On-line Trajectory Planning for Aerial Vehicles: a Safe Approach with Guaranteed Task Completion

David A. Anisi[*]

Royal Institute of Technology (KTH), SE-100 44, Stockholm, Sweden

John W.C. Robinson[†] and Petter Ögren[†]

Swedish Defence Research Agency (FOI), SE-164 90, Stockholm, Sweden

On-line trajectory optimization in three dimensional space is the main topic of the paper at hand. The high-level framework augments on-line receding horizon control with an off-line computed terminal cost that captures the global characteristics of the environment, as well as any possible mission objectives. The first part of the paper is devoted to the single vehicle case while the second part considers the problem of simultaneous arrival of multiple aerial vehicles.

The main contribution of the first part is two-fold. Firstly, by augmenting a so called safety maneuver at the end of the planned trajectory, this paper extends previous results by addressing provable safety properties in a 3D setting. Secondly, assuming initial feasibility, the planning method presented is shown to have finite time task completion. Moreover, a quantitative comparison between the two competing objectives of optimality and computational tractability is made. Finally, some other key characteristics of the trajectory planner, such as ability to minimize threat exposure and robustness, are highlighted through simulations.

As for the simultaneous arrival problem considered in the second part, by using a time-scale separation principle, we are able to adopt standard Laplacian control to a consensus problem which is neither unconstrained, nor first order.

I. Introduction

In this paper, on-line trajectory planning for an aerial vehicle subject to simultaneous kinematic and dynamic constraints is considered. The trajectory planning problem is formulated as a somewhat modified Optimal Control Problem (OCP), where optimality can be dictated by a mixture of conditions and penalties relating to time- and energy efficiency, threat avoidance, stealth behavior and various end-point constraints imposed by the target-seeker and/or terrain. An underlying assumption however, is that due to imperfect information, the kinematic constraints, as well as the location of the target set and possible threats, might change during the course of flight. Consequently, we can not use the family of techniques that rely on off-line generation of a trajectory database for on-line interrogation [1–3]. Also, assuming the problem originates from a complex real-world application, the existence of analytical solutions is unlikely; thus we seek fast computational algorithms for solving the OCP.

[*]Research Assistant, Division of Optimization and Systems Theory, KTH, Student Member AIAA, anisi@math.kth.se.

[†]Senior Researcher, Department of Autonomous Systems, FOI, Member AIAA.

Background and Solution Foundation

In late 80's, by extending their "free path encoding method" [4], Canny and Reid demonstrated the $\mathcal{NP}$ - hardness of finding a shortest kinodynamic path for a point moving amidst polyhedral obstacles in a three dimensional environment [5]. Therefore, in order to meet the on-line computational requirement, attention must be paid to *approximation methods*, *i.e.* computationally efficient algorithms that compute kinodynamically feasible trajectories that are "near-optimal" in some sense.

One way to pursue, is to re-evaluate our notion of optimality. If finding a truly optimal solution is intractable, one might instead consider finding *pareto-optimal* solutions, where the computational load is considered as one of the objective functions to be minimized (*cf.* [6]). Pareto-optimality is a concept of multiple objective decision making [7] and, intuitively, a pareto-optimal solution is one in which no other solution can improve one objective without a simultaneous deterioration of at least one of the others. It should however be noted that in our case, not every pareto-optimal solution will be regarded as feasible. If just augmenting the objective function from the original OCP with a quantification of the computational load, one pareto-optimal solution would be not to optimize the trajectory at all. To see this, notice that the computational load would then be zero and thus any effort to improve the quality of the trajectory would require increased computational load. In particular, such a trivially pareto-optimal trajectory, would typically *not* end in the target set.

To set this right, it must be noted that although formulated as an OCP, finding a provably collision free path that is guaranteed to end in the target set must be given higher *priority* than the optimality properties thereof. This diversification, or ranking, of our objectives is quite natural since the optimal control formulation can be considered as a *tool* for choosing one single input in the set of controls that fulfill our minimum requirements[a], which in our particular case will be to generate collision free paths that lead us to the target set. A collision free vehicle path is called *safe*, while reaching the target set will hence-forth be referred to as *task completion.* As will be shown, the controller design of this paper have both provable safety properties, as well as a guaranteed finite time task completion.

The line of thought presented in this paper merges that of point-wise satisficing control [9, 10] and Receding Horizon Control (RHC) or Model Predictive Control (MPC) [11]. In RHC/MPC, the doubtful viability of long term optimization under uncertain conditions is adhered, so that instead of solving the OCP on the full interval, one repeatedly solves it on the interval $[t_c, t_c + T_p]$ instead. Here t_c denotes the current time instance and T_p is the planning horizon. Upon applying the first control element, measuring the obtained state and moving the time interval forward in time, the optimization step is iteratively performed. This closes the loop and obtains a certain robustness against modeling errors or disturbances. Unfortunately, it is known that in the absence of particular precautions, closed-loop stability[b] cannot be assured. In principle however, this issue is relatively easily tackled by introducing stability terms or constraints. In the literature (see *e.g.* [11]), several approaches to ensure stability of RHC/MPC schemes can be found, one of the most appealing approaches being that of utilizing a constrained control Lyapunov function (CLF) as terminal cost [12, 13].

As a matter of fact, it has been shown in [14], that OCP, RHC as well as the set of CLF-based continuous control designs (such as Sontag's universal formula [15], Freeman and Kokotović's min-norm formula [16], but also the satisficing control methods [9, 10]) can be viewed in a unified manner; namely that OCP and CLF-based methods are the two limiting cases of RHC when the planning horizon goes to infinity and zero respectively (see Figure 1). This enables us to adopt a *horizon independent* point of view, where the length of the planning horizon, T_p, is determined based on accuracy demands and computational resources, while inherently more global properties (such as stability or task completion) are handled by the monotonicity properties of the composite cost.

[a]Any solution meeting the minimum requirements is called *satisficing* [8]. For a more control-oriented presentation of the concept, consult [9, 10].

[b]Standard RHC/MPC is tailored for steady-state control or asymptotic stabilization to the origin in the Lyapunov sense. The notion of *task completion* considered in this paper is a *different problem*, namely it aims at controlling the plant into a target set which is not necessarily control invariant or contain any equilibrium points.

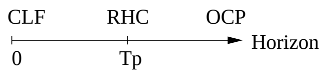


Figure 1. A unifying view on OCP, RHC and the CLF-based control methods.

A. Related work

Regarding safety concerns, reference should be made to the recent works of Schouwenaars *et. al* [17] and Kuwata and How [18, 19], who consider safe RHC of autonomous vehicles. Both approaches are however tailored for a 2D setting, and further utilize visibility graphs for environmental representation. Although visibility graph based methods are effective in urban terrain (where the buildings provide neat blocks and corner points to exploit), they are not suitable for the mountain terrain data used in this paper. In addition to regarding a 3D problem, our work shows that it is possible to address both safety and task completion issues in a unified manner without introducing any integer variables. It also differs from the mentioned papers by the fact that in our case, safety also renders task completion possible. This is simply due to an elaborate choice of the so called "set of safe states", in which the safety maneuver is to end. Moreover, as a subsidiary consequence, introducing the safety maneuver makes it possible to cope with hard real-time systems.

Task completion has been previously considered by Richards and How [20,21]. By augmenting the system with a binary "target state", that indicates whether the target set is reached or not, the authors end up with a hybrid system at hand. Task completion is then guaranteed by imposing a hard terminal equality constraint on the target state which restricts the trajectory candidates to those that end up in the terminal set at the end of the planning horizon. Although intuitively appealing, this is indeed a very restrictive and computationally demanding constraint that beside the introducement of binary states, require needlessly long planning horizons. In addition, early termination of the optimization routine may cause violation of the equality constraint and consequently jeopardize the task completion objective. The alternative solution outlined in this paper, argues that requiring monotonicity of the composite cost is sufficient to obtain the desired task completion property. This decouples the length of the planning horizon from task completion and thus allows us to choose the planning horizon only taking computational resources and real-time constraints into account.

The general framework utilized in this paper for path planning in three dimensional space, is reminiscent of that presented in [22]. In both papers, the global characteristics of the environment and mission objectives are captured in a functional, calculated off-line and passed to the on-line receding horizon controller as a terminal cost. However, safety and task completion concerns are the pivotal differences between these two papers. This is also a convenient point at which to mention that the possibility of updating the "off-line" computed terminal cost should not be overlooked. As pointed out in [22], the term "off-line" is rather to be interpreted as, at a much slower sampling rate than the control loop, *i.e.* in the order of tens of seconds. As new information about the environment or mission objectives is gathered when the mission unfolds, it can be processed and fed back regularly to the vehicle through an updated terminal cost, as discussed in *e.g.* [23].

Simultaneous arrival of multiple vehicles is an application of the wider class of consensus problems. Standard results in this field involve consensus problems described by unconstrained, scalar (or fully actuated) first order linear systems (see *e.g.* [24–26]). In our case however, the relation between the control (*i.e.* the vehicle acceleration), and the consensus quantity (which is the Estimated Time to Arrival, ETA) is neither unconstrained nor first order. However, by using a time-scale separation principle, we are able to achieve consensus by using standard Laplacian control. In a non receding horizon setting, synchronization of timing-critical missions have been considered in a number of papers, including [27, 28].

This paper is organized as follows; Section II introduces the trajectory optimization problem as well as some basic terminology used in this paper. Section III presents the problem formulation and clarifies the relation between the original OCP and the transcription thereof into a finite-dimensional nonlinear mathematical programming problem (NLP). Subsequently, the fundamental role of the safety maneuver and the implication of it on task completion is discussed in Section IV. Details on environment representation and terminal cost computation can be found in Section V. In Section VI, a quantitative comparison between the two competing objectives of optimality and computational tractability is made for the problem at hand. Section VII presents a small selection of the simulations made with the proposed trajectory planning algorithm while Section VIII extends these results to the multi-vehicle case. Finally, Section IX concludes this paper with some expository remarks.

II. Preliminaries

In this section, we review some standard background material and present the used terminology. Consequently, the more familiar reader may prefer to skip it at a first reading.
Consider the following trajectory optimization or optimal control problem (OCP):

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_{t_c}^{T} \mathcal{L}(x,u)\mathrm{d}t \\ \text{s.t.} \quad & \dot{x} = f(x) + g(x)u \\ & d(x,u) \leq 0 \\ & x(t_c) = x_c \in \mathbb{R}^n \\ & x(T) \in S_f \subseteq \mathbb{R}^n, \end{aligned} \tag{1}$$

where the state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$, the control $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$, the constraints $d : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^p$, the terminal time T is a possibly *free* variable, and t_c can be read as the "current" or "considered" time. All mappings are assumed to be sufficiently smooth and the dynamical system complete. To further unburden the discussion, we make a standing assumption throughout the paper, that all stated minimization problems with respect to u are well–posed and that the minimum is attained.

Optimal Control Problem

We consider henceforth only *feasible* state trajectories x, *i.e.* state histories such that all conditions in (1) are met for all times t and the closed target set S_f is reached in finite time $T \in [0,\infty)$. Define $\bar{\mathcal{U}}$ as the set of control functions such that u remains in $\mathcal{U}$ at all times and a feasible state trajectory x is generated. Let the *value function* $J : \mathcal{X} \times \bar{\mathcal{U}} \mapsto [0,\infty)$ be defined by

$$J(x(t_c),u) = \int_{t_c}^{T} \mathcal{L}(x(t),u(t))\,\mathrm{d}t, \quad t_c \in [0,T],$$

that is the cost to go from $x(t_c)$ to S_f using a control u. We assume that an optimum to (1) always exists and define accordingly the *optimal value function* $J^* : \mathcal{X} \mapsto [0,\infty)$ as

$$J^*(x(t_c)) = \min_{u\in\bar{\mathcal{U}}} \int_{t_c}^{T} \mathcal{L}(x(t),u(t))\,\mathrm{d}t, \quad t_c \in [0,T],$$

which thus represents the optimal cost to go from $x(t_c)$ to S_f. Without loss of generality we may assume that $\mathcal{L}(x(t_c),u(t_c)) = 0$ if and only if $x(t_c) \in S_f$ so that $J^*(x(t_c)) = 0$ is equivalent to $x(t_c) \in S_f$.

Receding Horizon Control

In the receding horizon setting, we consider a slightly modified OCP where the *planning horizon* $T_p \in [0, T - t_c]$ is given relative to some $t_c \in [0, T]$, and the *discarded optimal tail cost*, defined as

$$\min_{u \in \bar{\mathcal{U}}} \int_{t_c+T_p}^{T} \mathcal{L}(x(t), u(t))\, \mathrm{d}t,$$

is upper bounded by some *terminal cost*, $\Psi : \mathcal{X} \mapsto [0, \infty)$. In other words, we have that

$$\Psi(x(t_c + T_p)) \geq \min_{u \in \bar{\mathcal{U}}} \int_{t_c+T_p}^{T} \mathcal{L}(x(t), u(t))\, \mathrm{d}t.$$

The terminal cost, $\Psi(x(t_c+T_p))$, is meant to act as a more easily computed (but conservative) approximation to the optimal cost to go from $x(t_c + T_p)$ to S_f. With the aid of the terminal cost, the *composite cost* $\tilde{J} : \mathcal{X} \times \bar{\mathcal{U}} \mapsto [0, \infty)$, is defined as

$$\tilde{J}(x(t_c), u) \int_{t_c}^{t_c+T_p} \mathcal{L}(x(t), u(t))\, \mathrm{d}t + \Psi(x(t_c + T_p)).$$

Since the terminal cost acts as an upper bound on the discarded optimal tail cost we have

$$\begin{aligned} \tilde{J}(x(t_c), u) &= \int_{t_c}^{t_c+T_p} \mathcal{L}(x(t), u(t))\, \mathrm{d}t + \Psi(x(t_c + T_p)) \\ &\geq \int_{t_c}^{t_c+T_p} \mathcal{L}(x(t), u(t))\, \mathrm{d}t + \min_{u \in \bar{\mathcal{U}}} \int_{t_c+T_p}^{T} \mathcal{L}(x(t), u(t))\, \mathrm{d}t \\ &\geq \min_{u \in \bar{\mathcal{U}}} \int_{t_c}^{T} \mathcal{L}(x(t), u(t))\, \mathrm{d}t = J^*(x(t_c)), \end{aligned}$$

that is, the composite cost provides an upper bound on the optimal value function. Now, *if* the control u can be chosen such that $\tilde{J}$ is monotonically decreasing along a feasible trajectory, then $\tilde{J}$ acts as a Lyapunov–like function for the receding horizon controlled system. This will, together with a sandwiching argument, guarantee task completion for the original OCP, *i.e.* $J^*(x(T)) = 0$ for some $T < \infty$.
Here, one of the key issues that motivates the present work reveals itself. Namely, since Ψ is merely an approximation to the optimal cost to go, it will in general *not* be possible to find a control u that both satisfies all the constraints in (1), and cause a decrease in $\tilde{J}$ (unless Ψ is chosen as a constrained CLF, which will not be assumed in this paper). Using the terminology of [9, 10], the set of satisficing controls might be *empty.* In these cases, a reserve plan (namely using the augmented safety maneuver) turns out to be a useful tool for ensuring both safety and task completion.

Ideally, $\Psi(x(t_c + T_p))$ should be chosen to equal the optimal value function,

$$J^*(x(t_c + T_p)) = \min_{u \in \bar{\mathcal{U}}} \int_{t_c+T_p}^{T} \mathcal{L}(x, u)\mathrm{d}t,$$

which can be found by solving the Hamilton-Jacobi-Bellman partial differential equation (HJBE); a highly nontrivial task. If one would be able to solve HJBE for J^* and set the terminal cost equal to it, then the RHC scheme would coincide with that of solving the OCP on the full time interval, $[t_c,\ T]$.

III. Problem Formulation

We are now ready to state our trajectory planning problem. The aerial vehicle is modeled as a unit-mass point in $\mathbb{R}^3$ with bounded velocity and acceleration. The equations of motion are those of Newtonian mechanics so that the control inputs are the applied forces or accelerations (by virtue of the unit-mass assumption). Let $p = [x\ y\ z]^T \in \mathbb{R}^3$ denote the position of the vehicle in its work-space, $\mathbb{R}^3$, while $v = \dot{p} = [v_x\ v_y\ v_z]^T$ denotes its velocity. Let further

$$\mathcal{X} = \{(p, v) : p \in \mathbb{R}^3, v \in \mathbb{R}^3\},$$

denote the state-space. A state $s = (p, v) \in \mathcal{X}$, is a position-velocity-tuple and hence $\mathcal{X}$ is isomorphic to $\mathbb{R}^6$. Our objective is to find a minimal-cost trajectory from a specified initial vehicle position and velocity, *i.e.* from a given state $s_i = (p_i, v_i) \in \mathcal{X}$, to a final vehicle position with an *arbitrary* final velocity. Hence, the target set can be written as

$$S_f = \{(p, v) \in \mathcal{X} : p = p_f\},$$

where p_f is given. Note that S_f will in general not be an invariant set, hence there is no need to stay in the target set once it is entered.

Problem Transcription

For the actual design of the receding horizon controller, the OCP (1), which is an infinite-dimensional problem of choosing a control function in a given space, has to be *transcribed* into a finite-dimensional parameter selection problem. To this end, uniform temporal discretization is performed and the differential operator is approximated with a zero-order sampled dynamic model. Although it is possible to adopt higher order quadrature rules [29], a straightforward Riemannian sum is used to approximate the integral cost. The objective will be to minimize a combination of the L_2-norm of the applied control and the task completion time. Other types of mission objectives may be incorporated by the terminal cost. On top of the kinematic constraints, the dynamic constraints we consider are hard L_∞-norm constraints on both applied acceleration and vehicle velocity. Then, at each time-step, $k \in \mathbb{N} = \{1, 2, \dots\}$, upon measuring the current state, (p_c, v_c), the receding horizon control action, $a^*_{k,1}$ is computed. Here, the *control sequence* $a^*_k = [a^*_{k,1}, \cdots, a^*_{k,N}] \in \mathbb{R}^{3N}$, denotes a solution to the finite-dimensional optimal control problem[c];

$$\begin{aligned}
\underset{a_k}{\text{minimize}}\quad & \sum_{i=1}^{N-1} h(\|a_{k,i}\|_2^2 + \alpha) + \Psi(p_{k,N}) && && (2)\\
\text{s.t.}\quad & p_{k,i+1} = p_{k,i} + h\, v_{k,i} && i = 1, \cdots, N-1\\
& v_{k,i+1} = v_{k,i} + h\, a_{k,i} && i = 1, \cdots, N-1\\
& d(p_{k,i}) \le 0 && i = 1, \cdots, N\\
& \|v_{k,i}\|_\infty \le v_{\max} && i = 1, \cdots, N\\
& \|a_{k,i}\|_\infty \le a_{\max} && i = 1, \cdots, N\\
& p_{k,1} = p_c\\
& v_{k,1} = v_c\\
& a_{k,N} \in S_\varepsilon(s_{k,N}),
\end{aligned}$$

where $h > 0$ is the sampling interval and the design parameter $\alpha > 0$ determines the relative importance between time-optimality and energy efficiency. In addition, the set $S_\varepsilon(s_{k,N})$ is defined as follows.

Definition 1 (Target approaching controls). *The set of target approaching controls, $S_\varepsilon(s_{k,i})$, is defined to be the subset of control values in $\mathcal{U}$, such that*

$$\Big[\Psi(p_{k,i} + hv_{k,i}) - \Psi(p_{k,i})\Big] + h\mathcal{L}(s_{k,i}, a_{k,i}) \le -\varepsilon.$$

[c]Mind the difference between the time step index, $k \in \mathbb{N}$, and the index used in the RHC sequence, $i \in \{1, \cdots, N\}$.

Remark 1. An utmost important question, involves the non–emptiness of the set of target approaching controls. Under what conditions can we assure that S_ε is nonempty and what can be done in the case of infeasibility? It can be shown (*c.f.* [10]) that if Ψ is chosen as a constrained CLF and the stage cost, $\mathcal{L}$, is convex in the control variable, then S_ε is a convex, and hence connected set. The parameter ε, determines the size of this set and if fulfilling a simple inequality constraint, always turns S_ε nonempty. However, because of the apparent difficulties with choosing Ψ as a CLF, this work will recognize the existence of cases when S_ε turns out to be empty. The augmented safety maneuver, which will be introduced properly in Section IV, provides a solution to this infeasibility problem.

The motivation behind the nomenclature used for S_ε, will be clear from the proposition to follow. Let $a_k = [a_{k,1}, \cdots, a_{k,N}] \in \mathbb{R}^{3N}$ denote the control sequence used at time step k. Let further

$$a_{k+1} = \overleftarrow{T}(a_k, \star) = [a_{k,2}, \cdots, a_{k,N}, \star] \in \mathbb{R}^{3N},$$

denote the control sequence used at time step $k+1$. Here, $\overleftarrow{T}$ denotes a left shift operator and the star symbol, $\star$, denotes an arbitrary control element. These control sequences, give rise to the trajectories

$$\begin{cases} p_k = [p_{k,1}, \cdots, p_{k,N}] \\ v_k = [v_{k,1}, \cdots, v_{k,N}] \end{cases} \quad \text{and} \quad \begin{cases} p_{k+1} = \overleftarrow{T}(p_k, p_{k,N} + hv_{k,N}) = [p_{k,2}, \cdots, p_{k,N}, p_{k,N} + hv_{k,N}] \\ v_{k+1} = \overleftarrow{T}(v_k, v_{k,N} + ha_{k,N}) = [v_{k,2}, \cdots, v_{k,N}, v_{k,N} + ha_{k,N}] \end{cases}$$

respectively.

Proposition 1 (Target set approaching). *If choosing $a_{k,N} \in S_\varepsilon(s_{k,N})$, then*

$$\tilde{J}(p_{k+1,1}, a_{k+1}) < \tilde{J}(p_{k,1}, a_k),$$

that is, the composite cost at the next time step has been reduced, hence the target set is approaching.

Proof.

$$\begin{aligned}
\tilde{J}(p_{k+1,1}, a_{k+1}) &= \sum_{i=1}^{N-1} h\mathcal{L}(s_{k+1,i}, a_{k+1,i}) + \Psi(p_{k+1,N}) \\
&= \sum_{i=2}^{N} h\mathcal{L}(s_{k,i}, a_{k,i}) + \Psi(p_{k,N}) - \Psi(p_{k,N}) + \Psi(p_{k,N} + hv_{k,N}) + h\mathcal{L}(s_{k,1}, a_{k,1}) - h\mathcal{L}(s_{k,1}, a_{k,1}) \\
&= \sum_{i=1}^{N-1} h\mathcal{L}(s_{k,i}, a_{k,i}) + \Psi(p_{k,N}) - \Psi(p_{k,N}) + \Psi(p_{k,N} + hv_{k,N}) + h\mathcal{L}(s_{k,N}, a_{k,N}) - h\mathcal{L}(s_{k,1}, a_{k,1}) \\
&\leq \tilde{J}(p_{k,1}, a_k) - \varepsilon - h\mathcal{L}(s_{k,1}, a_{k,1}) < \tilde{J}(p_{k,1}, a_k).
\end{aligned}$$

Remark 2. Although insignificant for the proof of this particular proposition, recursive use of this result (in order to achieve target set *reaching*) requires

$$\star = a_{k+1,N} \in S_\varepsilon(s_{k+1,N}).$$

For the sake of completeness and to make the effect of the above described transcription clear, if we define a parameter vector,

$$\xi^k = [p_{k,1}, \cdots, p_{k,N},\ v_{k,1}, \cdots, v_{k,N},\ a_{k,1}, \cdots, a_{k,N}]^T \in \mathbb{R}^{9N},$$

an objective function,

$$\mathcal{F}(\xi^k) = \sum_{i=1}^{N-1} h(\|a_{k,i}\|_2^2 + \alpha) + \Psi(p_{k,N}),$$

and constraint functions, $\mathcal{G}^1(\xi^k)$ and $\mathcal{G}^2(\xi^k)$, according to

$$\mathcal{G}^1(\xi^k) = \begin{bmatrix} p_{k,i+1} - p_{k,i} - h\ v_{k,i} \\ v_{k,i+1} - v_{k,i} - h\ a_{k,i} \\ p_{k,1} - p_c \\ v_{k,1} - v_c \end{bmatrix}, \qquad \mathcal{G}^2(\xi^k) = \begin{bmatrix} d(p_{k,i}) \\ \|v_{k,i}\|_\infty - v_{\max} \\ \|a_{k,i}\|_\infty - a_{\max} \\ h(\|a_{k,N}\|_2^2 + \alpha) + [\Psi(p_{k,N} + h\ v_{k,N}) - \Psi(p_{k,N})] + \varepsilon \end{bmatrix},$$

then (2) can be written as

$$\begin{aligned} \underset{\xi^k}{\text{minimize}} \quad & \mathcal{F}(\xi^k) \\ \text{s.t.} \quad & \mathcal{G}^1(\xi^k) = 0 \\ & \mathcal{G}^2(\xi^k) \leq 0 \end{aligned} \tag{3}$$

which is the constrained NLP that needs to be solved on-line at each time-step, k.

IV. The Safety Maneuver and Task Completion

This section addresses the infeasibility problem that *may* originate from the approximative nature of the terminal cost, Ψ. Other sources of infeasibility include various optimization routine failures. In addition to the possibility of the set of target approaching controls being empty (see Remark 1), it must be noted that despite the hard anti-collision constraints $d(p_{k,i}) \leq 0$, which are only effective throughout the planning horizon, obstacle avoidance in the far future cannot be guaranteed *a priori*. The bottom line is that the terminal cost is most often calculated from a graph representation of the environment and might therefore lead to paths that turn out to be dynamically infeasible in the future.
Previously, in a 2D setting, Schouwenaars *et. al* [17] and Kuwata and How [18, 19], have considered safe RHC of autonomous vehicles. By constraining the computed path at each time-step to end on either a right, or a left turning collision free circle, where the vehicle can safely remain for an indefinite period of time (or at least until it runs out of fuel), the first mentioned authors account for safety. The authors of the latter mentioned papers make use of three circles to smoothen out all the corners of the straight line segments in the visibility graph, modify the cost map and thereby incorporate vehicle dynamics in the terminal cost. To the contrary, our work considers path planning in 3D space and further differs from previous results, by the fact that in our case, safety also renders task completion possible. This is simply due to an elaborate choice of the so called set of safe states, in which the safety maneuver is to end. We argue that a safe state should be chosen such that there is more options left than just aimlessly lingering in a loiter pattern. The choice made in this paper (vertically aligned vehicle), always leaves you with the possibility of continuing upward and thus, upon a full turn above the target point, p_f, renders task completion possible (see Figure 2). The remaining of this section is devoted to show how this simple observation underlies both the safety and task completions results, presented as Proposition 2 and 3 respectively.

Definition 2 (Safe path). *A path, $p_j = [x_j\ y_j\ z_j]^T, j \in \mathcal{I} \subset \mathbb{N}$, is called safe if and only if its vertical elements, z_j, are all located above the terrain surface,* i.e.

$$d(p_j) \triangleq H(x_j, y_j) + h_{min} - z_j \leq 0, \quad \forall\, j \in \mathcal{I}.$$

Here, $H(x_j, y_j)$ denotes the altitude of the mountain at the point (x_j, y_j) (as interrogated from a more detailed map than the one used for spatial decomposition), and $h_{min} > 0$ denotes the minimum clearance distance.

Definition 3 (Set of safe states). *A state, $s = (p, v) \in \mathcal{X}$, is called* safe *if there exists a safe, as well as dynamically feasible path linking p to the target set, S_f. The collection of all such states constitute the set of safe states. For the particular choice of mountainous terrain considered in this paper,*

$$S_s = \{(p, v) \in \mathcal{X} :\ d(p) \leq 0,\ v_x = v_y = 0,\ v_z \geq 0\},$$

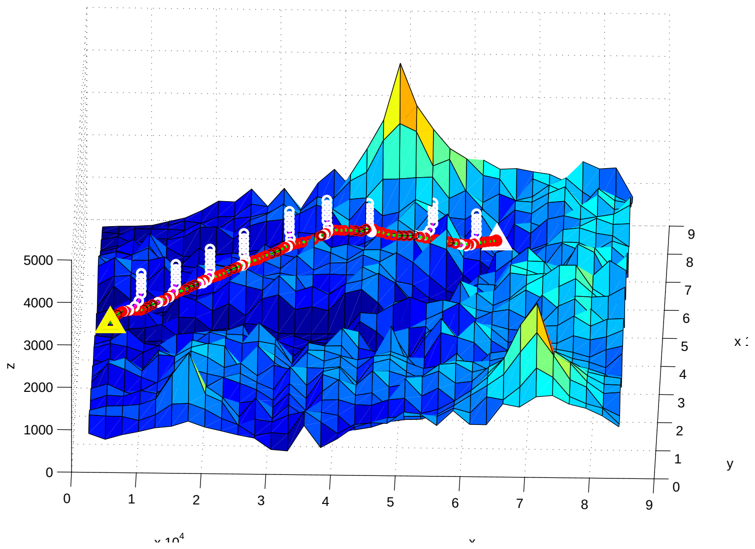


Figure 2. The safety maneuver connects the planned trajectory with the set of safe states from where the existence of a safe path to the target set is known to exist. Here, to simplify exposition, the augmented safety maneuver is shown at every tenth time step.

that is, the states where the vehicle is flying vertically above the terrain surface, constitute a distinguishable subset of the safe states.

The way to ensure collision avoidance, lies in the construction of a so called "safety maneuver",

$$\breve{p}_k = [\breve{p}_{k,1}, \cdots, \breve{p}_{k,\bar{N}}] \in \mathbb{R}^{3\bar{N}}, \quad \breve{v}_k = [\breve{v}_{k,1}, \cdots, \breve{v}_{k,\bar{N}}] \in \mathbb{R}^{3\bar{N}}, \quad \breve{a}_k = [\breve{a}_{k,1}, \cdots, \breve{a}_{k,\bar{N}}] \in \mathbb{R}^{3\bar{N}},$$

that connects the optimized trajectory,

$$p_k^* = [p_{k,1}^*, \cdots, p_{k,N}^*] \in \mathbb{R}^{3N}, \quad v_k^* = [v_{k,1}^*, \cdots, v_{k,N}^*] \in \mathbb{R}^{3N}, \quad a_k^* = [a_{k,1}^*, \cdots, a_{k,N}^*] \in \mathbb{R}^{3N},$$

to S_s. In other words, we require $(\breve{p}_{k,\bar{N}}, \breve{v}_{k,\bar{N}}) \in S_s$. Here, $\bar{N}$ denotes the maximum number of steps needed to perform the safety maneuver. How to determine $\bar{N}$ will be discussed in more detail here-below. This approach can be thought of as increasing the number of nodes in the temporal discretization from N to $N + \bar{N}$. However, in order to be computationally efficient, the augmented part, *i.e.* the components of the safety maneuver, are predetermined and do not take part in the optimization process. The freedom in predetermining the safety maneuver is limited by the following set of constraints

$$\begin{aligned}
&\breve{p}_{k,i+1} = \breve{p}_{k,i} + h\ \breve{v}_{k,i} && i = 1, \cdots, \bar{N}-1 \\
&\breve{v}_{k,i+1} = \breve{v}_{k,i} + h\ \breve{a}_{k,i} && i = 1, \cdots, \bar{N}-1 \\
&\breve{p}_{k,i}(z) \geq H(\breve{p}_{k,i}(x), \breve{p}_{k,i}(y)) + h_{\min} && i = 1, \cdots, \bar{N} \\
&\|\breve{v}_{k,i}\|_\infty \leq v_{\max} && i = 1, \cdots, \bar{N} \\
&\|\breve{a}_{k,i}\|_\infty \leq a_{\max} && i = 1, \cdots, \bar{N} \\
&(\breve{p}_{k,1}, \breve{v}_{k,1}) = (p_{k,N}^* + h\ v_{k,N}^*, v_{k,N}^* + h\ a_{k,N}^*) \\
&(\breve{p}_{k,\bar{N}}, \breve{v}_{k,\bar{N}}) \in S_s,
\end{aligned} \tag{4}$$

which restrict the safety maneuver to be a kinodynamically feasible trajectory for the aerial vehicle. In the following, the term *concatenated solution* refers to the augmentation of the optimized solution, (p_k^*, v_k^*, a_k^*), with the safety maneuver, $(\breve{p}_k, \breve{v}_k, \breve{a}_k)$.

Proposition 2 (Safety). *Assume the existence of a feasible concatenated solution at time $k = 1$. If the safety maneuver is chosen such that it fulfills the constraints* (4)*, then there exists a safe path for all future time steps, $k \in \mathbb{N}\backslash 1$.*

Proof. The proposition will be proved through mathematical induction. The initialization step is trivially fullfilled due to the assumption made in the proposition formulation. Next, assume the existence of a feasible concatenated solution at time step k. In particular, this assumption implies that

$$(\breve{p}_{k,\bar{N}}, \breve{v}_{k,\bar{N}}) \in S_s. \tag{5}$$

Remains to show the existence of a feasible concatenated solution at time step, $k+1$. To this end, using the left shift operator, $\overleftarrow{T}$, we set

$$\begin{aligned}
p_{k+1} &\triangleq \overleftarrow{T}(p_k^*, \breve{p}_{k,1}) &&= [p_{k,2}^*, \cdots, p_{k,N}^*, \breve{p}_{k,1}]\\
v_{k+1} &\triangleq \overleftarrow{T}(v_k^*, \breve{v}_{k,1}) &&= [v_{k,2}^*, \cdots, v_{k,N}^*, \breve{v}_{k,1}]\\
a_{k+1} &\triangleq \overleftarrow{T}(a_k^*, \breve{a}_{k,1}) &&= [a_{k,2}^*, \cdots, a_{k,N}^*, \breve{a}_{k,1}]\\
\breve{p}_{k+1} &\triangleq \overleftarrow{T}(\breve{p}_k, \breve{p}_{k,\bar{N}} + h\breve{v}_{k,\bar{N}}) &&= [\breve{p}_{k,2}, \cdots, \breve{p}_{k,\bar{N}}, \breve{p}_{k,\bar{N}} + h\breve{v}_{k,\bar{N}}]\\
\breve{v}_{k+1} &\triangleq \overleftarrow{T}(\breve{v}_k, \breve{v}_{k,\bar{N}}) &&= [\breve{v}_{k,2}, \cdots, \breve{v}_{k,\bar{N}}, \breve{v}_{k,\bar{N}}]\\
\breve{a}_{k+1} &\triangleq \overleftarrow{T}(\breve{a}_k, \bar{0}) &&= [\breve{a}_{k,2}, \cdots, \breve{a}_{k,\bar{N}}, \bar{0}].
\end{aligned}$$

Now, from Definition (3) and Equation (5) it follows that

$$(\breve{p}_{k+1,\bar{N}}, \breve{v}_{k+1,\bar{N}}) = (\breve{p}_{k,\bar{N}} + h\breve{v}_{k,\bar{N}}, \breve{v}_{k,\bar{N}}) \in S_s.$$

Consequently,

$$\begin{aligned}
\breve{p}_{k+1,\bar{N}}(z) &= \breve{p}_{k,\bar{N}}(z) + h\breve{v}_{k,\bar{N}}(z) \geq \breve{p}_{k,\bar{N}}(z) \geq H(\breve{p}_{k,\bar{N}}(x), \breve{p}_{k,\bar{N}}(y)) + h_{\min}\\
&= H(\breve{p}_{k,\bar{N}}(x) + h\breve{v}_{k,\bar{N}}(x), \breve{p}_{k,\bar{N}}(y) + h\breve{v}_{k,\bar{N}}(y)) + h_{\min} = H(\breve{p}_{k+1,\bar{N}}(x), \breve{p}_{k+1,\bar{N}}(y)) + h_{\min}.
\end{aligned}$$

It is then straightforward to verify that $(\breve{p}_{k+1}, \breve{v}_{k+1}, \breve{a}_{k+1})$ fulfills the remaining constraints of (4) and thus, augmented with $(p_{k+1}, v_{k+1}, a_{k+1})$ serves as a feasible concatenated solution at time step $k+1$. ∎

Remark 3. Mind the intentional digression from the notation used earlier; solution $(p_{k+1}, v_{k+1}, a_{k+1})$ is defined by the left shift operator, $\overleftarrow{T}$, and does not necessarily equal $(p_{k+1}^*, v_{k+1}^*, a_{k+1}^*)$ - the output obtained from solving the NLP. This is an important point to make, since the essence of Proposition 2 is that if the NLP output is missing or do not satisfy the feasibility constraints of (3) and (4) at the time step $k+1$, then one always have the choice of taking

$$(p_{k+1}^*, v_{k+1}^*, a_{k+1}^*) = (p_{k+1}, v_{k+1}, a_{k+1}),$$

that is, sticking with the demonstrably feasible solution obtained from the previous time step, k. In this view, the introduction of the safety maneuver also makes it possible to cope with *hard* real-time constraints; namely it provides an applicable option even when the NLP solver has not converged, or terminates abnormally.

Remark 4. The proposed safety maneuver requires that the vehicle has a thrust-to-weight ratio larger than one, whereas many of today's UAVs have a value in the order of 0.5. However, it is possible to modify the theory to cover also the latter class of cases by using a somewhat elaborate set of safe states utilizing maximal terrain inclination [31].

Proposition 3 (Task completion). *Assuming the existence of a feasible concatenated solution at time $k = 1$, the trajectory planner will generate safe paths that end in the target set, S_f. Hence, task completion is guaranteed.*

Proof. Let $\bar{k}$ denote the largest time step for which there exist control inputs $a_{m,N} \in S_{\varepsilon_m}(s_{m,N})$, giving rise to feasible concatenated solutions for all $m \leq \bar{k}$. Here, ε_m denotes the parameter used in time step m. From the initial assumption made, we have $\bar{k} \geq 1$. Now, if

$$\sum_{m=1}^{\bar{k}} \varepsilon_m + h\bar{k}\alpha \geq \tilde{J}(p_{1,1}, a_1),$$

task completion follows directly by a recursive call on Proposition 1 (*cf.* Remark 2), the non-negativity of $\tilde{J}$ and a sandwiching argument. Else, by setting

$$(p^*_{\bar{k}+l}, v^*_{\bar{k}+l}, a^*_{\bar{k}+l}) = (p_{\bar{k}+l}, v_{\bar{k}+l}, a_{\bar{k}+l})$$

for $l = 1, \cdots, \bar{N}$, and iteratively adopting the argument used in the proof of Proposition 2, we get

$$(p_{\bar{k}+\bar{N}+N,1}, v_{\bar{k}+\bar{N}+N,1}) \in S_s.$$

Task completion is then guaranteed due to Definition 3; namely the existence of a kinodynamically feasible path connecting the points of S_s to the target set. ■

Since $\bar{k}$ is ultimately determined by the terminal cost, Proposition 3 also reveals the importance of choosing Ψ with a small degree of conservatism.

Number of Steps in the Safety Maneuver

As previously mentioned, the trajectory optimization part is horizon independent, that is, N can be chosen arbitrarily. However, since the safety maneuver augmented at the end of the optimized trajectory takes some time steps to perform, and we do not want it to affect the optimization routine, we must give the vehicle enough time to perform the safety maneuver. Next, we put an upper bound on the number of steps needed to obtain this.

In order to be computationally efficient, there is no optimization step involved in the safety maneuver. Due to the constraints (4), it is only the three sequences of components of the safety maneuver control input, $\breve{a}$, that need to be predetermined. In the z-direction, $\breve{a}_z$ is predetermined by setting $\breve{a}_z = a_{\max}$. This, in order to increase the altitude as fast as possible and thereby minimize the risk of collision. In the x- and y-direction, the objective is to assign the safety maneuver control input such that the vehicle is transferred to a safe state, *i.e.* becomes vertically aligned (see Definition 3 and Figure 2). To do so, we set

$$\breve{a}_x = \begin{cases} -\text{sign}(\breve{v}_x)\ a_{\max} & \text{if } \breve{v}_x \geq h\ a_{\max} \\ -\frac{\breve{v}_x}{h} & \text{if } \breve{v}_x < h\ a_{\max}. \end{cases}$$

A similar argument is used to determine $\breve{a}_y$.

The worst case scenario - from the safety maneuver point of view - occurs when the z-component of the velocity vector is directed downwards with maximum allowable speed, *i.e.* $\breve{v}_z = -v_{\max}$. In this case, the number of steps required to accelerate the vehicle so that it moves upwards ($\breve{v}_z \not< 0$) equals

$$\bar{N} \triangleq \lceil \frac{v_{\max}}{h\ a_{\max}} \rceil.$$

Naturally, this bound is also sufficient for aligning the vehicle vertically, *i.e.* bringing $\breve{v}_x$ and $\breve{v}_y$ to zero. From this, we conclude that $\bar{N}$ serves as the maximum number of steps needed to perform the safety maneuver, without affecting the trajectory optimization part.

V. Environment Representation and Terminal Cost Computation

For environmental representation, real terrain elevation data over the Cascade mountains, WA, have been used (see Figure 3). The dataset used is a subset extracted from the one appearing in Reference [30][d]. The full-resolution elevation image, is made up of $16,385 \times 16,385$ nodes at 10 meters horizontal spacing. The vertical resolution is 0.1 meters. This dataset occupies roughly 5 GB on disk and is therefore impractical to work with. However, as will be seen from the simulation results but also pointed out in [22], the environment should be decomposed in a manner that is consistent with the maneuvering capabilities of the vehicle. Therefore, this high level of accuracy is not needed to capture the global characteristics of the environment by the terminal cost, Ψ. The lower-resolution maps used in the simulations have therefore been sub-sampled at every 16th and 256th instance, resulting in a inter-pixel spacing of 160 and 2560 meters respectively. In the vertical direction, there are five horizontal layers with 600 meters in between. The vertical positions of each node depend on the altitude of the terrain at that particular point of the map. The non-uniform grid built this way, can be seen as stretching out the layers of a uniform grid on the terrain surface.

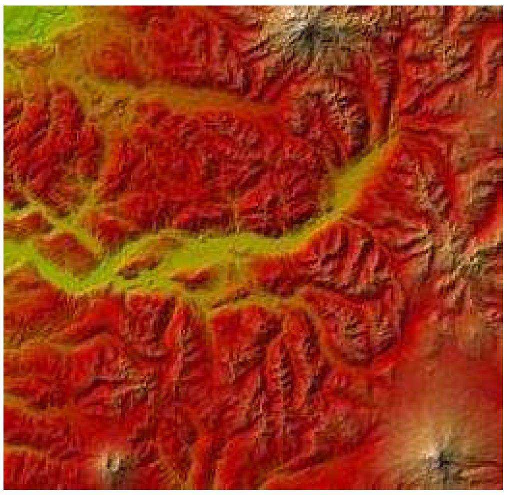


Figure 3. The terrain elevation map used in the simulations represents an area of more than 82km ×82km taken from the Cascade range, WA. It contains the summit of Mt. Rainier, Mt. Adams and Mt. St. Helen's.

Upon this spatial-decomposition procedure, we end up with a grid having only $33 \times 33 \times 5$ nodes. Given the target node, $n_f = S_f$, the terminal cost at any node, $\Psi(n_i)$, is an approximation of the cost to go from n_i to n_f. Naturally, we have that $\Psi(n_f) = 0$. There is a cost of c_{ij} to be associated with a transition between node i and j. In the nominal case, this cost will be taken proportional to the Euclidean distance [e]. However, if there exist threat zones or other preferences regarding the path of the vehicle, these costs are readily modified to account for them as well. For instance, the transition cost is to be set to a higher value within the detection area of a known SAM-site or radar. The actual values of $\Psi(n_i)$ are calculated off-line using an implementation based on Dijkstra's algorithm, which returns the cheapest path (as defined by the costs, c_{ij}) from node n_f to all other nodes. The value of the terminal cost at an arbitrary position in the grid is then found using interpolation between the nodes surrounding that point. The interpolation routine used, can be shown to be consistent and free from local minimums inside each cube in the grid.
Finally, as pointed out in Section I, the term "off-line" here is rather to be interpreted as at a much slower sampling rate than the control loop, *i.e.* in the order of tens of seconds. As new information about the environment or mission objectives is gathered as the mission unfolds, it can be processed and fed back

[d]This freely available data can also be found at `http://duff.geology.washington.edu/data/raster/tenmeter/onebytwo10/`.

[e]Beside its computational simplicity, this choice is motivated by its utility for the simultaneous arrival problem considered in Section VIII.

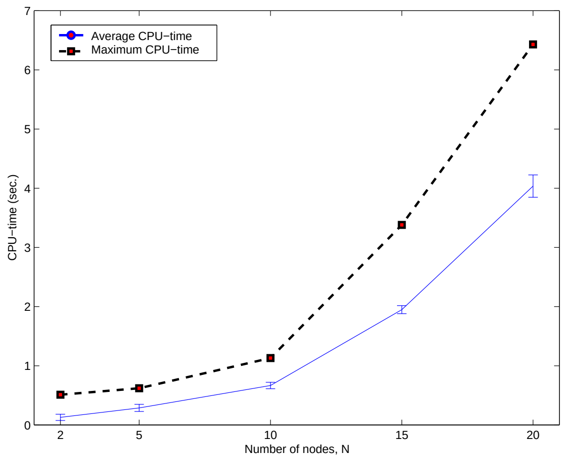
regularly to the vehicle through an updated terminal cost (*cf.* [22, 23]). With the non-optimized Matlab code used, it takes on average 15.8 seconds to both built the graph representing the environment and calculate the terminal cost. Modifying an existing graph (in order to incorporate mission objectives), takes only 1.5 seconds on average. All computations have been performed on a shared Linux cluster, using one of its four 2.80 GHz Intel® Xeon processors.

VI. Optimality and Computational Load

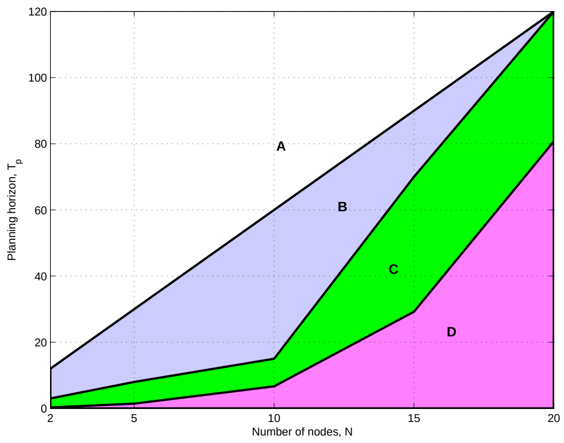
In this section, we first flesh out the relation between the prediction horizon, T_p, and the number of nodes used for the temporal discretization, N. Then, we empirically verify the intuition that a longer prediction horizon (paired with a suitable N), generates solutions closer to optimum. As expected, the price for this reduction in the objective function has to be payed in terms of increased computational load. It is then important to set up a quantitative comparison between these two competing objectives. As a particular example, it is shown that by accepting a deterioration of less than 3% in the objective function, it is possible to reduce the run-times with more than 38%.

Prediction Horizon and Temporal Discretization

Generally, trajectory optimization run-times are critically depending on the number of variables in the underlying NLP. These in turn, are proportional to the number of nodes in the temporal discretization, N. How the solution time varies as a function of N is depending on the particular NLP solver used. Figure 4(a) illustrates the average, and maximum run-times of NPOPT; the solver used for all simulations here-within. NPOPT is an updated version of NPSOL; a sequential quadratic programming (SQP) based method for solving NLPs [32]. It it worth mentioning, that the average and maximum have been taken both over a number of planning horizons (typically 10 different values) and iterations (typically $100 - 150$ iterations per planning horizon).



(a) The increasing average and maximum run-times of NPOPT as a function of N. Computations are performed on a shared Linux cluster, using one of its four 2.80 GHz Intel® Xeon processors.



(b) Area A and D: set of infeasible planning horizons. B: feasible but non-optimal planning horizons. C: strip of appropriate planning horizons.

Figure 4.

The error bars in Figure 4(a) represent deviation from the mean value when different planning horizons are chosen. From the raw data, it can be extracted that there is no dependency between these deviations and the length of the planning horizon, T_p. Hence these relatively small error bars, rather stem from the naturally existing fluctuations in solution times of NLPs. It then follows that the choice of N, is to a large extent restricted by the real-time computational requirement.

Once N has been chosen depending on computational resources[f], a guideline for choosing an appropriate planning horizon is as follows. A lower bound for it is achieved by the requirement that the sampling time,

$$h = \frac{T_p}{N-1},$$

must be large enough to nest the average run-time. Since constraint fulfillment can only be imposed at the sampling instances, a hard upper bound for T_p can be found by choosing a highest allowable inter-node spacing. By inter-node spacing, d, we mean

$$d = \max_{k \in \{2,\dots,N\}} \|p_k - p_{k-1}\| \leq d_{\max}, \tag{6}$$

that is, the maximum distance (in the workspace) between two subsequent sampling instances. However, simulations indicate the existence of a more vague upper bound, above which a deterioration in the objective function arises. All these three bounds are depicted in Figure 4(b). It is to be interpreted as follows. For a given N, all the planning horizons lying in area A (above the first mentioned upper bound) are infeasible since they violate the highest allowable inter-node spacing constraint (6) and are hence to be disregarded. Although feasible, the planning horizons of strip B are non-optimal in the sense that there always exists a smaller T_p in C that gives a lower value on the objective function. Moving on to strip C, it constitutes the set of applicable planning horizons. It is in this sector that the choice of T_p should be made. Finally, all planning horizons in area D (below the lower bound) give rise to sampling times smaller than the average run-time. The optimization routine will then typically not have enough time to converge. In this sense, all T_p in D are to be considered as infeasible.

A Quantitative Comparison Method

In general, a longer prediction horizon - paired with a suitable N as extracted from Figure 4(b) - generates solutions closer to optimum. As seen from Figure 5 and Table 1, longer prediction horizon gives rise to shorter trajectories that in addition require less control effort, *i.e.* have lower value on the objective function. Referring to Figure 5, as T_p increases, the planner becomes more predictive and “cuts corners”, resulting in shorter trajectories. Regarding the objective function, it can be noted from the fourth column of Table 1, that the control effort is also inversely proportional to the horizon length. However, as evident from the last two columns of Table 1, this brings a dramatic increase in average and maximum run-times.

When going from one planning horizon to another, two important issues are those of a possible decrease in the objective function (relative optimality) and the increased run-time associated with it (relative computational load). Table 2 puts these two aspects in perspective. There, comparison can be made between the relative increase in the objective function and the relative reduction in run-time, when going from one planning horizon, T_{p_1} to a shorter one, T_{p_2}. This allows us to quantify the “select-ability” and “reject-ability” properties of this choice. As a particular example, it can be seen from the last row of Table 2, that by accepting a deterioration of less than 3% in the objective function, it is possible to reduce the run-times with more than 38% by reducing T_p from twenty to fifteen seconds.

[f]The horizon independent approach of this paper, allows us to choose N only taking computational resources and real-time objectives into account; this without jeopardizing task completion/stability. Other works, where the length of the planning horizon is involved in the task completion/stability proof, do not enjoy this characteristic.

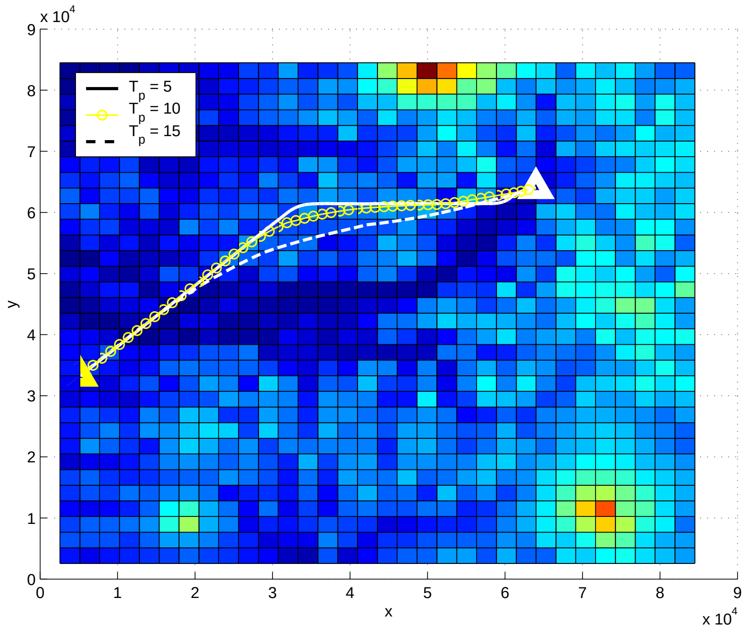


Figure 5. The impact of varying the planing horizon length, $T_p \in \{5, 10, 15\}$. As T_p increases, the planner becomes more predictive and "cuts corners".

T_p	N	Path length	$\|a\|_2$	Mean CPU-time	Max CPU-time
5 s	4	71 080 m	6.07	0.17 s	0.37 s
10 s	7	70 382 m	4.37	0.35 s	0.74 s
15 s	10	69 919 m	3.72	0.62 s	1.07 s
20 s	13	69 551 m	3.62	1.01 s	1.51 s

Table 1. Longer prediction horizon generates solutions with lower value on the objective function. The price of that being the evident increase in computational load.

T_{p_1}	$\rightarrow$	T_{p_2}	Rel. optimality	Rel. comp. load (mean)	Rel. comp. load (max)
10	$\rightarrow$	5 s	+38.9%	−52.3%	−50.0%
15	$\rightarrow$	10 s	+17.4%	−43.2%	−30.84%
20	$\rightarrow$	15 s	+ 2.9%	−38.6%	−29.14%

Table 2. The impact of the length of T_p on the optimality and computational load. By accepting a deterioration of less than 3% in the objective function, it is possible to reduce the run-times with more than 38%.

VII. Simulations

In this section, a small selection of the simulations made with the proposed trajectory planning algorithm is presented. The intention is to merely illustrate two specific properties of it; ability to minimize threat exposure, and robustness. This is done is Section A and B respectively.

A. Threat Exposure

As earlier depicted, it is the terminal cost, $\Psi(p)$, that captures the global characteristics of both the environment and the mission objectives. This is readily done by varying the costs, c_{ij}, in the graph representation of the environment. Figure 6 shows the effect of switching on a radar having a detection radius of 10km. The position of the radar is marked with a black triangle, while yellow circles are used to map out the volume where the vehicle is visible to the radar. The path with circles, shows the outcome of the trajectory planner when the radar is not accounted for. Unaware of its existence, the generated path passes right through the detection area of the radar. The other path, namely the one marked with squares, shows the outcome when the terminal cost incorporates the radar. The threat exposure is now minimized by flying at a much lower altitude, utilizing the protection provided by the terrain and thereby avoiding radar detection.

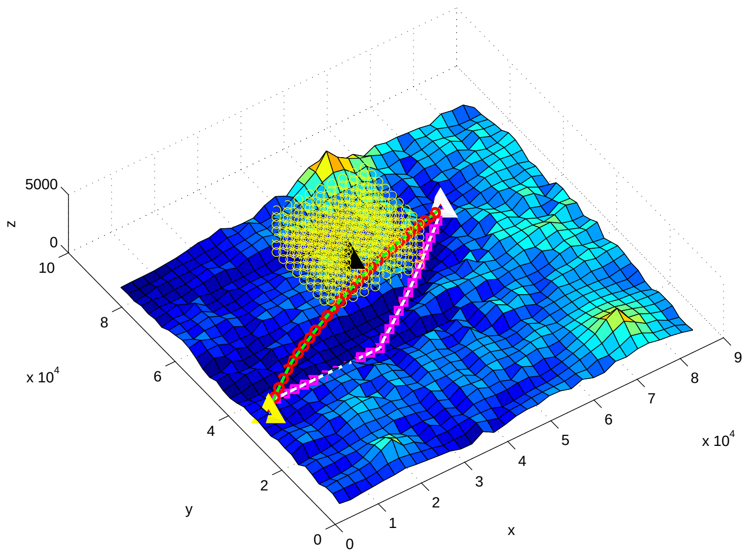


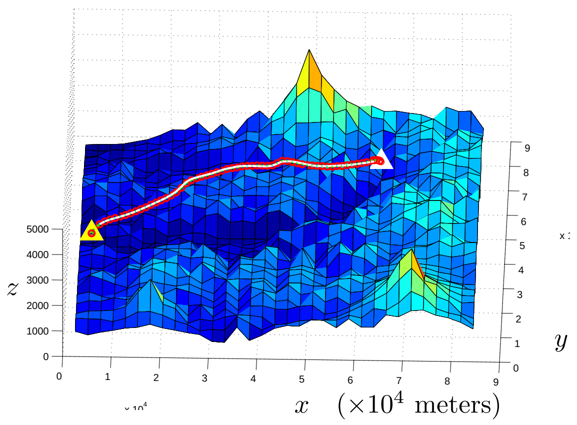
Figure 6. The effect of threat exposure on the generated path.

B. Robustness

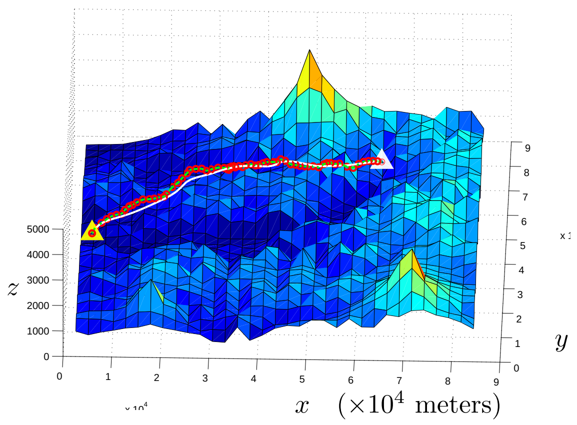
One of the most prominent characteristics of adopting a RHC scheme is that, by reducing the computational effort drastically, it gives us the possibility to repeatedly solve the NLP on-line with the current state as a new initial value. This way, feedback is incorporated and a certain degree of robustness is obtained. Next, to put the robustness properties of the trajectory planner to test, the existence of parametric uncertainty, measurement noise and other disturbances (such as wind gust or plant-model mismatch) is introduced. To this end, the nominal update equation, $p_{k+1,1} = p_{k,2}$, is modified to

$$p_{k+1,1} = p_{k,2} + w,$$

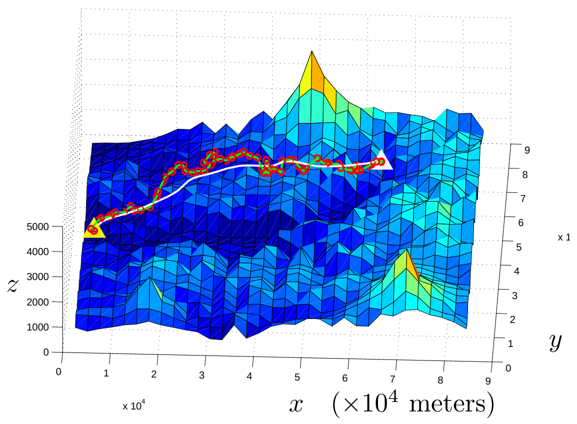
where w is a uniformly distributed noise parameter, $U(-\bar{w}, \bar{w})$. This modification implies that we, at the next time instance, will not move exactly to the nominal position we aimed for but rather to a random point in its vicinity. In what follows, in order to isolate the effect of the noise parameter, we set $T_p = 10$, $N = 6$, and study the generated paths and objective function (control effort) as $\bar{w}$ varies in the interval $[0 \quad 0.5]$. Figure 7 shows the generated paths corresponding to four increasing values on the noise parameter. As can be seen in the third column of Table 3, the disturbance corresponds to a displacement of more than 40%. Table 3 offers additional insight into the simulations made. In addition to the particular values of $\bar{w}$ that



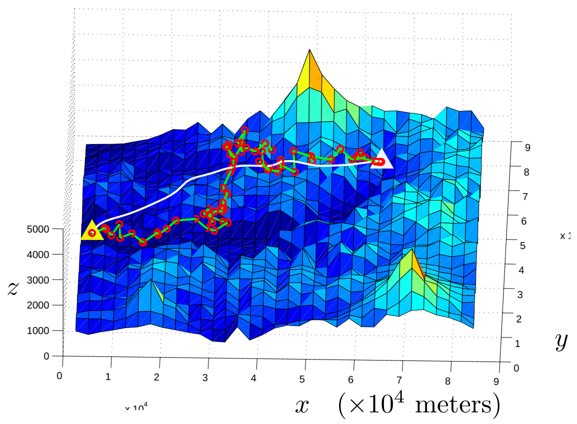
(a) $\bar{w} = 0.01$



(b) $\bar{w} = 0.1$



(c) $\bar{w} = 0.3$



(d) $\bar{w} = 0.5$

Figure 7. The effect of the noise parameter, w, on the generated paths. For the sake of reference, the nominal path ($\bar{w} = 0$), has been sketched with a white solid line. As $\bar{w}$ increases, the offset from the nominal path becomes more evident, but is repeatedly suppressed by the planner.

have been chosen, the actual effect of it, expressed as maximum offset (in meters) from the nominal position, $p_{k,2}$, can be read from the second column. The relative disturbance, *i.e.* the ratio between the maximum offset and the maximum inter-node spacing, d (see Equation (6)), serves as a comparative measure of the size of the disturbance. Finally, as seen in the fourth column, the control effort increases as a function of $\bar{w}$.

$\bar{w}$	Max. offset	Relative disturbance	$\|a\|_2$
0.0	–	–	4.88
0.01	$\pm$ 37 m	2.5%	4.98
0.05	$\pm$ 184 m	10.8%	7.94
0.1	$\pm$ 367 m	20.3%	15.12
0.2	$\pm$ 734 m	29.6%	17.32
0.3	$\pm$1101 m	36.0%	30.48
0.5	$\pm$1835 m	42.4%	48.73

Table 3. Impact of the added disturbance.

VIII. Simultaneous Arrival of Multiple Aerial-Vehicles

In this section we extend the trajectory optimization problem considered in Section III to the multi-vehicle case. In a centralized setting, such generalization is straightforward with inter-vehicle collision avoidance and increased computational effort as the main hurdles. The first issue can be handled explicitly by the trajectory planner by imposing anti-collision constraints such as safety zones around each vehicle. To hold the computational burden in check, parallel and distributed computing techniques might be considered. A decentralized RHC scheme to generate collision free trajectories have been proposed in [33] where the vehicles, based on local information, *sequentially* plan their individual trajectories. Further, in the venue of formation stabilization of a group of vehicles that are only coupled through the cost function, [34] proposes a distributed implementation of RHC. In this paper, we outset from these works and pay attention to cooperative decision making and consensus seeking in the multi-vehicle network. In particular, we are interested in the problem of agreeing on a simultaneous arrival time. Task synchronization gives major strategical advantages for a large class of mission scenarios, including the case when jamming the enemy radar.

In what follows, in Section A, we present the framework that will capture this consensus problem of ours and review some established results. Standard results in this field involve consensus problems described by unconstrained, scalar (or fully actuated) first order linear systems (see *e.g.* [24–26]). Extensions in the direction of time dependent interaction topology and communication delays have also been made [35–37]. In our case however, the relation between the control (*i.e.* the vehicle acceleration), and the consensus quantity (which is the Estimated Time to Arrival, ETA) is neither unconstrained nor first order. To remedy this, our design study is presented, where we propose a time-scale separation principle (Section B) which gives rise to simple equations of motion for the ETA. The interaction between between the consensus planner and the trajectory planner is discussed in Section C while simulation results are presented in Section D.

A. Consensus Problems on Weighted Graphs

Consider a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ representing the interaction topology of the network of vehicles. Here $\mathcal{V}$ denotes the set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, while $\mathcal{A}$ is a weighted adjacency matrix, $\mathcal{A} = [a_{lm}]$, with non-negative elements, a_{lm}. Each vertex $\mathrm{v}_m \in \mathcal{V}$ is an aerial-vehicle. Let $n = |\mathcal{V}|$ denote the cardinality of the vertex set, *i.e.* the total number of vehicles in the group, so that $m \in \mathbb{N}_n \triangleq \{1, \cdots, n\}$.

There is an edge $e_{lm} = (\mathrm{v}_l, \mathrm{v}_m)$ between any two vertices v_l and v_m, if and only if the two vehicles can communicate. The elements of the adjacency matrix $\mathcal{A}$ are associated with the edges of the graph through $e_{lm} \in \mathcal{E} \Longleftrightarrow a_{lm} > 0$.
In this work, we assume the information flow, and hence the graph $\mathcal{G}$, to be undirected and connected, *i.e.* there is a path connecting any two vertices of the graph. If we let $\mathcal{N}_m = \{l \in \mathbb{N}_n : e_{lm} \in \mathcal{E}\}$ denote the set of neighbors of vehicle m, assuming $\mathcal{G}$ to be connected implies that $\mathcal{N}_m \neq \emptyset$, for all $m \in \mathbb{N}_n$.

Let $\eta_m(t) \in \mathbb{R}$ denote the *consensus state* of vehicle m at time t. In our case η_m represents the Estimated Time to Arrival (ETA) for vehicle m. Simultaneous arrival then means reaching consensus regarding the ETA *i.e.*

$$\eta(t) = [\eta_1(t), \cdots, \eta_n(t)] \to \eta_d \mathbf{1}, \quad \text{as} \quad t \to \infty,$$

with $\mathbf{1} = [1, \cdots, 1] \in \mathbb{R}^n$ and the *consensus value* $\eta_d \in \mathbb{R}$.
Assuming the simple first-order dynamics[g]

$$\dot{\eta}_m = u_m, \quad \forall m \in \mathbb{N}_n, \tag{7}$$

it has been shown (see *e.g.* [25, 35]), that the following distributed control law (or *protocol*)

$$u_m = \sum_{l \in \mathcal{N}_m} a_{lm}(\eta_l - \eta_m) \tag{8}$$

will globally exponentially reach consensus with the speed of the algebraic connectivity of the graph (see *e.g.* Theorem 8 in [35]). Protocol (8) can be referred to as the "Laplacian protocol". The reason for this is apparent when considering the closed-loop network dynamics

$$\dot{\eta}(t) = \Delta\eta(t) = -L\eta(t), \tag{9}$$

where Δ denotes the discrete Laplace operator and L is the Laplacian matrix of the graph $\mathcal{G}$, defined as

$$L = \mathrm{diag}(d_1, \cdots, d_n) - \mathcal{A}.$$

Here d_m denotes the valency (*i.e.* the in/out degree) of vertex m.

B. The Time-scale Separation Principle

Standard results in the field of cooperative control and decision making involve consensus problems described by unconstrained, scalar (or fully actuated) first order linear systems (see *e.g.* [24–26]). As previously mentioned, for any vehicle in the group, the relation between the actual control inputs (*i.e.* the vehicle acceleration, a_m) and the consensus state (which is the ETA, η_m) is neither unconstrained nor first order. In this section however, by utilizing a time-scale separation principle and introducing the artificial consensus planner control inputs, u_m, we are able to still motivate the use of the unconstrained first order dynamics (7).

To set stage for the discussion, it might be fruitful to make a minor digression to the principle of guidance and control. Traditionally, the problem of steering a vehicle to its target is broken down into (at least) two subproblems: the problem of *trajectory optimization* (or guidance) and the problem of *auto-pilot design* (or control). This can be viewed as a control system having two degree of freedom; an inner loop (the auto-pilot) and an outer loop (the trajectory optimizer) (see Figure 8). The pivotal idea that justifies this separation is the multiple time-scales on which guidance and control occurs on. The discrete-time guidance problem considered in this paper occurs on a relatively slow time-scale (in the order of seconds). The inner control loop on the other hand takes place on a much faster time-scale (in the order of milliseconds) and is most often based on a dynamic model of the vehicle. It is noteworthy that by virtue of this separation,

[g]Technically, we do not have this dynamics at hand, but in a discrete setting, we can use a time-scale separation principle to still motivate the use of this (see Section B for details).

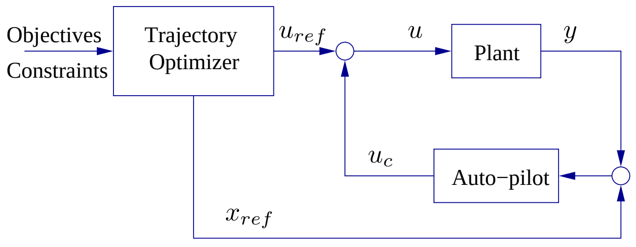


Figure 8. The two level separation of the control system is possible due to the multiple time-scales of guidance and control.

only *suboptimal* solutions can in general be found, but the advantage is that the details of the (nonlinear) dynamics of the vehicle only enters into the trajectory optimization part as relatively simple conditions on the reference trajectory. This separation thus justifies the simple linear dynamics used to describe the aerial vehicle in (2).

As for the consensus problem of ours, we imitate the above described idea of time-scale separation principle and design the consensus planner on a fast time-scale. More precisely, between any two subsequent time-steps in the discrete RHC setting (*i.e.* between time-steps k and $k+1$), we adopt a consensus problem on the fast time-scale, where each vehicle, v_m, shares its consensus state η_m with its neighbors, $\mathcal{N}_m$. A schematic block-diagram representation of the multi-vehicle network can be seen in Figure 9. By possibly scaling the Laplace protocol, its exponential convergence rate implies practical convergence of the consensus states at the end of the k^{th} time-step. Subsequently, the consensus planner sends back the consensus value (*i.e.* the desired ETA for all vehicles), denoted η_d in Figure 9. The trajectory planner is then to take η_d into account when planning the trajectory in the next time-step. How to do this, as well as how to calculate η_m, is discussed in Section C.

By virtue of this time-scale separation, the freedom of design of the consensus planner box is obtained. To this end, the artificial consensus planner control inputs, u_m, are introduced. Further, the dynamics of the state of the consensus planner (*i.e.* η_m) may have the linear first order dynamics (7).

Remark 5. As pointed out in [38], all the previously mentioned works on consensus problems (*i.e.* References [24–28, 35–37]) assume there is a global clock that synchronizes the state updates. This ruins the truly decentralized touch of the approaches. However, based on the time-scale separation principle depicted above, we argue that by introducing the clock as another *actuated* consensus state in the consensus planner (on the fast time-scale), the vehicles can reach consensus regarding the global clock in the same way as with the ETA.

C. Consensus Planner Interaction

As depicted in Figure 9, the consensus planner of the m^{th} vehicle receives the ETA, η_m, at the beginning of each time-step. At the end of the time-step, the consensus planner returns the consensus value, η_d, to the trajectory planner. Two important questions regarding this interaction are:

1. How the trajectory planner should estimate the time to arrival (*i.e.* calculate η_m) based on the output from the finite-dimensional optimization problem (3).

2. How the trajectory planner should take η_d into account when planning the trajectory in the next time-step.

These two issues are further discussed here-below.

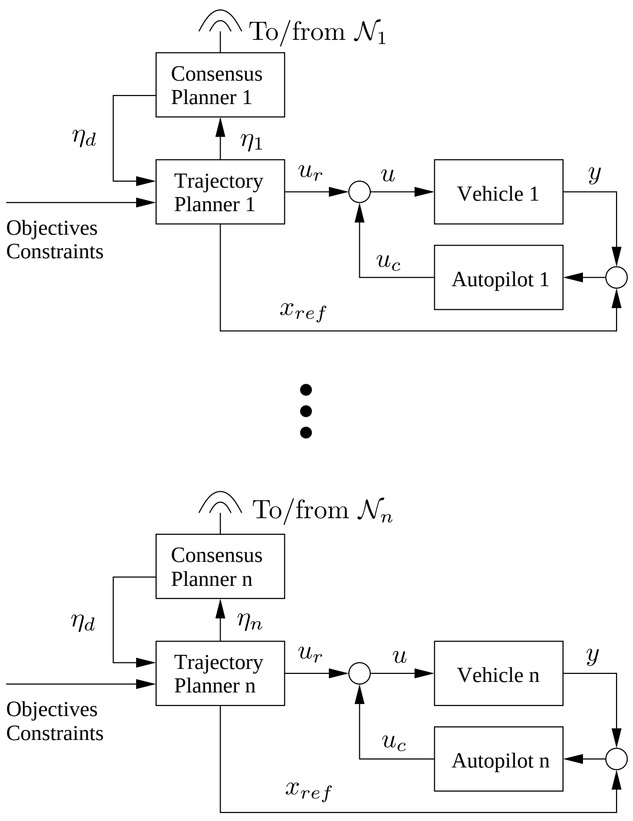


Figure 9. Block-diagram representing consensus planners and the information flow in the multi-vehicle network.

Estimating the Time to Arrival

As mentioned in Section III.V the terminal cost, $\Psi(p)$, conveys information about the distance from p to the target point, p_f. Our main objective for making this choice has been its computational simplicity. The consensus problem currently considered, reveals yet another advantage for making this choice; namely it facilitates arrival time estimation. Thus the simultaneous arrival problem currently considered serves as an example for a case when it might be beneficial not choosing the terminal cost, Ψ, close to the optimal value function, J^*.
Regarding the estimation of the arrival time based on the output of (3), the following is proposed:

$$\eta_{m,k} = T_p + \frac{\Psi(p_{k,N,m})}{v_{\max}}, \tag{10}$$

where $\eta_{m,k}$ denotes the ETA of the m^{th} vehicle at the k^{th} time-step, while $p_{k,N,m}$ is the final position of ditto. This choice is naturally not unique. In particular, one could choose the denominator in (10) differently, for instance as the average (or maximum) of the velocities in the planned trajectory, *i.e.*

$$\frac{1}{N}\sum_{i=1}^{N} v_{k,i,m}, \quad (\text{or } \max_i v_{k,i,m}).$$

The current choice (10) is however motivated by the fact that, if not updated in the far future, Ψ is an overestimation of the distance to go, why an apparent overestimation of the velocity in the denominator might be advantageous. In simulations, (10) has turned out to be a successful choice.

Consensus Value Incorporation

Next we consider the impact of the consensus planner on the trajectory optimizer, *i.e.* raise the question of how the trajectory planner should take η_d into account when planning the trajectory in the next time-step, $k+1$. In practice, the trajectory planner can satisfy the objective of the consensus planner by either imposing a hard constraint

$$T_p + \frac{\Psi(p_{k+1,N,m})}{v_{\max}} + h - \eta_d = 0, \tag{11}$$

or by penalizing deviations from the consensus value, η_d. For instance the following penalty term

$$\big[T_p + \frac{\Psi(p_{k+1,N,m})}{v_{\max}} + h - \eta_d\big]^q, \quad q \text{ positive even integer},$$

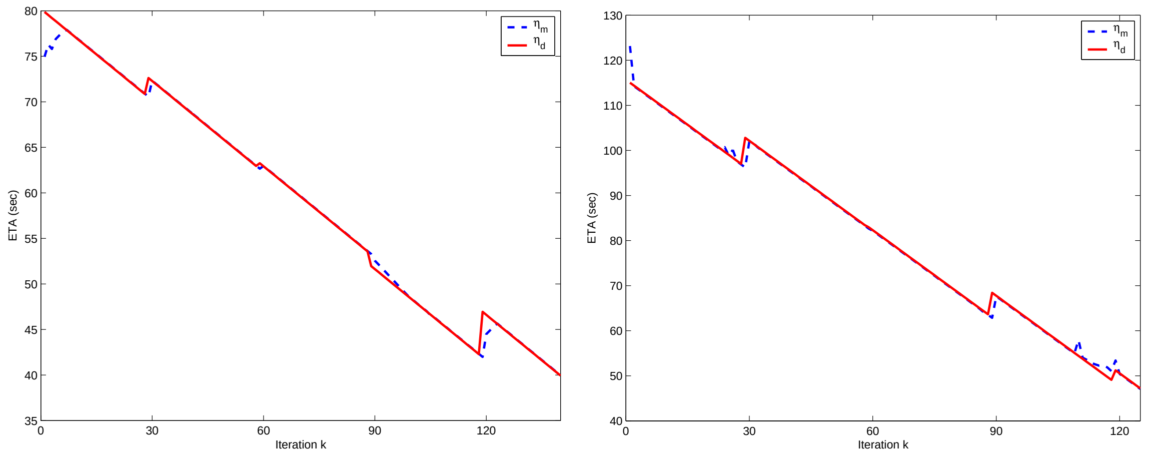
could be included in the objective function. In the simulations presented in Section D, the former approach has been taken. The main reason for that being its deterministic nature.
This is a convenient point at which to emphasize that since Ψ is merely an approximation of the distance to go (an overestimation if Ψ is not updated in the far future), one must expect and prepare for the case when the ETA turn out to be false at later time instances. To cope with this issue, as well as with the limited vehicle capability to adjust its ETA (in order to fulfill (11)), an "outlier detection rule" is adopted. The main idea is that if the consensus value is beyond the range which a vehicle can adjust its arrival time, it should classify itself as an "outlier" and as such, passively follow as the mission unfolds. Note however, that there is no need for an outlier to permanently resign from the mission as η_d might change non-uniformly in the far future. In addition, in order to keep the graph connected - but without affecting the consensus value - an outlier should communicate the average value of the consensus states of its neighbors as its own consensus state. How the outlier classification rule should be designed is an ambiguous question but, based on a scenario involving many vehicles with a few distinct outliers, the following rule

$$\frac{\eta_d - h}{\eta_{k,m}} \in [1-\delta, 1+\delta] \tag{12}$$

with $\delta = 0.3$ has been used in the simulations to follow.

D. Simulations

Since the convergence properties of the consensus value, η_d, can be handled separately, the principal objective for the simulations presented in this section, is to concentrate on the interaction between the consensus planner and the trajectory planner for one single vehicle. The following setup has been used to highlight this aspect and to incorporate the impact of the other vehicles.
In each iteration, a fictitious consensus planner receives η_m for the vehicle and sends back a modified consensus value, η_d. In the nominal case, in order to capture time-evolution, η_d will just decrease with a constant. Every 30th time-step however, η_d is set to random number that deviates from η_m by a given percentage at most. To connect to the outlier detection rule (12), a maximum deviation of 30% has been used in the simulations.
Although the simulations only involve the trajectory planner of one single vehicle, it must be emphasized that this setup enables us to capture several aspects of the impact of the other vehicles in the group as well. To start with, a deviating η_d may reflect the most ordinary case of different vehicles having different ETAs. Alternatively, this deviation may stem from the important case when the terminal cost, Ψ, turns out to be false for some of the vehicles. Finally, the occasionally occurring case when the optimization routine used in a vehicle fails to satisfy the hard equality constraint (11) may result in a deviating η_d. This latter case may occur even though η_d has passed the outlier detection rule (12) and has been observed in the simulations, as can be seen in Figure 10.
The simulations indicate that, in most cases, the trajectory planner manages to fulfill the will of the consensus



(a) (b)

Figure 10. Two simulations showing the ETA of a vehicle (dashed line) and the consensus value η_d (solid line) which incorporates the impact of the other vehicles in the group.

planner in one single time step. However, in some cases when there is a large change in the set point or when the optimization routine fail to satisfy the equality constraint (11) (most often since the limit for the maximum number of major iterations is reached), it might take more than one time-step for the trajectory planner to converge to the η_d dictated by the consensus planner.

IX. Conclusion

In this paper, results regarding trajectory optimization for aerial vehicles in three dimensional space has been presented. In particular, properties such as *safety*, *completion* and *simultaneous arrival* were in focus. Other prominent characteristics of the presented method, such as ability to minimize threat exposure and robustness, were highlighted through simulations.
As for the safety concerns, the alternative outlined in this paper, extends previous results by possessing provable safety properties in a 3D setting. In addition, in our case, safety also renders task completion possible. This is simply due to an elaborate choice of the set of safe states, in which the augmented safety maneuver is to end. As a subsidiary consequence, introducing the safety maneuver also makes it possible to cope with hard real-time systems.
Because of the computational burden it introduces, task completion is here not achieved by merely adjusting the length of the planning horizon. Instead, it is argued that requiring monotonicity of the composite cost is sufficient for approaching the target set. Decoupling the length of the planning horizon from our task completing objectives, enables us to determine it solely on the basis of accuracy demands and computational resources. To guide this selection, the interaction between optimality and computational load was empirically examined in Section VI, where a quantitative comparison between these two competing objectives where made.
Finally, by using a time-scale separation principle, we were able to adopt standard Laplacian control to a consensus problem which is neither unconstrained nor first order.

Acknowledgments

The first author would like to gratefully acknowledge the financial support by the Department of Autonomous Systems at the Swedish Defence Research Agency.

References

[1]Grancharova, A. and Johansen, T. A., "Explicit approaches to constrained model predictive control: a survey," *Modeling, Identification and Control*, Vol. 25, No. 3, 2004, pp. 131–157.

[2]Chen, Y. and Huang, J., "A new computational approach to solving a class of optimal control problems," *International Journal of Control*, Vol. 58, No. 6, 1993, pp. 1361–1383.

[3]Schierman, J., Hull, J., and Ward, D., "On-line trajectory command reshaping for reusable launch vehicles," *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exibit*, Aug. 2003.

[4]Canny, J. and Reif, J., "New lower bound techniques for robot motion planning problems," *Proc. of the* 28th *Annual IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, USA, Oct. 1987, pp. 49–60.

[5]Canny, J., Reif, J., Donald, B., and Xavier, P., "On the complexity of kinodynamic planning," *Proc. of the* 29th *Annual IEEE Symposium on Foundations of Computer Science*, Oct. 1988, pp. 306–316.

[6]Pirjanian, P., "Multiple objective behavior-based control," *Robotics and Autonomous Systems*, Vol. 31, No. 1–2, Apr. 2000, pp. 53–60.

[7]Chankong, V. and Haimes, Y. Y., *Multiobjective decision making*, Vol. 8 of *North-Holland Series in System Science and Engineering*, North-Holland Publishing Co., New York, 1983.

[8]Simon, H. A., *The new science of management decision*, The Ford Distinguished Lectures, Vol. III, Harper & Brothers Publishers, New York, 1960.

[9]Goodrich, M. A., Stirling, W. C., and Frost, R. L., "A theory of satisficing decisions and control," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 28, No. 6, Nov. 1998, pp. 763–779.

[10]Curtis, J. W. and Beard, R. W., "Satisficing: a new approach to constructive nonlinear control," *IEEE Trans. Automat. Control*, Vol. 49, No. 7, 2004, pp. 1090–1102.

[11]Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained model predictive control: stability and optimality," *Automatica*, Vol. 36, No. 6, 2000, pp. 789–814.

[12]Primbs, J. A. and Nevistić, V., "A new approach to stability analysis for constrained finite receding horizon control without end constraints," *IEEE Trans. Automat. Control*, Vol. 45, No. 8, Aug. 2000, pp. 1507–1512.

[13]Jadbabaie, A., Yu, J., and Hauser, J., "Unconstrained receding-horizon control of nonlinear systems," *IEEE Trans. Automat. Control*, Vol. 46, No. 5, May 2001, pp. 776–783.

[14]Primbs, J., Nevistic, V., and Doyle, J., "A receding horizon generalization of pointwise min-norm controllers," *IEEE Trans. Automat. Control*, Vol. 45, No. 5, May 2000, pp. 898–909.

[15]Sontag, E. D., "A "universal" construction of Artstein's theorem on nonlinear stabilization," *Systems & Control Letters*, Vol. 13, No. 2, 1989, pp. 117–123.

[16]Freeman, R. A. and Kokotovic, P. V., "Optimal nonlinear controllers for feedback linearizable systems," *Proc. of the IEEE American Control Conference*, Vol. 4, Seattle, WA, USA, Jun. 1995, pp. 2722–2726.

[17]Schouwenaars, T., How, J., and Feron, E., "Receding horizon path planning with implicit safety guarantees," *Proc. of the IEEE American Control Conference*, Vol. 6, 2004, pp. 5576–5581.

[18]Kuwata, Y. and How, J. P., "Stable trajectory design for highly constrained environments using receding horizon control," *Proc. of the IEEE American Control Conference*, Vol. 1, 2004, pp. 902–907.

[19]Bellingham, J., Kuwata, Y., and How, J. P., "Stable receding horizon trajectory control for complex environments," *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, Aug. 2003.

[20]Richards, A. and How, J. P., "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility," *Proc. of the IEEE American Control Conference*, Vol. 5, Jun. 2003, pp. 4034–4040.

[21]Richards, A. and How, J. P., "Robust variable horizon model predictive control for vehicle maneuvering," *Internat. J. Robust Nonlinear Control*, Vol. 16, No. 7, 2006, pp. 333–351.

[22]Mettler, B. and Bachelder, E., "Combining on- and offline optimization techniques for efficient autonomous vehicle's trajectory Planning," *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, USA, Aug. 2005.

[23]Ferguson, D. and Stentz, A., "The Ddlayed D* algorithm for efficient path replanning," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[24]Olfati-Saber, R., Fax, J. A., and Murray, R. M., "Consensus and cooperation in networked multi-agent systems," Proceedings of the IEEE, Invited Paper, Feb. 2006.

[25]Jadbabaie, A., Lin, J., and Morse, A. S., "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automat. Control*, Vol. 48, No. 6, 2003, pp. 988–1001.

[26]Fax, J. A. and Murray, R. M., "Information flow and cooperative control of vehicle formations," *IEEE Trans. Automat. Control*, Vol. 49, No. 9, 2004, pp. 1465–1476.

[27]McLain, T. W., Chandler, P. R., Rasmussen, S., and Pachter, M., "Cooperative control of UAV rendezvous," *Proc. of the IEEE American Control Conference*, Vol. 3, Arlington, VA, jun. 2001, pp. 2309–2314.

[28]McLain, T. W. and Beard, R. W., "Cooperative path planning for timing-critical missions," *Proc. of the IEEE American Control Conference*, Vol. 1, Jun. 2003, pp. 296–301.

[29]Davis, P. J. and Rabinowitz, P., *Methods of numerical integration*, Computer Science and Applied Mathematics, Academic Press Inc., Orlando, FL, 2nd ed., 1984.

[30]Lindström, P. and Pascucci, V., "Visualization of large terrains made easy," *Proc. of the IEEE Visualization, VIS '01*, San Diego, CA, USA, Oct. 2001, pp. 363–574, [Online] http://www.cc.gatech.edu/projects/large_models/ps.html.

[31]Anisi, D. A., Robinson, J. W., and Ögren, P., "Safe receding horizon control of an aerial vehicle," *Proc. of the* 45th *IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006.

[32]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M., "User's guide for NPSOL 5.0: a fortran package for nonlinear programming," Tech. rep., Systems Optimization Laboraty, Stanford University, Stanford, CA. 94 305, 1998.

[33]Kuwata, Y., Richards, A., Schouwenaars, T., and How, J., "Decentralized robust receding horizon control for multi-vehicle guidance," *Proc. of the IEEE American Control Conference*, Minneapolis, Minnesota, Jun. 2006.

[34]Dunbar, W. B. and Murray, R. M., "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica J. IFAC*, Vol. 42, No. 4, 2006, pp. 549–558.

[35]Olfati-Saber, R. and Murray, R. M., "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Control*, Vol. 49, No. 9, 2004, pp. 1520–1533.

[36]Moreau, L., "Stability of multiagent systems with time-dependent communication links," *IEEE Trans. Automat. Control*, Vol. 50, No. 2, 2005, pp. 169–182.

[37]Ren, W. and Beard, R. W., "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Automat. Control*, Vol. 50, No. 5, 2005, pp. 655–661.

[38]Fang, L. and Antsaklis, P. J., "Information consensus of asynchronous discrete-time multi-agent systems," *Proc. of the IEEE American Control Conference*, Vol. 3, Jun. 2005, pp. 1883–1888.