Proceedings of the American Control Conference
Arlington, VA June 25-27, 2001

Trajectory Generation for a UAV in Urban Terrain, using Nonlinear MPC [1]

Leena Singh — singhl@utrc.utc.com
James Fuller — fullerjw@utrc.utc.com
United Technologies Research Center
411 Silver Lane, East Hartford, CT 06118

Abstract

This paper describes a receding-horizon optimal control scheme for autonomous trajectory generation and flight control of an unmanned air vehicle in urban terrain. In such environments, the mission objective or terrain may be dynamic, and the vehicle may change dynamics mid-flight due to sensor or actuator failure; thus off-line pre-planned flight trajectories are limiting and insufficient. This technology is aimed at supporting guidance and control for future missions that will require vehicles with increased autonomy in dangerous situations and with tight maneuvering and operational capability e.g., missions in urban environments. A Model Predictive Control (MPC) scheme is described here that navigates a vehicle with nonlinear dynamics through a vector of known way-points to a goal, and manages constraints. In this MPC-based approach to trajectory planning with constraints, a feedforward nominal trajectory is used to convert the nonconvex, nonlinear optimal control problem into a time-varying linear, convex optimization or quadratic programming problem. The nonconvex, admissible path space is converted to a sequence of overlapping, convex spaces. The feedforward control that produces the nominal trajectory is found from the vehicle's differentially flat outputs. MPC is used to determine the optimal perturbations to the nominal control that will suitably navigate the vehicle through a constrained input/output space while minimizing actuation effort. Simulation results with a non-real-time, online MPC controller for a UAV in a planar urban terrain are included to support the proposed approach.

1 Introduction

Present vehicle control synthesis methods are predominantly based on control theories that assume linear, time-invariant (LTI) dynamics, unconstrained actuator authority, and unconstrained output responses. These assumptions do not correspond well to the typical flight control scenario where dynamics are nonlinear, actuators have limited authority and bandwidth, and physical and output constraints abound. Model Predictive Control (MPC), which is based on classical infinite-time optimal control theory, explicitly handles constraints and formulates an optimization index to compute a (locally) optimal sequence of controls over a finite length horizon in the future [1, 2]. Unfortunately, solving an optimization problem online - even a finite horizon optimization - is computationally burdensome; consequently, MPC has only been significantly used to control systems with large time-constants, such as process control applications in chemical engineering. Recently, however, due to ready access to faster, cheaper computers and efficient numerical algorithms for solving optimization problems, MPC has been popularly investigated as an alternative to contemporary control synthesis methods. [3, 4]

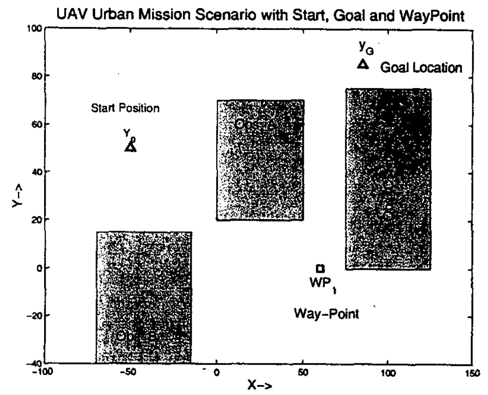


Figure 1: UAV navigation control problem in an urban environment: shows initial, & terminal positions and intermediate Way-Point.

This work uses MPC for planning trajectories between known set-points and around Cartesian obstacles; the objective is to use *a priori* path information about Cartesian way-points, supplied by a path planner, to compute the control that produces a (locally) optimal trajectory. The controller drives the vehicle to the final goal, y_G via the way-points which can be viewed as a series of tracking set-points. The obstacle locations impose state and output constraints. The problem scenario is displayed in Fig. 1 and discussed in later sections. Since the vehicle dynamics are nonlinear, we used perturbational analysis to linearize the dynamic model about a nominal (non-optimal) trajectory and thus formulate a convex optimization problem. This nominal, feedforward trajectory and its associated, admissible control are produced using classical spline-based trajectory planning and differential flatness methods. To complete the conversion of this finite-time optimal control problem to a tractable convex optimization, the non-convex feasible search space is converted to a series of overlapping convex regions. This is an important component of this work since we have developed an approach that enables us to derive the finite time optimal control in the highly constrained, nonconvex space of the optimization variables. We then solve a series of piecewise-convex constrained MPC problems as the vehicle moves forward in time.

The strategy was succesful in our planar scenario in Fig.1 where a 3-DOF vehicle must steer through a space with multiple obstacles from an initial location to a goal. The intermediate way-point provided by the path planner was such that a straight line path from one waypoint to another is infeasible as it cuts through obstacles. Our convex feasible set + MPC based trajectory controller accomodated this initial apparent fault from the path planner, without violating the other rate and acceleration constraints. The fault is realistic because one can easily expect offline path planners to produce paths with small errors due to an inaccurate terrain map, a changing urban scene due to other vehicles, or even sensor drifts that misalign true geographical coordinates from sensed ones.

Key components of our approach to trajectory control with state and input constraints using MPC that are presented here include: finding a linearized input-output vehicle model about a nominal trajectory and re-stating the performance index in terms of the perturbed state and outputs (Sec. 3), identifying the convex constraint regions (Sec. 4), and finding a feasible open-loop trajectory (Sec. 5). These subtopics are individually analyzed in the following sections and results obtained in simulation are presented in Sec. 6. First, however, we will present the vehicle model and pose the receding-horizon optimal control problem.

2 Formalization of the Optimal Control Problem and the UAV Dynamic Model

In this paper, we present a new rapidly reconfigurable navigation control algorithm for a UAV navigating in a highly collision-prone environment; the controller must avoid buildings or other vehicle constraints in the area, and must heed vehicle state and control constraints. We assume that an offline path planner has provided a series of waypoints to the goal; however, because the controller will be robust to uncertainties in the *a priori* information from which path plans were generated, the waypoints do not have to define feasible straight line paths in the environment. We also assume that a scene analysis software exists that localizes active threat obstacles as a function of the vehicle's position.

In this paper, the control problem is formulated as a discrete time, linear quadratic tracker with penalties on control effort and on tracking error relative to a reference trajectory. In receding horizon control this cost function only includes the penalty terms within a future time horizon of N time steps, $t_h = NT_s$ and not to $t = \infty$ as in LQR. (T_s is the sample rate of the control loop.) The problem then is to determine the sequence of actuator commands $w_k, k \in \{l, \dots (l+N-1)\}$, that minimizes the cost

$$J_l = \sum_{k=l}^{l+N-1} (y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) + w_k^T R w_k. \tag{1}$$

Only the first control element w_l, of this optimal control sequence is ever applied to the vehicle. The reference trajectory $\mathbf{r} = [r_0, r_1, r_2, \dots, r_M], M \geq N$ is obtained by simply interpolating from one set-point x_{m-1}, to the next waypoint location x_m at the vehicle's maximum allowable speed. This reference trajectory does not need to be admissible or analytic - it is only an artifact used to formulate a tracking error.

The forward control horizon, t_h, is chosen so that the inequality constraints beyond t_h do not significantly affect the control, w_l, at the present time. Presently, there are no formal ways to ensure that a horizon length is "long enough". Formal methods are being investigated in unconstrained MPC [5] that compute lower bounds on the horizon to ensure stability of the optimal solution. In this work, we used different horizon lengths, $t_h = NT_s$, and selected a 6 second interval in which the control was robust to changes in N.

A planar, Cartesian(X-Y), 3-DOF vehicle model is used to represent a simple, helicopter-like UAV. Vehicle dynamics are expressed in terms of its location (x, y), speed (v) and heading angle (α). The continuous time model of the vehicle in the plane is:

$$\dot{x}_1 = v\cos(\alpha) = x_3\cos(x_4) \tag{2}$$

$$\dot{x}_2 = v\sin(\alpha) = x_3\sin(x_4) \tag{3}$$

$$\dot{x}_3 = \dot{v} = w_1 \tag{4}$$

$$\dot{x}_4 = \dot{\alpha} = w_2 \tag{5}$$

$$y = [x_1, x_2]^T \tag{6}$$

Constraints on actuation commands of acceleration and heading turn rate and vehicle rates at every instant kT_s are:

$$(w_i)_{min} \leq w_i(k) \leq (w_i)_{max} \qquad i = \{1, 2\} \tag{7}$$

$$v_{min} \leq v(k) \leq v_{max} \tag{8}$$

The shaded geometrical shapes in Fig. 1 indicate city buildings which constrain UAV output states and which must also be transformed into inequality constraints to pose a constrained optimization problem. Due to the proximity and abundance of these output constraint surfaces in urban terrain navigation, the trajectory generator must explicitly and expediently handle constraints to prevent collisions. This capability allows the flight controller to safely and adaptively control the vehicle even if the offline terrain map is dated or otherwise imprecise, and is a key contribution of our research.

3 Transformation to a Convex, Quadratic Programming Problem

The optimal control problem is presently expressed as a time series in the input-output pair (w_k, y_k). It will be restated as an optimization problem in a time series of the input control vector w_k only, since the vehicles input-output dynamics are known and assure us that the input vector is the independent quantity and will drive the output. However, the nonlinear dynamic model poses a problem in this substitution since the quadratic cost component $(y_k - r_k)^T Q (y_k - r_k)$ becomes nonconvex when expressed in the input, and produces a nonconvex, nonlinear programming problem. Finding an optimum is, therefore, a difficult and computationally demanding task. A convex programming problem must be posed to expediently solve the optimization problem.

Linear perturbation methods were used to linearize the dynamic vehicle model about a nominal trajectory, $\mathbf{y}_0$ and form a convex, quadratic cost. We will defer discussion of the feasible trajectory and feedforward control sequence to Sec. 5. The next subsections briefly describe the perturbational approach to model linearization. Supporting details on our MPC approach is provided in [6].

3.1 Linearization of the Vehicle Model

Use perturbation techniques to expand the nonlinear model $\dot{x} = F(x, w), y = h(x, w)$ in a Taylor series about the nominal output and control trajectory $(\mathbf{y}_0, \mathbf{w}_0)$, to express the model as $\dot{x} = F(x_0, w_0) + \frac{\partial F}{\partial x}(x, w)|_{x_0, w_0}\delta x + \frac{\partial F}{\partial w}(x, w)|_{x_0, w_0}\delta w$, $y \sim h(x_0, w_0) + \frac{\partial h}{\partial x}(x, w)$. Retain the linear terms in state and input in the perturbational expansion.

$$\begin{aligned} \delta\dot{x} &= A(x_0, w_0)\delta x + B(x_0, w_0)\delta w \\ \eta &= C(x_0, w_0)\delta x + D(x_0, w_0)\delta w \end{aligned} \tag{9}$$

where $\delta\dot{x} = \dot{x} - F(x_0, w_0)$ and $\eta = y - h(x_0, w_0)$. Subsequently, the linear model is described in terms of the perturbed linear state $\delta x = x - x_0$ and perturbed linear control $\delta w = w - w_0$. We will refer to the perturbational controls $\delta w(t)$ as $u(t)$ in this work.

The discrete, linear, input-output mapping that will

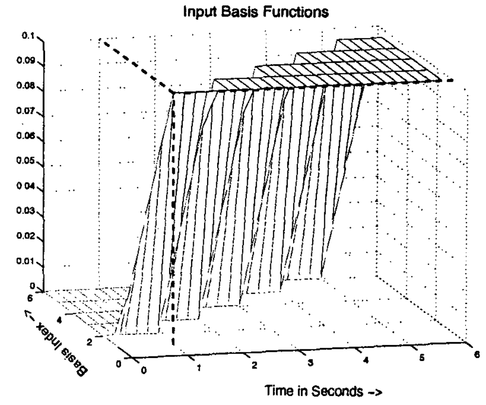


Figure 2: 3-D view of the Control Basis Functions, each basis is a ramp normalized to a maximum value

later transform the cost function is written as: $\eta_k = \sum_{j=0}^{k} h_{kj} u_j, \quad h_{kj} \in \Re^{m \times n}$ where matrix h_{kj} contains the input-output dynamics (A, B, C, D, T_s). In our 2-d vehicle model, u_j is a 2-vector input of acceleration and heading rate at time j; h_{kj} is a 2×2 matrix, at time jT_s, and T_s is the control loop rate.

Another computational simplification often used in MPC is to construct the input-output map using a set of *control basis functions.* Each input basis vector is N samples or t_h seconds long. In this analysis, we constructed the input basis set using the ramp functions shown in Fig. 2. The control is composed from a fixed number, s, of ***knots*** or deferred ramp functions. In our example, we used $T_s = 1/50s$, a $6s$ prediction horizon or control window and $1s$ ramps, with $s = 6$ knots. Any applied control is therefore constructed by linear superposition of the 6 knots or basis ramps. The control input represented in terms of its bases is:

$$\begin{array}{ccc} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+n_1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} & = & \begin{bmatrix} du & 0 & 0 & \dots & 0 \\ 2du & 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ p & du & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ p & p & p & \dots & p \end{bmatrix} \cdot \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \vdots \\ \nu_6 \end{bmatrix} \\ \mathbf{U} & = & S \quad\quad . \quad V \end{array}$$

where n_1 is the length of each ramp. With our bases of $1s$ long ramps, $V = [\nu_1, \dots \nu_6]$ is the vector of scale factors on the 6 knots, they multiply each basis function in S to produce the 6sec long perturbational input sequence, $\mathbf{U}$. In the notation above, each u_i, and ν_j is a column vector of length 2 since our vehicle system has 2-inputs; correspondingly, each u_k and du are 2-vectors in 3-space with entries $du_{\cdot} = \binom{du_{\cdot,1}}{du_{\cdot,2}}$, and $\nu_i = \binom{\nu_{i,1}}{\nu_{i,2}}$ and where $du_{\cdot,j} = pj/n_1$ as in Fig. 2 where $p = 0.1$. Note that each input basis vector of the j^{th} input $(j = \{1, 2\})$ is explicitly contained in the columns of S. $\mathbf{U} = SV$ only pertains to the *linear perturbational component of the input* $\mathbf{w}$; the actual inputs applied will include the feedforward, nominal control trajectory, $\mathbf{w}_0$ so that the control applied at time iT_s will be $\mathbf{w}(i) = \mathbf{w}_0(i) + S(i,:)V$.

We can now model the perturbational response η_k at kT_s as the output of a linear system:

$$\eta_k = \sum_{j=l}^{k} h_{kj}u_j = \sum_{j=l}^{k} h_{kj} \sum_i S_{ji}\nu_i = \sum_i g_{ki}\nu_i \tag{10}$$

$$\implies \quad \mathbf{y} = \mathbf{y}_0 + \boldsymbol{\eta} = \mathbf{y}_0 + GV, \quad G \in \Re^{2N\times 6} \tag{11}$$

The tensor G in Eq. 11 contains the response of the vehicle model to the input sequence in the control basis functions. In practise, G is constructed from the perturbational system response trajectory to the i^{th} input basis $S_{:i}$:

$$G_{:,i} = \eta^i = H(\mathbf{w}_0 + S_{:i}) - H(\mathbf{w}_0) \tag{12}$$

where $H(\mathbf{w}_0)$ is the nonlinear model's exact response to the first N elements of the nominal input sequence, and $H(\mathbf{w}_0 + S_{:i})$, the exact model response to the (first N) nominal + i^{th} perturbational input basis vector. The i^{th} column of G is, accordingly, the η^i output trajectory due to the i^{th} basis vector. The superposition principle allows us to shape output η by appropriately scaling the input bases, (and therefore, the linear perturbational input-output mapping GV); each "input", ν_i, is appropriated from the 12 scale factors in V that forms the perturbational control $U = SV$.

3.2 Performance Index Characterization

The performance index used in this work weights tracking error and control effort. Since the control and actual output trajectories contain nominal (non-optimal) feedforward components, $J = \sum_{i=0}^{N-1} (\mathbf{y}_i - \mathbf{r}_i)^T Q (\mathbf{y}_i - \mathbf{r}_i) + \mathbf{w}_i^T R \mathbf{w}_i$ will be restated in terms of (u_k, η_k), instead of the pair (w_k, y_k).

$$J_l = \sum_{i=l}^{l+N-1} (\mathbf{y_0}_i + \boldsymbol{\eta}_i - \mathbf{r}_i)^T Q (\mathbf{y_0}_i + \boldsymbol{\eta}_i - \mathbf{r}_i) + (\mathbf{w_0}_i + \mathbf{u}_i)^T R (\mathbf{w_0}_i + \mathbf{u}_i) \tag{13}$$

$$= (\mathbf{y}_0 - \mathbf{r})^T \mathbf{Q} (\mathbf{y}_0 - \mathbf{r}) + \mathbf{w}_0^T \mathbf{R} \mathbf{w}_0 + \sum_{i=0}^{N-1} \boldsymbol{\eta}_i^T Q \boldsymbol{\eta}_i + \mathbf{u}_i^T R \mathbf{u}_i + 2\left((\mathbf{y_0}_i - \mathbf{r}_i)^T Q \boldsymbol{\eta}_i + \mathbf{w_0}_i^T R \mathbf{u}_i\right). \tag{14}$$

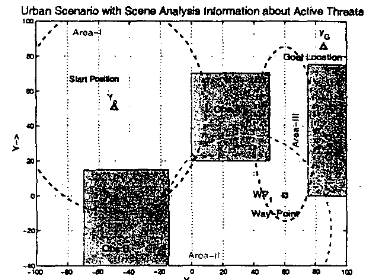
where y_{0i} is the nominal response at iT_s. The terms before the summation are independent of the optimization parameters and can therefore be dropped without penalty. After substituting $\mathbf{U} = SV$ and $\eta = GV$, the optimization cost index in the optimal control problem becomes the convex function: $J_l = V^T(G^T\mathbf{Q}G)V + V^T(S^T\mathbf{R}S)V + 2((\mathbf{y}_0 - \mathbf{r})_i^T \mathbf{Q}GV + \mathbf{w}_0\mathbf{R}SV)$ and the optimization problem becomes one of minimizing J_l by choosing the 12 scales of V. We used MATLAB's quadratic programming solver `quadprog()` to solve for V over the prediction horizon. Once V is available, the first control $u_1 = S_{1:} \cdot V$ is then applied to the vehicle.

4 Forming the Convex Constraint Solution Spaces

An optimization problem is convex if both the performance index and the space over which the optimization is performed, are individually convex. A reliable numerical implementation of optimization in real-time requires a convex search space. We define a *feasible solution space* to be the admissible part of the solution space: i.e. one that meets constraints on vehicle speed, acceleration and turn rate limits, and an output space that does not intersect obstacles. The input constraints considered in this work naturally define a convex feasible space. The set of feasible points in the Cartesian output space is almost always non-convex in an urban terrain scenario. Therefore, the nonconvex, feasible output space is reduced to one or more overlapping, convex subsets. We call these subsets the *convex, feasible space* or CFS; the optimization solver will only search one convex, feasible subset of the complete (nonconvex) feasible space at a time, for the optimal MPC solution. The method used to produce the local convex overlapping sets is an important component of this work to onboard control synthesis in highly constrained geometries, using MPC.

Our approach to forming the convex feasible space requires some scene interpretation from a high level scene-analysis software. Given the terrain shown in Fig. 1, one can see that when the vehicle is at $y(0)$, obstacle C is not a collision threat. Consequently, we presume that the scene-analysis software coarsely demarkates areas in the terrain map, labelled Areas I, II, III, IV, in Fig. 3, which qualify relevant obstacles that can "see" the vehicle, when it is in a particular area $(I, \dots IV)$ of the urban map. This subdivided scene is the starting point in our convex feasible space generation method.

Convex feasible subsets (CFS) are formed from vectors or rays from the vehicle location to each vertex of the active obstacles in that particular area, such that the vertices can "see" both the present vehicle position and the next way-point. In Fig. 4 where the vehicle is in Area-I, Obs_A and Obs_B are the active obstacles. A vertex is "visible" from a point if the vector between them is not occluded by any surface. With this method, we arrive at two overlapping convex polygonal regions: one polygon, (OAB), is produced by rays from the vehicle position to the commonly "visible" (from both vehicle and next way-point) vertices on the



Urban Scenario with Scene Analysis Information about Active Threats
Area-I
Start Position
Goal Location
Area-III
Area-II
WayPoint
X->
Y->

Figure 3: Urban Terrain after a scene analysis routine has identified relevant threats by geographical location

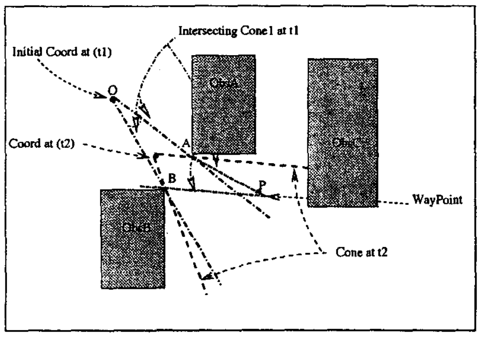


Figure 4: Convex Feasible Subset generation using intersecting cones from vehicle location, O, to appropriate "visible" obstacle vertices A, B.

active obstacles, and the other, polygon (ABP), from those "visible" vertices to the next way-point. We ignore the second convex set, ABP until the vehicle has entered the overlapping region at the intersection of the two polygons, and then activate and only consider constraints defined by the second convex space. Clearly, the constraint sets thus produced are only a subset of the true feasible state space.

This method has limitations when the vehicle is far from the obstacles. The constraint surfaces produced by this approach can be overly conservative when the angle subtended at the vehicle by the polygonal bounding vectors becomes very small; the enclosed area then defines an unnatural "feasible" (but convex!) space. We have developed another heuristic to grow this feasible region along the edges of the obstacle but will not develop it here for lack of space.

Finally, the vehicle must segue from one convex feasible set to the next to eventually reach the intended subgoal. For reasons that will be developed in Sec. 5 we insert a new *approximate waypoint* at the center of the intersection area between the two convex polygons **OAB** and **ABP**.

5 Feasible Nominal Trajectory and Control

The nominal trajectory used to linearize the vehicle model and thus produce a convex performance index must be exact and feasible i.e. it should be an exact solution to the nonlinear vehicle model in Eqs. 2-6 (in response to some control sequence $\mathbf{w}_0$) and satisfy all state constraints. Also, the corresponding nominal control must be admissible and satisfy control constraints. This nominal is only generated when the vehicle first enters a new convex feasible space. Subsequently, the nominal is continually "improved" by incorporating the optimal control sequence $U = SV$ into the "previous" nominal control vector.

In this work, we use polynomial interpolation to form the initial nominal trajectory in the output space to connect one way-point, y_m, to the next, independent of vehicle dynamics. Vehicle coordinates, $y(t)$ in the interval $\{t = 0, T_m\}$ are expressed as a polynomial series in some n-dimensional basis, $\{\phi_j\}$, so that:

$$y_i(t) = \sum_j \alpha_{ij}\phi_j(t), \quad i = \{1,2\}; \quad y_i, \phi_j \in \Re^2, \alpha_{ij} \in \Re^1 \tag{15}$$

where the basis functions $\phi_j(t), j = 0 \dots (n-1)$ are differentiable, the output is $y = [x_1, x_2]^T$ in our planar example, and the multipliers α_{ij} completely parameterize the output trajectory. In this work, we define a polynomial bases in the time series as $\phi_j(t) = t^j, j = 0 \dots 3$.

α_{ij} are computed using state requirements at present time, and at the next m^{th} way-point, as boundary conditions to ensure analyticity of the resulting output trajectory. The output trajectory in the interval $t \in [t_0, t_m]$ is thus constrained by $y_0, y_{t_m}, \dot{y}_0, \dot{y}_{t_m}$. Once the α_{ij} are found, $y, \dot{y}, \ddot{y}$ can be completely specified at any time t_k, $t_k \in [t_0, t_m]$, by $y_i^{(m)} = \alpha_{ij}\phi_j^{(m)}$. At each way-point, the trajectory is similarly extended to the next way-point in a polynomial continuation. The resulting trajectory is referred to as the nominal trajectory, $\mathbf{y}_0$ and is used to linearize the nonlinear aircraft model. The nominal trajectory used in Region III is shown in Fig. 5.

If any point in this open loop trajectory falls outside the convex feasible space, a new way-point is inserted in the middle of the feasible set, the spline equations recomputed and rechecked to make sure that the new trajectory is feasible. We require that each convex polygon contains a subgoal or way-point so that a nominal trajectory can be found that lies entirely within the CFS.

5.1 Nominal Control Using Differential Flatness

Given the feasible, nominal trajectory $\mathbf{y}_0$ we can find the nominal control sequence that produces it. In this paper, we exploit the differential flatness of the vehicle

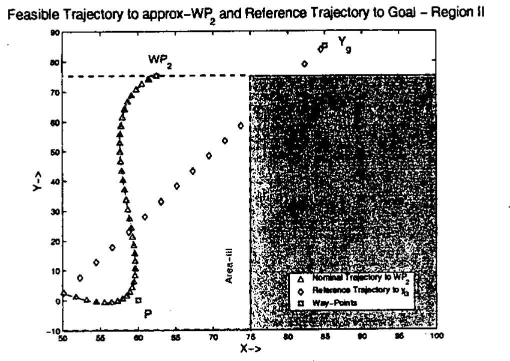


Figure 5: Nominal Trajectory produced in Convex Region III in Fig.2. Algorithm automatically inserts the approximate waypoint P_2 so that the nominal trajectory is admissible in the constrained set.

dynamics to produce the nominal, feedforward control. Differentially flat systems are those for which there is a one-to-one correspondence between the trajectories of a set of *flat outputs*, and those of the full state space and input. These methods have been extensively used in trajectory generation applications because a feasible trajectory can be planned in the output space and "lifted" to the state and input space, through an algebraic mapping to determine the feed-forward control. The method was originated by Fliess et. al. [7]. See [8] for a survey of this method, in particular, its use in real-time trajectory generation. In essence, if we can find the differentially flat outputs $z \in \Re^m$ of the form $z = \xi(y, w, \dot{w}, \ddot{w}, \ldots, w^{(l)})$ such that trajectories, $\mathbf{x}$ and $\mathbf{w}$ can be expressed in differential orders of z, we will have the pointwise algebraic method for transforming the planned feasible trajectory $\mathbf{y}_0$, into the control trajectory, $\mathbf{w}_0$.

$$
\begin{aligned}
x &= X(z, \dot{z}, \ldots, z^{(l)}) \\
w &= U(z, \dot{z}, \ldots, z^{(l)})
\end{aligned}
\tag{16}
$$

The vehicle model used in this work is differentially flat with the flat outputs $z = [x_1, x_2]$, identical to its *tracking output*, y. As a result, it was intuitively easy to construct the nominal flat trajectory in the physical output space. Since the nominal trajectory $\mathbf{y}_0$ is formed using spline bases that are explicit functions of time, its derivatives are easily computed from the appropriate basis parameters at any time. To derive closed form expressions for the state and control in terms of $z(t), \dot{z}(t), \ddot{z}(t)$, differentiate the nonlinear vehicle dynamics in Eq. 2-6 and substitute for z to obtain:

$$x_3(i) = \sqrt{\dot{z}_1^2 + \dot{z}_2^2}(i) \tag{17}$$

$$x_4(i) = \tan^{-1}\left(\frac{\dot{z}_2}{\dot{z}_1}\right)(i) \tag{18}$$

$$\begin{bmatrix} \ddot{z}_1(i) \\ \ddot{z}_2(i) \end{bmatrix} = \begin{bmatrix} \cos(x_4) & -\dot{z}_2 \\ \sin(x_4) & \dot{z}_1 \end{bmatrix}_i \begin{bmatrix} w_1(i) \\ w_2(i) \end{bmatrix} \tag{19}$$

The sequence of $w(i)$ in Eq. 16, $\mathbf{w}_0(i)$, is lifted from the flat output trajectory and its derivatives at each instant, by solving the (inverse) algebraic equations for $w(t)$. This is the associated nominal control trajectory. The nominal control associated with the nominal trajectory in Fig. 5 is shown in Figs. 6 and 7.

6 Simulation Results

A control algorithm for steering a vehicle through the urban terrain described in Fig. 1 was developed using MPC. The nonlinear vehicle model is in Eqs.2-6. The simulation initial conditions are $y(0) = [-50, 50], v(0) = 2.5m/s, \alpha(0) = +25\,\text{deg}$. The goal location is: $y_G = [65, 85]$, and the known way-point is at: $[60, 0]$. Actuation and state constraints are:

$$
\begin{array}{rcccl}
0 & \leq & v & \leq & 8.0m/s \\
-15m/s^2 & \leq & \dot{v} = u_1 & \leq & 12m/s^2 \\
-60\,\text{deg}/s & \leq & \dot{\alpha} = u_2 & \leq & 60\,\text{deg}/s
\end{array}
\tag{20}
$$

Output constraints are intended to keep the UAV from hitting buildings in the terrain map. The way-point is an intermediate tracking set-point to reach y_G. The Q and R matrices in the cost function weight tracking error 4 times greater than control effort.

We fragmented the problem into the four zones, I-IV as mentioned in Sec. 4, formed the convex polygonal feasible spaces, ensured that there is a subgoal contained within each convex region, formed the tracking reference trajectory $\mathbf{r}(t)$, and posed the appropriate input and output constraints. The subgoal in each feasible, convex space may either be the final goal, a path-planner supplied way-point, or an approximate way-point obtained when forming the feasible output spaces. The algorithm placed two subgoals at the intersection of CFS I and II, at $[-5, 15]$ and at $[65, 85]$ at the intersection line of CFS III and IV. Recall that the reference trajectory did not need to be feasible.

In each output space, we used the vehicle location to limit the constraint set to a convex polygon that uses the visible edges of active obstacles, and that will overlap with a convex, feasible space in the next region near a way-point. At present, the constraint forming software is automatic but not dynamic - we generate one convex polygonal set in an area.

The way-points are used when the vehicle enters a new feasible space to produce the nominal trajectory in

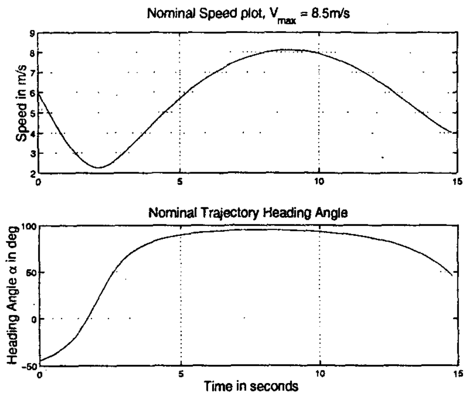


Figure 6: Nominal Velocity and Heading Angle Trajectories, in Feasible Space III

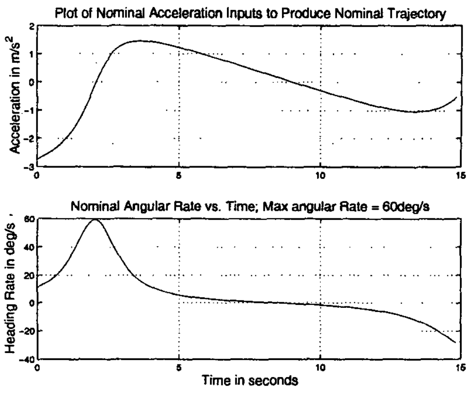


Figure 7: Nominal Vehicle Acceleration and Angular Rate Control, in Space III

the flat output space using the polynomial equation: $z(t) = \alpha_0+\alpha_1 t+\alpha_2 t^2+\alpha_3 t^3$. We chose a terminal velocity that ensures analytic continuation of the trajectory at each way-point, by orienting the terminal velocity towards the next, upcoming way-point at half V_{max}. Thus, the nominal trajectory was designed to have a final orientation towards y_G when it reached approximate wayPoint $P_2 = [65, 85]$. The associated nominal vehicle speed approaches and flattens at a maximum of $8m/s$, and heading control rates at $\dot{\alpha}|_{max} = 60\deg/s$.

As the vehicle moves within the polygonal set towards the way-point, it enters the next polygonal set; when this happens, the active constraints change so that they are defined by the bounding surfaces of the new convex set. Likewise, the subgoal also changes - the next way-point becomes the new goal. We compute the new nominal trajectory and the attendant control sequence $(\mathbf{y}_0, \mathbf{w}_0)$, and the new reference tracking trajectory, $\mathbf{r}$ if WP_i is a true way-point. The G tensor contains the linear perturbational input-output mapping and is recomputed at each iteration using Eqs. 12.

The optimization proceeds as before using MATLAB's quadprog() routine. Quadprog() takes the following arguments: the convex term $G^T QG + S^T RS$, the linear term $(\mathbf{Y}_0 - \mathbf{r})^T QG + \mathbf{w}_0^T RS$, and state and input constraints and produces the vector of input scale factors, V. Recall that the 12 parameters of V are the optimization parameters needed to produce the linear perturbational control in a $6s$ window from the input sequence $U = SV$. However, the actual applied control is only the first term $\mathbf{u}_l = \mathbf{w}_0(l) + S_{1:}V$.

The control constraints on V are w_0: $w_{min} \leq w_0(i) + S_{:i}V \leq w_{max}$. However, since quadprog() only takes upperlimits, we restate this as: $\left(\begin{smallmatrix} S \\ -S \end{smallmatrix}\right)\cdot V \leq \left(\begin{smallmatrix} W_{max}-W_0 \\ W_0-W_{min} \end{smallmatrix}\right)$. State constraints are similarly formed in terms of the nominal trajectory $\mathbf{y}_0$ and parameterized in V.

The nominal input and output vector are constantly updated with the optimal perturbations. At the 1st iteration, when an optimal perturbational input sequence is generated, the N sample control contains the feed-forward control sequence modified by the perturbational control within the prediction horizon: $\mathbf{w}'(i = 1 : N) = \mathbf{w}_0(i = 1 : N) + SV$, (Only $w'(i = 1)$ is applied). We perturb our definition of the nominal control and call the N-sample vector, $\mathbf{w}'$, our new nominal control. When the window slides forward at the next iteration, we correspondingly slide this "perturbed nominal" control forward: $\mathbf{w}'(2 : N)$; and to keep the window size intact, we append the $(N + 1)^{th}$ control element from the *original* w_0 to the end of this new nominal control: $[w_0'(2 : N), w_0(N + 1)]$.

The method produced the output trajectory shown in Fig. 8 proceeding through the four regions to arrive at the goal. In the first convex feasible space, (CFS1), the trajectory follows the reference quite closely until the obstacle constraints creep into view in the prediction horizon; it then starts pulling away from the reference and into the next CFS that contains the true waypoint, WP_1. Speed and turn rate commands are shown in figure Figs. 9, 10. Speed is limited to $8m/s$ and since the tracking reference trajectory was designed with this speed, the controller pushes the vehicle at maximum speed to track the reference whenever possible. One can envision a situation where similar limits on actuator authority are approached and obeyed. In these simulations, we see that controller ensures that constraints on speed and Cartesian position are met.

7 Conclusions

In this paper, we presented an MPC based method for online flight navigation and control in an urban environment. The approach is targeted at future guidance and control applications where constraints are more stringent, the need for reconfigurable control becomes immediate and where variable mission-dependent performance considerations present a challenge to current flight control design and software methodologies. We presented an approach to navigate a nonlinear, planar vehicle in an urban environment with tight output, state and control constraints using onboard control synthesis using Model Predictive Control. The proposed approach worked well in that it synthesized in real-time, an aggressive trajectory that satisfied constraints and steered the vehicle around the obstacles to a desired goal location. The trajectory is composed from a path supplied by a path planner; the plan is infeasible due to uncertainty in the terrain maps used by the offline planner. Our optimization based control algorithm accomodated these initial faults and designed a robust trajectory that satisfied output (obstacle) and control constraints. We presented a method for transforming the initial non-convex optimization by forming a series of propagating convex optimization problems that are convex at any instant and vehicle location. This work will be further extended by expanding the model and the simulation to a 3-D scenario and including a more realistic UAV model.

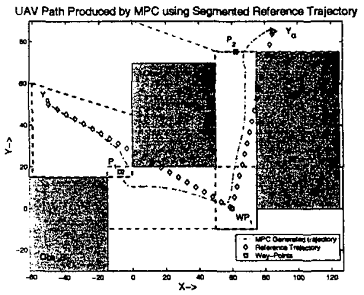


Figure 8: Recorded UAV Output Trajectory, using online MPC with Overlapping Feasible Convex Regions Used to Generate Optimal Solutions

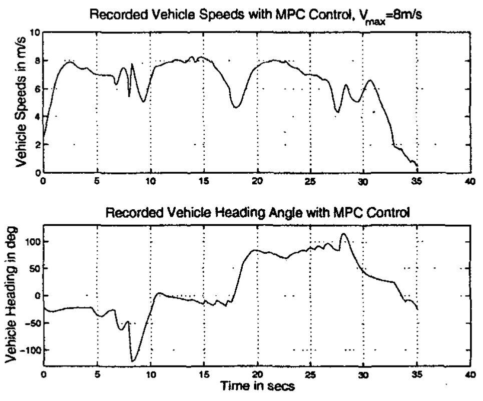


Figure 9: Recorded UAV Velocity and Heading Angle using online MPC and 6s Prediction Horizons

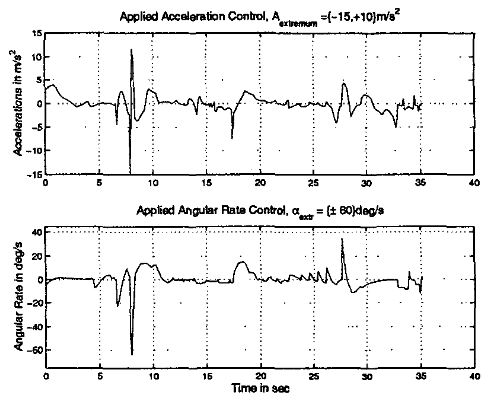


Figure 10: Recorded UAV cl lp. MPC Controls in 6s Prediction Horizon

References

[1] C.E.Garcia, D.M.Prett, and M.Morari, "Model predictive control: Theory and practice - a survey," in *Automatica*, vol. 25, 1989.

[2] D.Q.Mayne, J.B.Rawlings, C.V.Rao, and P.O.M.Scokaert, "Constrained model predictive control: Stability and optimality," in *Automatica*, vol. 36, pp. 789–814, 2000.

[3] S.Boyd, C.Crusius, and A. Hansson, "Control applications of nonlinear convex programming," in *Journal of Process Control*, vol. 8, pp. 313–324, 1998.

[4] S.J.Wright, *Primal-Dual Interior-Point Methods.* Philadelphia: SIAM, 1997.

[5] J.Primbs and V. Nevistic, "Feasibility and stability of constrained finite receding horizon control," in *Automatica*, no. 36, pp. 965–971, 2000.

[6] J.Fuller, D. Seto, and R. Meisner, "Optimization-based control for flight vehicles," in *AIAA Guidance Navigation and Control Conference*, June 2000.

[7] M.Fliess, J.Levine, Ph.Martin, and P.Rouchon, "Sur les systemes non lineaires differentiellement plats," in *C.R. Acad. Sci. Paris, Serie I*, pp. 619–624, 1992.

[8] M. Nieuwstadt and R.M.Murray, "Real time trajectory generation for differentially flat systems," in *Journal of Robust and Nonlinear Control*, no. 11, pp. 995–1020, 1998.