Path Planning of Unmanned Aerial Vehicles in a Dynamic Environment

Jeong-Won Lee ¹, Bruce Walker² and Kelly Cohen ³

School of Aerospace Systems, University of Cincinnati, Cincinnati, OH, 45221

The main goal of this research effort is to determine the optimal trajectory for an unmanned aerial vehicle (UAV) in a dynamic environment. A Model Predictive Control (MPC) approach is utilized to provide collision avoidance in view of pop-up threats and a random set of moving and stationary obstacles (no fly zones). The UAV path planning needs to adapt in near real-time to the dynamic nature of the operational scenario, and to react rapidly to updates in the situational awareness, given the vehicle’s maneuvering constraints. To achieve this objective, the UAV navigates from a given starting point to a desired target point via selected intermediate waypoints. The possible waypoints are geometrically obtained with additional waypoints placed near the vertices of each polygon-shaped obstacle. The MPC optimizer minimizes a cost function at each control cycle using a nonlinear dynamic model of the situation with maneuvering constraints included. The MPC algorithm is selected because it can improve system performance while effectively handling constraints. However, the huge computational effort required for a complete nonlinear realization of the MPC algorithm renders infeasible a comprehensive real-time optimization for this application. To circumvent this problem, discrete-time Laguerre functions are used as basis functions to represent the control inputs instead of using their complete time histories. A representative scenario, consisting of five targets, five static obstacles, nine pop-up threats, and a large, moving no-fly zone, is used to demonstrate this algorithm. Results are presented based on MATLAB® simulation experiments using a UAV model that represents a RQ-7 Shadow 200 to demonstrate the effectiveness of this approach.

I. Introduction

FLYING a piloted aircraft in a hazardous environment, such as a battle zone or severe weather, is a risky venture, because the life of the pilot is at stake. The merits of unmanned aerial vehicles (UAV’s) are that they can accomplish many missions at relatively low cost, they are especially effective in long endurance surveillance missions [1], and they do not put human pilots in danger. UAVs are widely used today in military operations, and they have considerable potential for civilian applications as well [2].

UAVs need to avoid obstacles and pop-up threats as they maneuver through dynamic environments. To enhance its mission effectiveness, a fully autonomous UAV must follow an optimal, or nearly optimal, trajectory in terms of some cost function that reflects its effectiveness, such as total fuel use or total mission time. A primary challenge associated with UAV path planning is in dealing with the uncertainty associated with a dynamic environment and the implementation of autonomous decision-making within a given time frame. Many researchers have introduced various path planning algorithms for UAV’s, such as Voronoi diagrams [3-6], A* algorithms [7-9], genetic algorithms (GA) [10-16], or virtual potential field (VPF) methods [17-19]. A Voronoi diagram is constructed based on the range of known threats, e.g., radar exposure. Each edge of the diagram is based on the threat cost and the fuel cost. Then, a feasible flying path is designed from coarse paths, i.e. smooth paths replace sharp turns. The shortcoming of using Voronoi diagrams is the “curse of dimensionality,” i.e., the computation time for the algorithm is increased substantially as the problem moves from 2-dimensional to 3-dimensional.

¹Graduate student, School of Aerospace Systems, University of Cincinnati, AIAA Student Member
²Associate Professor, School of Aerospace Systems, University of Cincinnati, AIAA Associate Fellow
³Associate Professor, School of Aerospace Systems, University of Cincinnati, AIAA Associate Fellow

American Institute of Aeronautics and Astronautics
The A* algorithm is a very popular graph-based search algorithm based on both on-line and off-line components. It basically determines the shortest trajectory from an initial point to the desired end point using a heuristic approach. In the off-line component, motion segments are built and stored in a library, and then segments are sequenced to determine the optimized trajectory on-line [7]. The drawback of this approach is the limitation of using pre-estimated flight segments in an unknown environment. The Online Sparse A* (OSAS) algorithm [8] searches for a near-optimal trajectory, which consists of straight line segments, using onboard sensors to detect the unknown environment. Unfortunately, the near-optimal trajectory is often not a feasible path due to the non-holonomic constraints of the vehicle. In order to get a feasible path, the computation time increases substantially. Howlett, et al [9] consider the sensor footprint and the UAV do not require passing directly over the targets. Therefore, using the Learning Real-Time A* (LRTA*) algorithm, the shortest path can be obtained because the sensor footprint covers the targets. However, the aircraft dynamics used in this application were assumed to be linear and the initialization of the heuristics took a long time.

GA is an adaptive heuristic search algorithm based on Charles Darwin’s theory of evolution. GA uses historical information and the principle of “survival of the fittest” to yield ever-higher performing potential solutions among the population of potential solutions. The advantages of GA are its applicability to nonlinear systems and adaptability to uncertain environments. During the evolution of the algorithm, better performing solution sets, which are evaluated using a given fitness function, are retained by natural selection and weaker ones are eliminated. In [10-13, 15], a feasible path was obtained within the generated population. In [14, 16], waypoints were first obtained using GA, and then the best trajectory for the UAV was established by connecting waypoints. Even though GA handles nonlinear systems well, the computation time increases as more detailed paths are required.

The virtual potential field (VPF) method is founded on the principle that when a mass approaches an obstacle, a repulsive field exerts force to push the mass away from obstacle. And as the mass nears a target position, an attractive force increases and the mass approaches the target. The VPF method is used for avoiding collision in 2D scenarios [17] and in 3D scenarios [18, 19]. However, the case of a dynamic environment has not been considered. This approach is known to yield a local minimum in the field, especially when dealing with geometrically convex obstacles. Therefore, before approaching the target position, the UAV may become trapped at one of these local minima [17].

The Model Predictive Control (MPC) algorithm improves the performance of systems while handling constraints effectively. The MPC algorithm was originally developed for controlling power plants and oil refineries and has been used extensively over the last three decades [20]. Qin et al [21] states that MPC is widely used in various areas such as automotive, chemicals, robotics, and aerospace applications. Recently, MPC has been applied to generating collision-free trajectories. However, the shortcoming of using MPC in a dynamic environment is its intensive need for real-time computation due to on-line optimization at each control cycle. References [22-25] use basis functions with MPC to reduce the heavy computation load. The control can be rewritten as linear combinations of a set of basis functions weighted by scale factors. The optimizer solves for the optimized scale factors instead of for the full control histories over the control horizon. Ramp [22] and tent [23] functions may also be introduced. Singh and Lapp [24] introduce Laguerre and Legendre polynomials as basis functions. Wang [25] gives the Laguerre functions in a discrete time form. It should be noted that the obstacles considered in the references [22, 23] are static.

This research effort focuses on the development of an efficient UAV trajectory planning algorithm that has the ability to effectively adapt to dynamic environments. The approach uses the MPC algorithm and discrete-time forms of Laguerre functions as basis functions. The optimizer finds the optimal coefficients for a combination of Laguerre functions that replaces the full optimization of the input time histories to obtain computationally feasible solutions for real-time applications. Scenarios for testing this algorithm include moving obstacles and pop-up threats in a two-dimensional environment. The formulation may later be extended to a three-dimensional scenario.

II. Problem Formulation

We assume that the mission is for the UAV should to fly to a designated target point. There are three considerations in performing this task: 1) the time histories of the control inputs must be determined subject to some constraints on the maneuverability of the vehicle; 2) the UAV must avoid collisions and threats in a dynamic environment; and 3) a specified cost function should be minimized.

The UAV is operating in a dynamic environment and when an unexpected obstacle or a pop-up threat appears, such as an enemy air defense system. The UAV should modify its path accordingly to avoid any obstacles or threats and approach the target based on the generation of a modified trajectory computed within the allocated time window. In order to achieve this goal, the operating space for the UAV mission is specified as potential waypoints.
with additional waypoints added near obstacles and threats, and the Model Predictive Control (MPC) is used to optimize the trajectory effectively with consideration of the flight dynamics limitations. At each control cycle, MPC computes the optimal trajectory for the remainder of the mission based on the current situation, and then uses the first control input from this optimal sequence as the current control to be applied. In order to reduce the dimensionality of the optimization, the control inputs can be approximated by using orthogonal basis function such as Laguerre, Legendre, Chebyshev, or Hermite functions. In the effort reported here, Laguerre functions are used as the basis functions for the control inputs. Jung et al [26] mention that the Laguerre functions exponentially decrease near the end of the time horizon, and this eliminates problems associated with having the optimal control values peak at the end of horizon, as occurs with some other basis functions. At the beginning of the time horizon, the use of Laguerre functions as basis functions allows for faster early control action, which in turn provides better performance.

The problem discussed here is to find the optimal sequence of control variables \( u^* \) which guides the UAV to the target in a dynamic environment while minimizing the cost function, which often included total fuel usage. A finite time horizon and discretized system dynamics are used in order to solve this problem numerically. The finite time horizon is first divided into \( N_p \) discrete time steps, where:

\[
N_p = \frac{t}{\Delta t}
\]  

(1)

In order to find \( u^* \), the following cost function must be minimized:

\[
J_{NAV} = \sum_{n=1}^{N_p} (e^T Q e + U^T RU)
\]  

(2)

Eq. (2) is a classical quadratic form, and it is utilized here as a navigation cost function which combines a penalty on the tracking error between the UAV’s current position and the target point, and a penalty on the control effort [25]. We wish to also consider minimum fuel consumption in this paper, therefore, a fuel cost component is also required to be present in the cost function. To this end, the following term is also included in the cost:

\[
J_{FUEL} = \frac{1}{t_f} \int_0^{t_f} c \text{Tr} \, dt
\]  

(3)

Eq. (3) is the average fuel cost function, but it is formulated in a continuous time form [27]. So, it must be modified to a discrete-time form. The “\( c \cdot \text{Tr} \)” term here is thrust-specific fuel consumption \( c \) times the thrust, and is therefore the mass flow rate of fuel, and \( t_f \) is the length of the prediction horizon because the duration of each phase of the predicted trajectory of the UAV is the same as the prediction horizon length. Let the mass flow rate or the fuel be represented by the term “\( \dot{m}_f \)”, i.e.:

\[
c \cdot \text{Tr} = -\dot{m}_f = -\frac{dm_f}{dt}
\]  

(4)

By substituting this into Eq. (3) and noting that the term “\( dt \)” is canceled out, we have:

\[
-\frac{1}{t_f} \int_0^{t_f} \dot{m}_f \, dt = \frac{1}{t_f} \int_0^{t_f} \dot{m}_f \, dt = \frac{1}{N_p} \sum_{l=0}^{N_p} \Delta m_f
\]  

(5)

Therefore, the discrete-time representation of the fuel usage is:

\[
J_{FUEL} = \frac{1}{N_p} \sum_{l=0}^{N_p} (m_{f+l} - m_f)
\]  

(6)

And the overall cost function is then:

\[
J = J_{NAV} + J_{FUEL}
\]  

(7)

where \( J_{NAV} \) and \( J_{FUEL} \) are respectively the cost function for navigation and fuel consumption. \( m_f \) and \( w_f \) are the fuel mass and the fuel consumption weighting. \( N_p \) is the prediction horizon or the length of the optimization window. The cost function presented in Eq. (7) must be minimized while avoiding the obstacles and threats. We assume that the vehicle’s motion dynamics are given by the following nonlinear, discrete-time, flight dynamic system in state space form:

\[
X(k+1) = \phi(X(k),u(k))
\]  

(8)

3

American Institute of Aeronautics and Astronautics
The sequence of control variables must further satisfy the constraints:

\[ u_{\text{min}} \leq u \leq u_{\text{max}} \]  \hspace{1cm} (9)

where \( t \) is a finite time horizon, and \( e \) is the tracking error between the target position and the output UAV position in Cartesian coordinates. Thus, \( e \) is defined as \( e = R_f(k_i) - Y(k_i + m|k|) \). \( Q \) and \( R \) are output and control weighting matrices, respectively, and are assumed to be diagonal matrices with \( Q \) positive semi-definite and \( R \) positive definite.

The desired path must satisfy the physical constraints of the UAV. The UAV here is treated as a point mass that moves at constant speed. Nonlinear differential equations are solved using an approximate Euler’s method or a 4th order Runge-Kutta (RK4) method. This research uses a nonlinear dynamic system because although the linearization of dynamics helps reduce computational complexity, the linearized aircraft dynamics limits the maneuverability in a dynamic environment.

The developed approach is evaluated using a simulated scenario comprising a dynamic environment that contains static obstacles, randomly generated pop-up threats, and a moving no-fly zone. The pop-up threats are designed by a Markov chain method. The desired UAV trajectory must obviously avoid all obstacles and pop-up threats. The cost function related with obstacle avoidance must be added to Eq. (2). The following section details this process.

III. Method Development

A. UAV Dynamic Model

The equations of motion for a point mass vehicle are given by the following system of ordinary differential equations [28, 29]

\[
\begin{align*}
\dot{V} &= \frac{Tr - Dg}{mass} - g \sin \gamma \quad \text{(10)} \\
\dot{n} &= \frac{g}{V} \left[ n \cos \phi - \cos \gamma \right] \quad \text{(11)} \\
\dot{\psi} &= \frac{g n \sin \phi}{V \cos \gamma} \quad \text{(12)} \\
\dot{\gamma} &= -V \sin \gamma \quad \text{(13)} \\
\dot{x} &= V \cos \gamma \cos \chi \quad \text{(14)} \\
\dot{y} &= V \cos \gamma \sin \chi \quad \text{(15)} \\
m_f &= -c Tr \quad \text{(16)}
\end{align*}
\]

where \( Tr \) is thrust, \( Dg \) is drag, \( c \) is specific fuel consumption, \( z \) is altitude change, \( n \) is the load factor, \( \phi \) is the bank angle, and \( \gamma \) is the flight path angle.

We assume here that the UAV flies in 2-D environment with constant speed. Therefore, in the above equations, we have let \( \dot{V} = 0, \dot{n} = 0, \gamma = 0, \) and \( \dot{\gamma} = 0. \) The altitude is constant, therefore a steady and coordinated turn is required. Therefore, the load factor becomes

\[ n = \frac{1}{\cos \phi} \]  \hspace{1cm} (17)

and Eq. (17) is substituted in Eq. (12) and it turns out to be Eq. (18).

\[ \dot{\psi} = \frac{g \tan \phi}{V \cos \gamma} \]  \hspace{1cm} (18)

The state variables are \( X = [x \; y \; m_f]^T \) and input variables are \( u = [Tr \; \phi]^T \).
B. Laguerre Functions

The control vector is a sequence of \( N_c \) vectors over the control horizon of length \( N_c \) time steps, defined by \( U = [u(k_i) \ u(k_i+1) \ u(k_i+2) ... u(k_i+N_c)]^T \). \( k_i \) is the initial time of the moving horizon window. Using an expansion of the control inputs in terms of discrete time Laguerre functions, the control vector \( N_c \) can be reduced to the number of \( n_\eta \), \( N_c > n_\eta \). Wahlberg [30] defines the following discrete-time Laguerre function in the Z-transform domain:

\[
\Gamma_m(z) = \frac{\sqrt{1-a^2}}{z-a} \left( 1 - \frac{z}{z-a} \right)^{m-1}
\]  

(19)

where \( a \) is pole in Z-domain / scaling factor and \( m \) is \( m^{th} \) future sample in discrete time. The pole \( a \) is chosen between 0 and 1 for stability.

Laguerre functions possess the property of orthonormality, and Wang [25] verifies the orthonormal property in the frequency domain using Eq. (19), as:

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_m(e^{j\omega}) \cdot \Gamma_m(e^{j\omega})^* d\omega = 1
\]

(20)

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_m(e^{j\omega}) \cdot \Gamma_n(e^{j\omega})^* d\omega = 0 \quad \text{if} \ m \neq n
\]

(21)

The asterisk denotes the complex conjugate in Eq. (20) and Eq. (21). In the time domain, the orthonormality of Eq. (20) and Eq. (21) can be written as:

\[
\sum_{m=n}^{NP} l_m(m) \cdot l_j(m) = 0 \quad \text{for} \ i \neq j
\]

(22)

\[
\sum_{m=n}^{NP} l_i(m) \cdot l_j(m) = 1 \quad \text{for} \ i = j
\]

(23)

where \( l_m \) is the \( m^{th} \) discrete time Laguerre function.

The corresponding discrete time domain Laguerre function can be obtained by inverting the z-transform of \( \Gamma_m(z) \), i.e. \( l_m(k) = Z^{-1}[\Gamma_m(z)] \). Moreover, the impulse response, \( h \), of a stable system can be represented as:

\[
h(k) = \sum_{m=1}^{\infty} g_m l_m(k)
\]

(24)

where \( g_m \) is the coefficient associated with the \( m^{th} \) Laguerre function.

At time \( k_i \), the sequence of control variables \( U = [u(k_i) \ u(k_i+1) ... u(k_i+k) ... ] \) can be regarded as the impulse response of a stable system. Therefore, at the future sampling instant \( k \), the input can be approximately expressed as:

\[
u(k_i+k) \approx \sum_{m=1}^{n_\eta} g_m l_m(k)
\]

(25)

\[
u(k_i+k) = L(k)^T \eta
\]

(26)

with the number of terms \( n_\eta \) in Eq. (25). Eq. (26) is the vector form, where \( \eta \) and \( L \) are, respectively, a coefficient vector and a vector of discrete time Laguerre functions. Let \( \eta \) and \( L \) be \( \eta = [g1 \ g2 \ g3 \ ... \ g_{n_\eta}]^T \) and \( L = [l_1(k) \ l_2(k) \ l_3(k) \ ... \ l_{n_\eta}(k)]^T \).

The set of Laguerre function satisfies the following difference equation:

\[
L(k+1) = \Xi \cdot L(k)
\]

(27)
Which we obtain by taking the realization of the $z$-transfer function in Eq. (19). In Eq. (27), where

$$ G(z) = \begin{bmatrix}
  \frac{a}{1 - a^2} & 0 & \cdots & 0 \\
  -a \cdot (1 - a^2) & \frac{a}{1 - a^2} & \cdots & 0 \\
  a^2 \cdot (1 - a^2) & -a \cdot (1 - a^2) & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  (-1)^{n \eta - 2} \cdot a^{n \eta - 2} \cdot (1 - a^2) & (-1)^{n \eta - 3} \cdot a^{n \eta - 3} \cdot (1 - a^2) & \cdots & a
\end{bmatrix} $$

and the initial condition of the Laguerre function is $L(0) = \sqrt{1 - a^2} \left[ 1 - a \cdot a^2 \cdot (-1)^{n \eta - 1} \cdot a^{n \eta - 1} \right]^T$.

Therefore, the control effort in the cost function can be rewritten by substituting Eq. (22), Eq. (23), and Eq. (26) into Eq. (2). The resultant cost function is:

$$ J = \sum_{m=1}^{N_p} \left( e^T Q e + \eta^T R \eta + \frac{1}{N_p} \sum_{t=m-1}^{N_p} \left( m_{f_{t+1}} - m_{f_{t}} \right) \right) $$

The first part of Eq. (28) is given in [25]. However, if the representation of the control is changed, then the values of the control constraints must also be modified in terms of the Laguerre functions, i.e., it is not easy to select the initial $\eta$ within proper boundaries. So, the initial $u$, $u_{\text{min}}$, and $u_{\text{max}}$ are first selected with a specification of small UAV. Then, the initial $\eta$ and constraints are obtained as follows:

$$ \eta_i = L^{-1} u_i \quad \text{for } i = \text{initial, min, or max} $$

Therefore, the constraint Eq. (9) can be rewritten:

$$ \eta_{\text{min}} \leq \eta \leq \eta_{\text{max}} $$

The next step consists of selecting the proper values for the pole $a$ and the appropriate number of terms $n_\eta$. Selecting the values for $a$ and $n_\eta$ affect the UAV performance and the computational load. Consider Figs. 1-2, generated using SIMULINK®. The heading angle time history when $a = 0.4$ and $a = 0.9$ are selected for a constant value of $n_\eta = 10$. In this case, the UAV flies from the initial position (0,0) to the target position (500, 500) with 0 degree initial heading angle without any obstacles. When $a = 0.4$, the output value of heading angle is increased to around 0.9 radians and converges about 0.9 radians and the UAV succeeds in reaching the target. On the other hand, when $a = 0.9$, the output keeps increasing and the UAV flies in a big circle in order to reach the target, failing in its task.

![Figure 1. Comparison of the Heading angle time history between two different value of scaling factor, $a$](image.png)
Figure 2. Comparison of the flying path between two different value of scaling factor, \( a \)

Now, the focus moves to selecting a value for \( n_\eta \). \( n_\eta = 5, 12, \) and \( 20 \) are selected and let \( a = 0.4 \), based on the above evaluation. Figure 3 plots the heading angle time history. As \( n_\eta = 20 \) is chosen, the output is more jittery than at \( n_\eta = 5 \) which is the selected value. Note that the computation time of two cases is definitely different, i.e., when \( n_\eta = 20 \) is selected, the computation time is twice as slower than for the \( n_\eta = 5 \) case. The value of \( n_\eta \) has to be selected so that it is less than or equal to that of control prediction.

Figure 3. Comparison of the Heading angle time history between two different number of parameter, \( n_\eta \)

C. Model Predictive Control

The MPC algorithm uses an internal model to predict the plant behavior and repeatedly minimizes the cost function the next \( N_p \) time steps by the choice of the control over the next \( N_c \) time steps, where the present time is denoted \( k \). MPC is also called Receding Horizon Control. Figure 4 illustrates the basic concept of a discrete-time MPC algorithm. Each discrete time step size is \( \Delta t \) seconds and the prediction horizon and control horizon are \( N_p \) and \( N_c \); therefore, the prediction interval and the control sequence spans \( N_p \cdot \Delta t \) and \( N_c \cdot \Delta t \) respectively. The control values are held constant until the next time step, and then their values are instantaneously changed. After the end of control horizon, the control value is held constant until the prediction horizon is ended.

Figure 4. The concept of model predictive control
The MPC uses the nonlinear dynamic model of the plant. The differential equations, Eq. (13) - (16) and (18), need to be approximated and 4th order Runge-Kutta (RK4) method is used to achieve a higher accuracy. RK4 requires four evaluations such that derivative once at the beginning point, twice at trial midpoints and once at a trial endpoint to calculate $x(k+1)$ [31]. The higher order term $O(\Delta t^5)$ is neglected from Eq. (31). Eq. (31) applies to nonlinear continuous time model with the values of Eq. (32) - (35) and Eq. (8) is then obtained in discrete time domain.

$$X(k + 1) = X(k) + \frac{\Delta t}{6} (K_1 + 2K_2 + 2K_3 + K_4) + O(\Delta t^5)$$  \hspace{1cm} (31)

$$K_1 = F(X(t), \eta, L)$$  \hspace{1cm} (32)

$$K_2 = F\left(X(t) + \frac{\Delta t}{2} K_2, \eta, L\right)$$  \hspace{1cm} (33)

$$K_3 = F\left(X(t) + \frac{\Delta t}{2} K_2, \eta, L\right)$$  \hspace{1cm} (34)

$$K_4 = F\left(X(t) + \Delta t K_3, \eta, L\right)$$  \hspace{1cm} (35)

In the case of using the Laguerre function, the algorithm computes a sequence of the Laguerre parameter by repeatedly minimizing the cost function, Eq. (28). When the optimal sequence of Laguerre parameter is obtained, the receding horizon control law is applied as:

$$u^*(k) = L(0)^T \eta^*$$  \hspace{1cm} (36)

From Eq. (36), the computed optimal control variables, $u^*$, are applied to the plant. The predicted trajectory of plant is then developed. This procedure is repeatedly executed until the UAV approaches the target and a follow-up path of the UAV. The concept of MPC, based on the Laguerre function, is summarized in the following Pseudo code A.

Pseudo code A

1. Selecting initial input $u$ and boundaries $u_{\text{max}}$ and $u_{\text{min}}$
2. Computing initial $\eta$, $\eta_{\text{max}}$, and $\eta_{\text{min}}$ (see Eq. (29))
3. while task $\neq$ done
4.   Optimizing the cost function for $\eta^*$ (see Eq. (28))
5.   Updating state variables using the optimized $\eta^*$ (see Eq. (36))
6.   check for completion of task
7.     if true,
8.        task $=$ done
9.     else
10.    task $\neq$ done
11. end
12. end

After changing notation of input variables to coefficient of the Laguerre function, the control effort part in the cost function doesn't have any specific physical meaning. So, an intuitive trial-and-error method for selection of the weighting matrices is not practical. Instead, the approach introduced by Zhang et al [32] to choose $Q$ and $R$ efficiently is employed as follows:

$$Q = \text{diag}[\eta_1, \eta_2], \quad R = \text{diag}[r_1, r_2, \ldots, r_m]$$  \hspace{1cm} (37)

The feasible region of coefficients of Laguerre function, Eq. (30), is rewritten as

$$\eta_{i_{\text{min}}} \leq \eta_i \leq \eta_{i_{\text{max}}} \quad i = 1, 2, \ldots, n_{\eta}$$  \hspace{1cm} (38)

Then the diagonal elements of $R$ is then computed in Eq. (39).

$$r_i = \frac{1}{(\eta_{i_{\text{min}}} - \eta_{i_{\text{max}}})^2} \quad i = 1, 2, \ldots, n_{\eta}$$  \hspace{1cm} (39)
D. Obstacle Avoidance

The UAV flies in an environment which may contain static obstacles, moving obstacles, or pop-up threats. Static obstacles are described as polygons, the moving obstacle is treated as a rectangular no-fly zone, and pop-up threats are geometrically described as circles (mainly for simplicity). In the scenario studied in this effort, the static obstacles are known apriori and they do not undergo any changes during the entire mission. Pop-up threats symbolize the action of the adversary (“Red Team”) and the spatio-temporal nature of these threats is uncertain. If pop-up threats appear on a specific target, the UAV has to abandon the target. The direction of the moving obstacle is randomly changed in the scenarios. At every time step, the motion of a moving obstacle is discretized and at the corresponding position, the obstacle is redefined as a new static obstacle at that instant of time. The discrete time moving obstacle dynamics are as follows:

\[
\begin{align*}
x_{\text{obs}}(k+1) &= x_{\text{obs}}(k) + \Delta t V_{\text{obs}} \cos \chi_{\text{obs}} \\
y_{\text{obs}}(k+1) &= y_{\text{obs}}(k) + \Delta t V_{\text{obs}} \sin \chi_{\text{obs}} \\
\chi_{\text{obs}}(k+1) &= \chi_{\text{obs}}(k) + \Delta t u_{\text{obs}}(k)
\end{align*}
\]

where \( V_{\text{obs}} \) and \( \chi_{\text{obs}} \) is a velocity and a heading angle of moving obstacle. \( u_{\text{obs}} \) is a heading angle input for an obstacle.

It is assumed that the UAV is equipped with a suitable sensor suite which enables it to detect obstacles as soon as they appear in the scenario space. Moreover, the MPC algorithm needs to have the capability of computing collision avoidance utilizing the situational awareness estimated from the sensor suite. Furthermore, the cost function (Eq. (28)) requires one more term, which corresponds to avoidance obstacle. Kang et al [33] introduce the collision avoidance cost as:

\[
J_{\text{obs}} = \sum_{i} N_{\text{obs}} \left[ \sum_{j=1}^{N_{\text{obs}}} \alpha_{j} \exp \left( \frac{(x(0) - x_{\text{obs}}(0))^2 + (y(0) - y_{\text{obs}}(0))^2}{\beta^2} \right) \right]
\]

where \( N_{\text{obs}} \) is the number of obstacles and \( \alpha \) and \( \beta \) are the gain values for avoiding obstacle. \( x \) and \( y \) provide the two-dimensional location of the UAV and \( x_{\text{obs}} \) and \( y_{\text{obs}} \) describe the location of the obstacle.

If the UAV closes to the position of each obstacle, the cost function is penalized due to the exponential term, which includes gains \( \alpha \) and \( \beta \). These gains affect the result of the exponential term. As the values of \( \alpha \) and \( \beta \) increase, the UAV is properly repelled from the obstacle. The composite cost function is adding up Eq. (28) and Eq. (43), i.e.,

\[
J_{\text{tot}} = J_{\text{nav}} + J_{\text{fuel}} + J_{\text{obs}}
\]

1. Obstacles and pop-up threat detection formulation

To include Eq. (43), the minimum distance between the UAV and a moving obstacle needs to be computed by using geometry and linear algebra [34]. This consists of computing the distance between a point and a line segment and a point and a circle. The UAV is treated as a point. So, if the computed minimum distance is less than predefined allowable distance, which is based on the UAV’s dimensions, the UAV will collide with the obstacle.

The equation of a line segment between two points, \( L_{p1} = [x_{\text{obs}1}, y_{\text{obs}1}] \) and \( L_{p2} = [x_{\text{obs}2}, y_{\text{obs}2}] \) is the following.

\[
L_{p2} = L_{p1} + \tau \mathbf{u}
\]

where \( \mathbf{u} = [x_{\text{obs}2} - x_{\text{obs}1}, y_{\text{obs}2} - y_{\text{obs}1}] = [u_x, u_y] \) and \( 0 \leq \tau \leq 1 \).

The minimum distance, \( d(\tau_{\text{min}}) \), between a point \( L_{p} = [x, y] \) and an arbitrary point on a line \( [x_{\text{obs}1} + \tau u_x, y_{\text{obs}1} + \tau u_y] \) is

\[
d(\tau_{\text{min}}) = \sqrt{(x - x_{\text{obs}1} - \tau u_x)^2 + (y - y_{\text{obs}1} - \tau u_y)^2}
\]

where \( \tau_{\text{min}} = \frac{u_x(x - x_{\text{obs}1}) + u_y(y - y_{\text{obs}1})}{u_x^2 + u_y^2} \) as illustrated in Figure 6.

For the line segment, if \( \tau_{\text{min}} > 1 \), then let \( \tau_{\text{min}} \) be 1 and if \( \tau_{\text{min}} < 0 \), then let \( \tau_{\text{min}} \) be 0. The shortest distance from the UAV to the obstacle can be found. When the shortest distance between the UAV and the moving obstacle is less than allowable distance, then Eq. (43) is applied.
When a pop-up threat appears, the UAV may take a long path instead of optimal path. If the UAV attempts to reach the target and the moving obstacle flies between them, then the UAV flies along beside the obstacle until the obstacle disappears between the target and the UAV. So, additional waypoints are needed and these waypoints are located around convex vertices of each obstacle. Placing waypoints at concave vertices are not necessary because when the UAV flies around the obstacle, waypoints at the convex vertices make the shortest path. Even though the UAV is in the concave region, the UAV approaches the waypoint at the convex vertex. If a straight line is drawn from the current position of the UAV to the target and the line passes through an obstacle, waypoints split into two groups. Distances from each waypoint to the target are computed and are added up by group. The waypoints from the group, which has less the summation than the other has, are selected. In the case of dealing with a moving obstacle, if the obstacle is faster than the UAV is, then the UAV should fly in the opposite direction of the obstacle; otherwise, the UAV can fly ahead of it. When the static obstacle is detected by the scanner of the UAV, the UAV searches pertinent waypoints; otherwise the UAV flies to the target directly. Figure 6 and Figure 7 depict the schemes. The mathematical procedure is shown as follows:

Let two lines be $A_1X - B_1Y = -C_1$ and $A_2X - B_2Y = -C_2$, and the matrix form can be written as:

$$
\begin{bmatrix}
A_1 & -B_1 \\
A_2 & -B_2
\end{bmatrix}
\begin{bmatrix}
X \\
Y
\end{bmatrix}
= 
\begin{bmatrix}
-C_1 \\
-C_2
\end{bmatrix}
$$

(47)

$$
\begin{bmatrix}
X \\
Y
\end{bmatrix} = 
\begin{bmatrix}
A_1 & -B_1 \\
A_2 & -B_2
\end{bmatrix}^{-1}
\begin{bmatrix}
-C_1 \\
-C_2
\end{bmatrix}
$$

(48)

If the inverse doesn’t exit, two lines are parallel. Otherwise, $\begin{bmatrix}
X \\
Y
\end{bmatrix}$ is the intersection point.
Finding the minimum distance between a point and a pop-up threat is straightforward. Let the center of the pop-up threat be \((x_{\text{obs}}, y_{\text{obs}})\) and its radius be \(R_r\). The radius of the pop-up threat can be determined by the sensor or scanner of the UAV. The radius is various at every different pop-up threat.

A line between two points, \(L_p = (x, y)\) and \(L_{p2} = (x_{\text{pop}}, y_{\text{pop}})\), is the following:

\[
\Psi = \xi \Theta + \lambda
\] (49)

where \(\xi = \frac{y_{\text{pop}} - y}{x_{\text{pop}} - x}\) and \(\lambda = \frac{x_{\text{obs}} y_{\text{obs}} - y_{\text{obs}} x_{\text{obs}}}{x_{\text{obs}} - x}\)

And the equation of circle is

\[
(X - x_{\text{pop}})^2 + (Y - y_{\text{pop}})^2 = R_r^2
\] (50)

If \(x_{\text{pop}} \neq x\) from \(\xi\), then substituting Eq. (49) into Eq. (50) and the resultant is then

\[
(\xi^2 + 1)X^2 + 2(\xi \lambda - \xi y_{\text{pop}} - x_{\text{pop}})X + (x_{\text{pop}}^2 + y_{\text{pop}}^2 - R_r^2 + \lambda^2 - 2 \lambda y_{\text{pop}}) = 0
\] (51)

and let the first term of Eq. (51) be \(A\), the second term be \(B\), and the last term be \(C\), i.e.,

\[
A = \xi^2 + 1
\]
\[
B = 2(\xi \lambda - \xi y_{\text{pop}} - x_{\text{pop}}) = 2B_1
\]
\[
C = (x_{\text{pop}}^2 + y_{\text{pop}}^2 - R_r^2 + \lambda^2 - 2 \lambda y_{\text{pop}})
\]

Then the discriminant, \(D/4\), is

\[
D/4 = B_1^2 - AC
\] (52)
The points on the circles, \((x_{\text{cir}1}, y_{\text{cir}1})\) and \((x_{\text{cir}2}, y_{\text{cir}2})\), are computed in Eqs. (53) - (54) and are shown in Figure 8.

\[
\begin{align*}
x_{\text{cir}1/\text{cir}2} &= \pm \frac{B1 + \sqrt{D/4}}{A} \\
y_{\text{cir}1} &= \xi x_{\text{cir}1} + \lambda \\
y_{\text{cir}2} &= \xi x_{\text{cir}2} + \lambda
\end{align*}
\] (53)

If \(x_{\text{pop}} = x\) from \(\xi\), then

\[
x_{\text{cir}1} = x
\] (54)

Then, the point which is closest to the location of UAV is chosen.

\[
y_{\text{cir}1/\text{cir}2} = y_{\text{pop}} \pm \sqrt{Rr^2 - (x - x_{\text{pop}})^2}
\]

Figure 8. Illustration of the distance between a UAV and a pop-up threat

2. Pop-up Threats Design

The locations of static obstacles, such as no fly zone or urban terrains, are previously known while the pop-up threats are not detected in advance at the beginning of operation. The appearance and disappearance of pop-up threats are randomly occurred in an adversarial area [35]. Lie et al [35] and Subramanian et al [36] use the first order Markov Chain (FOMC) with one step stationary transition probability for generating the sequence of location of pop-up threats. The concept of FOMC is that the next location depends only on the current location and independents on the previous location. This can be written in mathematical notation.

\[
\begin{align*}
\text{Prob}\{Z_{k+1} = j | Z_k = i, Z_{k-1} = i_{k-1}, \ldots, Z_0 = i_0\} &= \text{Prob}\{Z_{k+1} = j | Z_k = i\} \quad i, j \in \mathcal{O}
\end{align*}
\] (55)

The one-step stationary transition probability matrix (see Eq. (56)) is randomly set up. These random values are uniformly distributed on the unit interval \([0, 1]\) and the sum of each row has to be 1, i.e., \(p_{k1} + p_{k2} + \ldots + p_{kN} = 1\), \(k = 1, 2, 3, \ldots, N\).

\[
P^{(1)} = \begin{bmatrix}
p_{11} & p_{12} & \cdots & p_{1N} \\
p_{21} & p_{22} & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
p_{N1} & p_{N2} & \cdots & p_{NN}
\end{bmatrix}
\] (56)

For generating the sequence of location of pop-up threats, the initial distribution, Eq. (57), needs to be generated.

\[
P^{(0)} = [p_1, p_2, \ldots, p_N]
\] (57)
Given Eqs. (56) and (57), the sequence of pop-up location, Eq. (58), is generated using the initiation function, Eq. (59), and the update function, Eq. (60). Häggström [37] has checked the validity of selecting the initiation and update function with considering the properties of FOMC. The generated random numbers $U_0, U_1, \ldots, U_N$ are a sequence of independent identically distributed (I.I.D.) variables which are uniformly distributed on the unit interval $[0, 1]$. Let the sequence of location of pop-up threats be $\{Z_0, Z_1, \ldots, Z_N\}$. The first location of pop-up threat, $Z_0$, is generated by applying the initiation function. The next location, $Z_1$, is generated with using the previous location, $Z_0$, and the update function. This procedure is iteratively computed until all locations are generated.

$$Z_0 = \rho(U_0)$$

$$Z_2 = \sigma(Z_1, U_2)$$

$$\vdots$$

$$Z_N = \sigma(Z_{N-1}, U_N)$$

where the initiation function $\rho$ is set

$$\rho(x) = \begin{cases} 
1 & \text{for } x \in [0, p_1) \\
2 & \text{for } x \in [p_1, p_1 + p_2) \\
\vdots & \vdots \\
i & \text{for } x \in [\sum_{j=1}^{i-1} p_j, \sum_{j=1}^{i} p_j) \\
\vdots & \vdots \\
N & \text{for } x \in [\sum_{j=1}^{N-1} p_j, 1]
\end{cases}$$

(59)

and the update function $\sigma$ is set

$$\sigma(i, x) = \begin{cases} 
1 & \text{for } x \in [0, p_{1,1}) \\
2 & \text{for } x \in [p_{1,1}, p_{1,1} + p_{2,1}) \\
\vdots & \vdots \\
j & \text{for } x \in [\sum_{j=1}^{i-1} p_{j,1}, \sum_{j=1}^{i} p_{j,1}) \\
\vdots & \vdots \\
N & \text{for } x \in [\sum_{j=1}^{N-1} p_{j,1}, 1]
\end{cases}$$

(60)

After generating the locations, their Cartesian coordinates can be computed in Eq. (61).

$$\begin{cases} 
x_{\text{pop}} = (\text{mod}(j, \delta) - 1) \times \frac{r}{\delta} + \frac{r}{2\delta} & \text{if mod}(j, \delta) \neq 0 \\
y_{\text{pop}} = \text{int}\left(\frac{1}{\mu}\right) \times \frac{w}{\mu} + \frac{w}{2\mu}
\end{cases}$$

(61)

where mod$(j, \delta)$ is the value of remainder after $j$ divided by $\delta$ and int$(j / \mu)$ is the value of integer after $j$ divided by $\mu$.

IV. Results

A representative dynamic scenario was selected and simulations were performed to evaluate the effectiveness of the MPC based UAV path planner. The velocity of UAV is constant at 20 m/s and the initial conditions are $\chi(0) = 0^\circ$ and $m_f = 3.3639$ kg.

The control constraint is as follows:

$$-43^\circ < \text{Bank angle} < 43^\circ$$

The thrust is set to be constant at 20 Nt. The size of UAV here is based on the model: RQ-7 shadow-200. For this UAV, the minimum radius of turn is approximately 40 m. The sampling time, the prediction horizon, and the control horizon are respectively $\Delta t = 0.4$ second, $N_p = 21$, and $N_c = 5$. The number of parameters and the scale factor of the Laguerre functions used for each control variable are $n_\eta = 10$ and $a = 0.1$. The weighing matrices, $Q = [1; 1]$ is chosen and $R$ is appropriately chosen by computing Eq. (39).

There are five targets and five static obstacles in a scenario. The size of the two dimensional scenario space is 13km X 9.5km. Initially, the UAV is randomly placed and flies to each target once and returns to its starting position upon completion of the tour. In this scenario, the starting positions of the UAV and moving obstacle are respectively near in the middle of the left side and in the bottom of the left side of the map. The target locations are (5700, 6500), (7000, 4450), (11000, 6000), (6800, 2000), and (11500, 3500). In Figure 9, a red lined rectangle is a moving obstacle and it moves around the area at 100 m/s. The trajectory of the UAV is represented by little triangles. There are 9 pop-up threats and some pop-up may have the same location. When the pop-up threat appears once, the UAV collects the information about the pop-up location and its size and keeps updating the information. The traces of the pop-up threats remain where they appear because there may have a chance to pop up again; therefore, the UAV
needs to avoid this dangerous area in the future. Figure 10 shows a "zoom in" on the region where the UAV turns around. The diameter of turn between the target and the right hand side of the trace of UAV is approximately 80 meters. Even though the moving no fly zone approaches when the UAV is close to the static obstacle, the UAV avoids not only the static obstacle but also the moving no fly zone without violating control constraints (see Fig 11). The upper two sub-plots of Figure 11 indicate control trajectories and lower two sub-plots are state trajectories.
This paper presents the development of a new UAV path-planning algorithm which can be used in a dynamic environment. By using an MPC approach to optimization, this method can address collision avoidance and the physical constraints associated with nonlinear UAV dynamics. However, due to the intensive computational needs of an optimal non-linear MPC formulation, which is not feasible for meaningful UAV applications, discrete time Laguerre basis functions are used to represent the control inputs in order to obtain rapid yet robust solutions. This proposed approach successfully satisfies that the UAV can navigate from the starting point to the target without colliding with obstacles and pop-up threats in real-time. Further work will include six-DOF UAV dynamics and a 3D scenario.

References


