x}(k)),u^*(1:N_e;\mathbf{x}(k+N_p))} \\
& \qquad - F(\mathbf{x}(k+N_p))\Big|_{\mathbf{x}(k),U^{\text{o}}(\mathbf{x}(k))} \\
=\ & -N_e\ v_{\max}dt
\end{aligned} \tag{7}$$

The last equality will hold as long as $\mathbf{x}(k+N_e+N_p)$ is a distance at least $(N_e\ v_{\max}dt)$ from the goal. In that case, each time the optimization is performed, the optimal cost to go F^O will decrease by at least $(N_e\ v_{\max}dt)$. Thus after solving the optimization problem at most

$$K_{\max} = \text{floor}\left(\frac{F^O(\mathbf{x}(0+N_p))}{N_e\ v_{\max}dt}\right)$$

times, we have that

$$F^O(\mathbf{x}(k+N_p)) < N_e\ v_{\max}dt \equiv F_f$$

which implies that the distance from the terminal point $\mathbf{x}(k+N_p)$ to the goal *along a feasible path* is less than $N_e\ v_{\max}dt$, and furthermore, from this terminal point the mission can be completed in less than N_e steps (see Fig. 5). If $\mathbf{x}(k+N_e+N_p)$ is not $N_e\ v_{\max}dt$ away from the goal, Eq. 7 does not hold true, but since the distance from the terminal point $\mathbf{x}(k+N_e+N_p)$ to the goal along a feasible path is

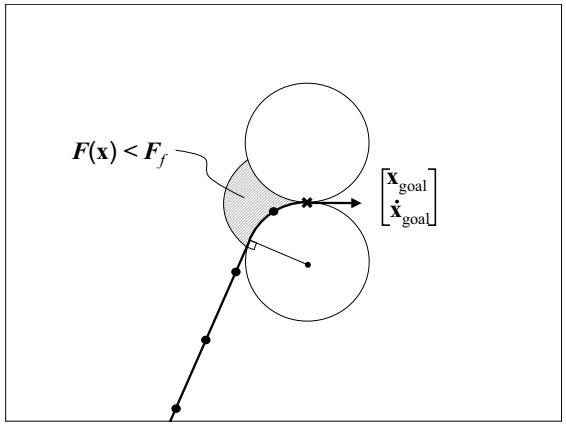

Fig. 5: $\times$ mark specifies goal and $\bullet$ specifies terminal points. Shaded region specifies the area the distance from which to the goal along a feasible path is less than $N_e\ v_{\max}dt$. Once $\mathbf{x}$ reaches the shaded area, it can reach the goal within less that N_e time step. $\star$

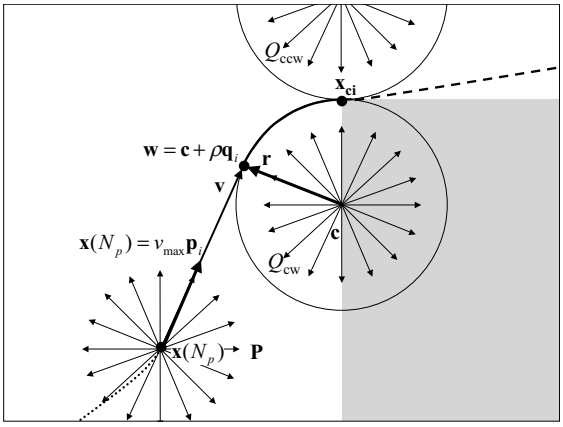

Fig. 6: The vectors involved in the tangency constraints. The controller simultaneously chooses the direction for $\dot{\mathbf{x}}(N_p)$ and a perpendicular direction for $\mathbf{w}-\mathbf{c}$ from two finite sets of mutually perpendicular unit vectors. The vector $\mathbf{v}$ is also constrained to be in the direction of $\dot{\mathbf{x}}(N_p)$. $\star$

less than $N_e\ v_{\max}dt$, the mission can be completed in less than N_e steps from this terminal point. Finally, once the terminal point is within $N_e\ v_{\max}dt$ of the goal, one could set $N_p = N_e = 1$ and apply the same arguments given above to guarantee that the terminal point gets within a distance of $v_{\max}dt$ (along a feasible path) from the goal.

Hence, the guaranteed arrival within a bounded time has been proved. Note that admissible but potentially suboptimal control input sequence U^* is used only to obtain the lower bound of the decrease of the terminal cost, and what is actually applied is the optimal control sequence obtained as a solution of the optimization problem.

MILP FORMULATION

While the formulation of the receding horizon controller presented in the previous section is provably stable, it is nonlinear. The optimization problem includes the requirement that the velocity at the end of the planning horizon $\dot{\mathbf{x}}(N_p)$ be tangent to one of the turning circles. As given in Eq. 4, this constraint can be written using a dot product of the decision variables $\dot{\mathbf{x}}$ and $\mathbf{r}$, showing that this constraint is quadratic. Furthermore, the angle through which the vehicle turns is used in the terminal penalty given in Eq. 6, but is not available directly.

Because nonlinear optimization is typically far more difficult than linear optimization, it is desirable to reformulate this controller as a linear optimization problem. This section presents a version of the receding horizon controller that uses a discretization in order to formulate the controller as a MILP. This discretized version is also stable, and its computational performance is discussed in the Results section.

Tangency Constraint

In section , Eq. 4 used a quadratic term to constrain the terminal velocity to be tangent to a turning circle. The MILP formulation of the trajectory designer converts this constraint into a discrete, linear form. The vectors involved in the new constraints are shown in Fig. 6. The controller's choice of one circle center $\mathbf{c}$ from the set C is encoded by the vectors of binary decision variables $\mathbf{b}_{\rm cw}$ and $\mathbf{b}_{\rm ccw}$.

$$\begin{aligned}\mathbf{c} &= \sum_{\mathrm{i}\in\mathcal{C}}(b_{\mathrm{cw,i}}\mathbf{c}_{\mathrm{cw,i}} + b_{\mathrm{ccw,i}}\mathbf{c}_{\mathrm{ccw,i}})\\ 1 &= \sum_{\mathrm{i}\in\mathcal{C}}(b_{\mathrm{cw,i}} + b_{\mathrm{ccw,i}})\\ \mathbf{b}_{\rm cw} \in \{0,1\}^{\rm N_C},&\quad \mathbf{b}_{\rm ccw} \in \{0,1\}^{\rm N_C}\end{aligned}$$

The choice of a direction for $\dot{\mathbf{x}}(N_p)$ from a set P of N_P possible unit vectors $\{\mathbf{p}_0,\ldots,\mathbf{p}_{N_P}\}$ is encoded with the vectors of binary decision variables $\mathbf{m}_{\rm cw} \cup \mathbf{m}_{\rm ccw}$.

$$\begin{aligned}\dot{\mathbf{x}}(N_p) &= v_{\max}\sum_{\mathrm{i}\ \in\ \mathcal{P}}(m_{\mathrm{cw,i}} + m_{\mathrm{ccw,i}})\mathbf{p}_i\\ 1 &= \sum_{\mathrm{i}\ \in\ \mathcal{P}}(m_{\mathrm{cw,i}} + m_{\mathrm{ccw,i}})\\ \mathbf{m}_{\rm cw} \in \{0,1\}^{\rm N_P},&\quad \mathbf{m}_{\rm ccw} \in \{0,1\}^{\rm N_P}\end{aligned}$$

where $\mathcal{P} = \{1 \ldots N_P\}$. To enforce orthogonality, the binary variables $\mathbf{m}_{\rm cw} \cup \mathbf{m}_{\rm ccw}$ simultaneously determine the direction of $\mathbf{r} = \mathbf{w} - \mathbf{c}$ from the set $Q = Q_{\rm cw} \cup Q_{\rm ccw}$ of perpendicular unit vectors. The set $Q_{\rm cw}$ ($Q_{\rm ccw}$) is constructed so that $\mathbf{q}_{\rm ccw,i}$ ($\mathbf{q}_{\rm ccw,i}$) is the correct direction for $\mathbf{r}$ if the vehicle reaches the clockwise (counter-clockwise) turning circle on a tangent in the direction $\mathbf{p}_i$.

$$\begin{aligned} \mathbf{q}_{\rm cw,i} &= \begin{bmatrix} -p_{i,2} \\ p_{i,1} \end{bmatrix} \\ \mathbf{q}_{\rm ccw,i} &= \begin{bmatrix} p_{i,2} \\ -p_{i,1} \end{bmatrix} \end{aligned}$$

The direction for $\mathbf{r}$ is selected as

$$\mathbf{r} = \rho \sum_{i \in \mathcal{P}} (m_{\rm cw,i}\mathbf{q}_{\rm cw,i} + m_{\rm ccw,i}\mathbf{q}_{\rm ccw,i})$$

If a turning circle is chosen from $C_{\rm cw}$ by setting one element of $\mathbf{b}_{\rm cw}$ 1, then the optimization must choose a direction for $\mathbf{r}$ from $Q_{\rm cw}$ by setting one of the $m_{\rm cw,j} = 1$ and similarly for the counter-clockwise direction. This is enforced with the following constraints:

$$\begin{aligned} \sum_{\rm j \in \mathcal{P}} m_{\rm cw,j} &= \sum_{\rm i \in \mathcal{C}} b_{\rm cw,i} \\ \sum_{\rm j \in \mathcal{P}} m_{\rm ccw,j} &= \sum_{\rm i \in \mathcal{C}} b_{\rm ccw,i} \end{aligned}$$

To enforce tangency, $\mathbf{v}$ must be parallel to $\dot{\mathbf{x}}(N_p)$ and thus perpendicular to the selected direction for $\mathbf{r}$.

$$\begin{aligned} \mathbf{v} &= \mathbf{c} + \mathbf{r} - \mathbf{x}(N_p) \\ \forall\, i \in \mathcal{P}&: \\ \mathbf{v} \cdot \mathbf{q}_{\rm cw,i} &\leq \epsilon + M(1 - p_{\rm cw,i}) \\ \mathbf{v} \cdot \mathbf{q}_{\rm cw,i} &\geq -\epsilon - M(1 - p_{\rm cw,i}) \\ \mathbf{v} \cdot \mathbf{q}_{\rm ccw,i} &\leq \epsilon + M(1 - p_{\rm ccw,i}) \\ \mathbf{v} \cdot \mathbf{q}_{\rm ccw,i} &\geq -\epsilon - M(1 - p_{\rm ccw,i}) \end{aligned}$$

where ϵ is a small positive number and M is a large positive number. The tangency constraints are enforced for $\mathbf{q}_{\rm cw,i}$ or $\mathbf{q}_{\rm ccw,i}$ if $m_{\rm cw,i} = 1$ or $m_{\rm ccw,i} = 1$, and are relaxed otherwise.

These constraints guarantee that the velocity vector $\dot{\mathbf{x}}(N_p)$ is tangent to one of the turning circles. Obstacle avoidance constraints

$$\begin{aligned} x_{k+i,1} &\leq u_{\rm low} + M\, b_{\rm obst,1} \\ x_{k+i,1} &\geq u_{\rm high} - M\, b_{\rm obst,2} \\ x_{k+i,2} &\leq v_{\rm low} + M\, b_{\rm obst,3} \\ x_{k+i,2} &\geq v_{\rm high} - M\, b_{\rm obst,4} \\ \sum_{\rm j=1}^{4} b_{\rm obst,j} &\leq 3, \end{aligned}$$

the discretized dynamics model

$$\begin{bmatrix} x_{\rm i+1,1} \\ x_{\rm i+1,2} \\ \dot{x}_{\rm i+1,1} \\ \dot{x}_{\rm i+1,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\rm i,1} \\ x_{\rm i,2} \\ \dot{x}_{\rm i,1} \\ \dot{x}_{\rm i,2} \end{bmatrix} + \begin{bmatrix} \frac{dt^2}{2m} & 0 \\ 0 & \frac{dt^2}{2m} \\ \frac{dt}{m} & 0 \\ 0 & \frac{dt}{m} \end{bmatrix} \begin{bmatrix} u_{\rm i,1} \\ u_{\rm i,2} \end{bmatrix},$$

and the velocity and control force limit

$$\begin{aligned} L_2(\dot{\mathbf{x}}(i)) &\leq v_{\max} \\ L_2(\mathbf{u}(i)) &\leq u_{\max} \end{aligned}$$

are also enforced in this formulation.

Obstacle avoidance over the tangent portion of the connecting segment is enforced by constraining a discrete set of interpolating points along $\mathbf{v}$ to lie outside of all obstacles as

$$\begin{aligned} &\forall\, \tau \in [0,1] : \\ &\quad \mathbf{x}(k+N_p) + \tau \cdot (\mathbf{x}_{\rm vis} - \mathbf{x}(k+N_p)) \notin \mathcal{X}_{\rm obst} \end{aligned}$$

Over the turn from $\mathbf{w}$ to the selected state of known cost $\mathbf{x}_{\rm c}$, constraints on the selected direction for $\mathbf{r}$ must be added to enforce obstacle avoidance. The turning path from $\mathbf{w} = \mathbf{c}_{\rm cw,i} + \rho\mathbf{q}_{\rm cw,j}$ to the state of known cost $\mathbf{x}_{\rm c,i}$ is checked before the trajectory design begins. If the path penetrates any obstacles, then the simultaneous selection of $\mathbf{x}_{\rm c,i}$ and $\mathbf{q}_{\rm cw,j}$ is excluded by setting a binary parameter $a_{\rm cw,i,j} = 0$. The following constraints enforce this exclusion:

$$\begin{aligned} &\forall\, j \in \mathcal{P} : \\ &\quad m_{\rm cw,j} \leq \sum_{i \in \mathcal{C}} a_{\rm cw,i,j} b_{\rm cw,i} \\ &\quad m_{\rm ccw,j} \leq \sum_{i \in \mathcal{C}} a_{\rm ccw,i,j} b_{\rm ccw,i} \end{aligned}$$

Selection of Unit Vectors in P

This discussion has not yet addressed the selection of particular directions for the unit vectors $\mathbf{p}_{\rm i} \in P$. There is some freedom in this choice. It is desirable to minimize the number N_P of unit vectors in P, since each of the directions requires two binary

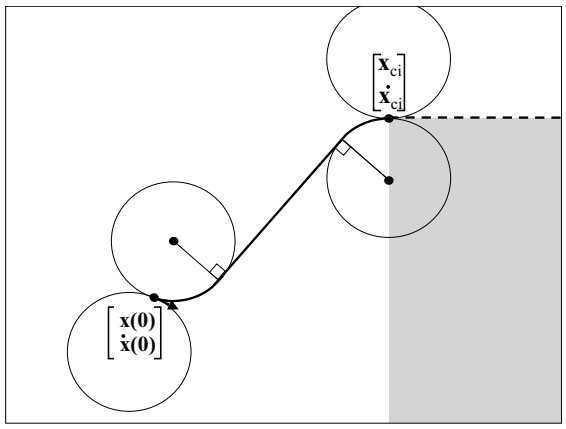

Fig. 7: The Trajectory between two states $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ and $[\mathbf{x}_{ci}^T\ \dot{\mathbf{x}}_{ci}^T]^T$ returned by FIND-CONNECTING-SEGMENT. First, it finds the shortest line that is tangent to turning circles for both states, and compatible with the turning direction of the circles. Turning paths from $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ to the tangent, and from the tangent to $[\mathbf{x}_{ci}^T\ \dot{\mathbf{x}}_{ci}^T]^T$ are then added. ⋆

variables to be added to the program, and the complexity of a MILP grows rapidly (non-polynomial) with the number of binary variables. However, the terminal state $\dot{\mathbf{x}}(N_p)$ must be positioned on one of the tangent lines corresponding to the available unit vectors. If too few unit vectors are used, it might not be possible for the vehicle to reach any tangent line at the end of its planning horizon, resulting an infeasible program.

It is straightforward to avoid this by adding a unit vector to P. After the tree of trajectory segments is found and before each trajectory optimization is performed, a connecting segment from the vehicle's current state $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ to a state of known cost $[\mathbf{x}_{ci}^T\ \dot{\mathbf{x}}_{ci}^T]^T$ is found using a modified version of the routine FIND-CONNECTING-SEGMENT. An example of such a path is shown in Fig. 7. The state of known cost is chosen to minimize the associated total path length. The state $[\mathbf{x}(N_p)^T\ \dot{\mathbf{x}}(N_p)^T]^T$ is found that would be reached at the end of the planning horizon if the connecting segment and the following tree segment were followed. By adding the vehicle's heading $\dot{\mathbf{x}}(N_p)/\|\dot{\mathbf{x}}(N_p)\|$ to P, a feasible heading is guaranteed to be available to the trajectory optimization. If FIND-CONNECTING-SEGMENT cannot find a path to a state of known cost before the first optimization is performed, then the trajectory design problem is rejected. Otherwise, FIND-CONNECTING-SEGMENT will always find a path to a state of known cost. This addition guarantees that the optimization program remains feasible, and adding more unit vectors may make it possible to find solutions of lower cost.

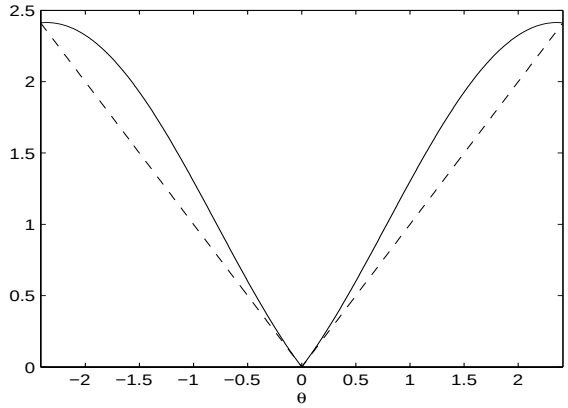

Fig. 8: Comparison of Upper Bounding Function $\|\hat{\dot{\mathbf{x}}}(N_p) - \hat{\dot{\mathbf{x}}}_c\|_1$ to $|\theta|$. $\|\hat{\dot{\mathbf{x}}}(N_p) - \hat{\dot{\mathbf{x}}}_c\|_1$ is shown with solid line, $|\theta|$ is shown with dashed line. ⋆

Arc Length Approximation and Linearized Terminal Penalty

In addition to enforcing the tangency requirement, the stable controller requires a nonlinear relationship to exactly evaluate the arc length $\rho|\theta|$ covered by the vehicle as it traverses the turning circle. This arc length contributes to the total length of the trajectory followed to the goal, so it must be included in the cost function.

The discretized stable controller replaces the exact evaluation of $|\theta|$ in Eq. 6 with an upper bound $|\theta|'$ on $|\theta|$ that is valid whenever $|\theta| \leq \theta_{\max}$. By using an upper bound, the inequality in Eq. 7 always holds. Recall that $|\theta|$ is the magnitude of the angle between the vehicle's heading at the start of the turn, $\hat{\dot{\mathbf{x}}}(N_p) = \dot{\mathbf{x}}(N_p)/v_{\max}$, and the vehicle's heading at the end of the turn, $\hat{\dot{\mathbf{x}}}_c = \dot{\mathbf{x}}_c/v_{\max}$. Fig. 8 shows that the one-norm $\|\hat{\dot{\mathbf{x}}}(N_p) - \hat{\dot{\mathbf{x}}}_c\|_1$ provides an upper bound on $|\theta|$ whenever $|\theta| \leq \theta_{\max}$. This $\theta_{\max}$ is calculated as follows. Without loss of generality, take the case where $\hat{\dot{\mathbf{x}}}(N_p) = [1\ 0]^T$ and $\hat{\dot{\mathbf{x}}}_c = [\cos\theta\ \sin\theta]^T$. Then,

$$\begin{aligned} f(\theta) &\equiv \|\hat{\dot{\mathbf{x}}}(N_p) - \hat{\dot{\mathbf{x}}}_c\|_1 - |\theta| \\ &= |\cos\theta - 1| + |\sin\theta| - |\theta| \end{aligned}$$

Over the range $0 \leq \theta \leq \pi$,

$$f(\theta) = -\cos\theta + 1 + \sin\theta - \theta$$

and $f(\theta) = 0$ gives $\theta = 0, \approx 2.412$. Thus, $\theta_{\max} =$

2.412. Since $f(\theta) = f(-\theta)$,

$$f(\theta) \geq 0 \qquad [-\theta_{\max},\ \theta_{\max}]$$

Therefore, $\|\hat{\mathbf{x}}(N_p) - \hat{\mathbf{x}}_c\|_1$ must be greater than $|\theta|$ and is an upper bound on $|\theta|$ over $|\theta| \leq \theta_{\max}$. The following constraints are added to the discretized controller to evaluate $|\theta|' \geq \|\hat{\mathbf{x}}(N_p) - \hat{\mathbf{x}}_c\|_1 \geq |\theta|$

$$\begin{aligned} |\theta|' &\geq (\hat{x}(N_p)_1 - \hat{x}_{c1}) + (\hat{x}(N_p)_2 - \hat{x}_{c2}) \\ \text{and } |\theta|' &\geq (\hat{x}(N_p)_1 - \hat{x}_{c1}) - (\hat{x}(N_p)_2 - \hat{x}_{c2}) \\ \text{and } |\theta|' &\geq -(\hat{x}(N_p)_1 - \hat{x}_{c1}) + (\hat{x}(N_p)_2 - \hat{x}_{c2}) \\ \text{and } |\theta|' &\geq -(\hat{x}(N_p)_1 - \hat{x}_{c1}) - (\hat{x}(N_p)_2 - \hat{x}_{c2}) \end{aligned}$$

In order to restrict the controller to turn at most $\theta_{\max}$ radians, binary variables associated with heading directions $m_{\text{cw,i}}$ ($m_{\text{ccw,i}}$) that would result in a larger turn to the state of known cost $\mathbf{x}_{\text{c,i}}$ are also set to 0 by setting $a_{\text{cw,i,j}} = 0$ ($a_{\text{ccw,i,j}} = 0$).

Finally, the linearized terminal penalty is

$$F(\mathbf{x}) = L(\mathbf{v}) + \rho|\theta|' + \sum_{i \in \mathcal{C}} (b_{\text{cw,i}} + b_{\text{ccw,i}}) d_i$$

where $L(\mathbf{v})$ approximates the two-norm of the vector $\mathbf{v}$ by taking the maximum value of its projection onto a series of N_u vectors with the magnitude $1/\cos\left(\frac{\pi}{N_u}\right)$, so that $L(\mathbf{v})$ is always larger than $\|\mathbf{v}\|_2$ and is an upper bound on $\|\mathbf{v}\|_2$.

SIMULATION RESULTS

Simulation results are presented below for the stable receding horizon trajectory designer. Commercially available software CPLEX/AMPL is used as a MILP solver. The results show that the stable receding horizon trajectory designer is capable of planning near-optimal trajectories in highly constrained environments, and does so with computational effort that is comparable to that of the SLC.

Operation of the Trajectory Designer

Successive trajectory segments designed by the stable trajectory designer are shown in Fig. 9. Although the controller selects a turning circle as part of every solution, it does not necessarily execute a trajectory that follows the circumference of a circle because the connecting segment that follows the suboptimal path is beyond the executing horizon, and the controller can replan it with new turning circles available before it is executed. This emphasizes the fact that, despite the sub-optimality of the path that is implicitly constructed from tangent lines and turning circles, the optimality of the executed trajectory is not limited to this construction. The tangent lines and turning circles simply provide a straightforward geometric construction of a feasible path to the goal. The stable receding horizon trajectory planner took 58.9 seconds to design this trajectory, and arrived at the goal in 36 steps.

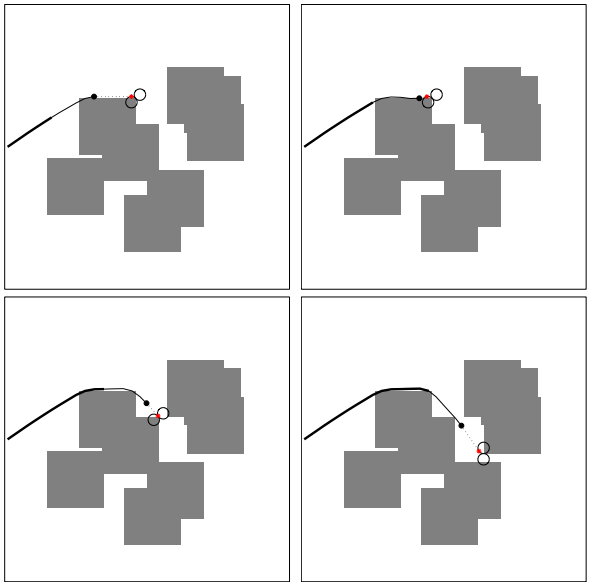

Fig. 9: Plans of the Stable Trajectory Designer. Successive trajectory segments designed by the stable trajectory designer are shown. Starting point is at left, goal is at right. The thick line shows executed steps, the thin line shows planned steps, and the dashed line shows $\mathbf{v}$. The terminal point $\mathbf{x}(k + N_p)$ is shown with $\bullet$, and $\mathbf{w}$ is shown with $*$. $\star$

Trajectory Design in Highly Constrained Environments

As described in previous sections, there is always an implicitly constructed kinodynamically feasible path from the terminal state $[\mathbf{x}(k + N_p)^T\ \dot{\mathbf{x}}(k + N_p)^T]^T$ chosen by the stable receding horizon designer to the goal. This prevents the designer from forming a plan that relies on passing through a series of gaps between obstacles past the planning horizon that is kinodynamically infeasible.

An obstacle field with such a series of gaps is shown in Fig. 10. The SLC is not prevented from selecting the dynamically infeasible path past the planning horizon, and it does so because the narrow gap provides the shortest straight-line path to the goal. The same straight line path would be found in the construction of the kinodynamically feasible path to the goal for a vehicle with turning radius $\rho = 0$. How-

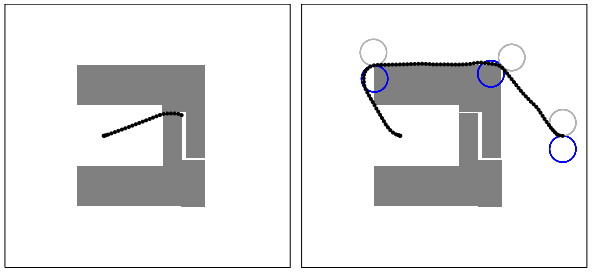

Fig. 10: Trajectory examples in highly constrained environment. Start is at the center, goal is at right. Left: trajectory planned with the SLC. Note that optimization problem becomes infeasible before the goal is reached. Right: trajectory planned with stable receding horizon controller. Note that narrow path through obstacles is ruled out by trajectory tree construction and a longer, but kinodynamically feasible, path is actually followed. ⋆

ever, if the vehicle's minimum turning radius is sufficiently large, then it cannot pass through this series of gaps and that trajectory becomes infeasible. If a vehicle actually followed this path, it would actually collide with the obstacles (left plot in Fig. 10).

Design of Long Trajectories

The stable receding horizon controller is guaranteed to reach the goal if its first optimization problem is feasible, but this controller's MILP formulation requires more binary variables than that of the SLC. The computational impact of this increase was examined by comparing the solution times of both controllers with equal planning horizons for a very long trajectory.

In order to shorten the solution time, candidate trajectory solutions were provided for each optimization problem that was solved. Starting the branch-and-bound search procedure with a lower upper bound on the optimal cost allows more of the search space to be pruned because its linear relaxation has a cost higher than the upper bound, shortening the search.

In the case of the SLC, the candidate trajectory was constructed by re-using the unexecuted trajectory points from the previous solution, and replacing the executed steps by extrapolating along the line of visibility to the next state of known cost. The candidate trajectory for the stable receding horizon controller was constructed by reusing unexecuted steps, and choosing trajectory points along the connecting and tree segments to the goal. Both of these procedures can be executed very rapidly. These candidate solutions were feasible more often for the stable controller than for the SLC, since there can be a sharp turn between the last state $[\mathbf{x}(k+N_p)^T\ \dot{\mathbf{x}}(k+N_p)^T]^T$ of the SLC's trajectory and the line of visibility, while the stable controller's terminal heading was guaranteed to be the same as that of the connecting segment.

Table 1: Cumulative Solution Times for Long Trajectories

Trajectory Design Approach	Without Initial Guess		With Initial Guess	
	Sol'n. Time (s)	Num. Steps	Sol'n. Time (s)	Num. Steps
SLC	136.2	160	91.45	160
Stable	168.2	161	116.8	161

The computation times for both formulations are given in Table 1, and the stable controller's tree of trajectories and the solution is shown in Fig. 11 and 12. The trajectories found with and without providing candidate solutions were almost identical. These results show that the stable trajectory designer takes 23.7% longer to compute than the SLC, and that the length of the trajectory planned by the stable controller was only 1 step longer than that designed by the SLC. It is interesting to note that the tree of trajectory segments selected the sequence AB**C**, whereas the RH-MILP, which has more freedom in designing a path around the corner A, selected a sequence A**DC**. This occurred because the estimated length from the terminal point $\mathbf{x}(k+N_p)$ along the tree D**C** was found to be shorter than that along the tree B**C**. Since this selection reduces the cost-to-go by more than the required $N_e\ v_{\max}dt$, the stability guarantee still holds.

CONCLUSIONS

This paper presents a stable receding horizon trajectory designer for a vehicle maneuvering in a complex environment with obstacles. Constraints are included that specify that each time a trajectory segment is optimized, a kinodynamically feasible trajectory exists past the extent of the planning horizon to the goal. This trajectory provides the next optimization with a feasible solution, and it is shown that the distance along this feasible path to the goal is reduced by a pre-specified amount at each step, thereby ensuring the stability of the controller. Computational results show that the stable trajectory designer gives trajectories through highly constrained obstacle fields, while requiring computation time that is approximately 25% longer than that of the controller using the straight line approximation.

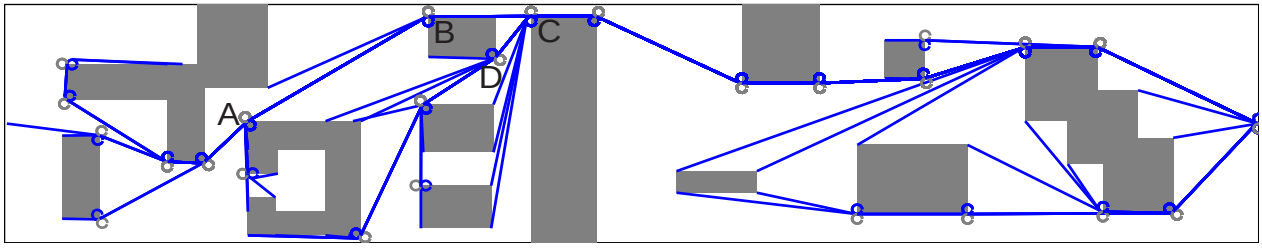

Fig. 11: Tree of Trajectory Segments for Long Trajectory. This tree of trajectory segments was produced by the stable receding horizon controller. ⋆

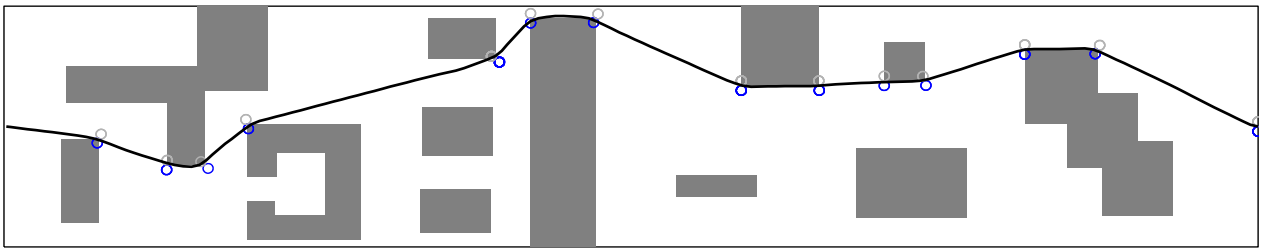

Fig. 12: Sample Long Trajectory Designed by Stable Controller. ⋆

ACKNOWLEDGMENTS

Research funded by DARPA contract (MICA) N6601-01-C-8075.

REFERENCES

[1] A. Jadbabaie, J. Primbs, and J. Hauser. Unconstrained receding horizon control with no terminal cost. In *Proceedings of the American Control Conference*, Arlington, VA, June 2001.

[2] J. S. Bellingham, A. G. Richards, J. P. How. Receding Horizon Control of Autonomous Aerial Vehicles In *Proceedings of the American Control Conference*, Anchorage AK, May 2002.

[3] J. S. Bellingham. Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming Master's thesis, Massachusetts Institute of Technology, 2002.

[4] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[5] D.Q.Mayne, J.B.Rawlings, C.V.Rao, and P.O.M.Scokaert. Constrained model predictiv