Real-time Obstacle Collision Avoidance for Fixed Wing Aircraft Using B-splines

Hamid Alturbeh
School of Engineering, Cranfield University
Bedfordshire MK43 0AL, UK.
Email:h.alturbeh@cranfield.ac.uk

James F. Whidborne
School of Engineering, Cranfield University
Bedfordshire MK43 0AL, UK.
Email:j.f.whidborne@cranfield.ac.uk

***Abstract*—A real-time collision avoidance algorithm is developed based on parameterizing an optimal control problem with B-spline curves. The optimal control problem is formulated in output space rather than control or input space, this is feasible because of the differential flatness of the system for a fixed wing aircraft. The flat output trajectory is parameterized using a B-spline curve representation. In order to reduce the computational time of the optimal problem, the aircraft and obstacle constraints are augmented in the cost function using a penalty function method. The developed algorithm has been simulated and tested in MATLAB/Simulink.**

***Keywords*—Aircraft control, Aerospace trajectories, Autonomous vehicles, Splines, Obstacle avoidance, Receding horizon.**

I. Introduction

Unmanned Aircraft Systems (UAS) are of increasing importance in the aerospace industry for both civilian and military applications due to their ability to complete dull, dirty and dangerous missions [1]. However, operation of Unmanned Aerial Vehicles (UAV's) in civil/non-segregated airspace is restricted by the policies of aviation authorities which require full compliance with rules and obligation that apply for manned aircraft [2]. Trajectory tracking and collision avoidance are issues that a UAV must deal with in a way that gives the UAV the ability to avoid conflict situations. Thus, any UAV that will be operated in civil/non-segregated airspace must be equipped with a collision avoidance system that has the ability to avoid conflict scenarios in full compliance with airspace traffic rules. Much research is being undertaken to enable the routine use of UAV's in all classes of airspace without the need for restrictive or specialized conditions of operation. The ASTRAEA program [3] is one example.

Trajectory planners can be divided into two main categories [4]; global planners which require good knowledge about the environment that the aircraft is going to fly in, and local trajectory planners which are algorithms that run continuously in order to allow the aircraft to deal with events that may happen during the flight. Many methods for generating trajectories that guarantee collision avoidance have been proposed in the literature. For example: predefined, protocol based [5], E-field [6], geometric [7] and automotive [8].

This paper presents an approach for generating collision avoidance trajectories based on B-spline curves. Essentially, a finite-horizon optimal control problem is periodically solved in real-time hence updating the aircraft trajectory to avoid obstacles and drive the aircraft to its global path. The proposed approach can be summarized as follows:

1) Given a global trajectory that the aircraft is required to follow, solve the optimal control problem

$$\min_{\boldsymbol{U}(t)\in\mathcal{U}} J(\boldsymbol{U}(t)) \tag{1}$$

subject to the aircraft dynamics constraints pair, $(\dot{\boldsymbol{X}} = f(\boldsymbol{X},\boldsymbol{U}), \boldsymbol{Y} = g(\boldsymbol{X}))$, state constraint, $\boldsymbol{X}(t) \in \mathcal{X}$, and aircraft trajectory obstacles constraint, $\boldsymbol{Y}(t) \in \mathcal{Y}$, where $\boldsymbol{U} \in \mathcal{U}$ is the control and J is a cost measured over a finite time horizon, $t \in [t_0, t_f]$, that drives the local trajectory to the global trajectory.
2) The problem is solved by a direct method by inverting the dynamics, so the optimization is performed in the output space $\boldsymbol{Y}(t) \in \mathcal{Y}$, and parameterizing the trajectory by a spline function. The cost is augmented to maintain the constraints.
3) The generated local trajectory allows the UAV to track the global trajectory while avoiding any intruder or conflict scenarios that may occur. The local trajectory optimization is periodically solved on-line in a receding horizon approach to account for system uncertainties and obstacle changes.

In Section II, the system model is described. Section III discusses B-spline basis functions and curves. Section IV shows the formulation of the optimal problem to find the optimal local trajectories. Section V presents the simulation results.

II. Fixed Wing Aircraft Model

A fixed wing aircraft dynamic can be expressed by a point-mass model [9]:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\gamma}\\ \dot{\chi}\\ \dot{V}\end{bmatrix} = \begin{bmatrix} V\cos\gamma\cos\chi \\ V\cos\gamma\sin\chi \\ V\sin\gamma \\ (g/V)(n\cos\phi - \cos\chi) \\ (g/V)(n\sin\phi/\cos\gamma) \\ (T-D)/m - g\sin\gamma \end{bmatrix} \tag{2}$$

where x, y, z are the aircraft center of gravity coordinates in earth axis, γ is the flight-path angle, χ is the heading angle, V is the aircraft speed, g is the gravity acceleration, ϕ is the bank angle, T is the thrust, D is the drag, m is the total mass, $n = L/(mg)$ is the load factor and L is the total aircraft lift.

The input and output vectors are defined respectively as:

$$\boldsymbol{U} = [\phi \quad T \quad n]^T \qquad \boldsymbol{Y} = [x \quad y \quad z]^T \tag{3}$$

In order to determine an optimal control trajectory for aircraft using direct methods, the optimal control problem is formulated in output space rather than control or input space. However, the output design space technique is only available when the system is differentially flat [10]. A system is differentially flat if its states and inputs can be expressed as functions of the output vector and its derivatives [10], [11]. Fortunately most fixed-wing aircraft systems can be considered as differentially flat system, the following discussion shows that the fixed wing aircraft possesses the property of flatness. Modifying (2) obtains:

$$V = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{4}$$
$$\gamma = \arcsin\left(\dot{z}/V\right) \tag{5}$$
$$\chi = \arcsin\left(\dot{y}/(V\cos\gamma)\right) \tag{6}$$
$$\phi = \arctan\left((\dot{\chi}V\cos\gamma)/(g\cos\gamma + V\dot{\dot{\gamma}})\right) \tag{7}$$
$$n = (g\cos\gamma + V\dot{\gamma})/(g\cos\phi) \tag{8}$$
$$T = D + m\dot{V} + mg\sin\gamma \tag{9}$$

The aerodynamic drag is given by [12]:

$$D = \tfrac{1}{2}\rho S C_D V^2 \tag{10}$$

where $C_D = C_{D0} + kC_L^2$ is the drag coefficient, $C_L = 2nmg/(\rho S V^2)$ is the lift coefficient, ρ is the air density, C_{D0} is the minimum drag coefficient of the aircraft and S is the wing area. It can be noticed from (4)–(10) that the inputs and the states of the system can be expressed as functions of the output vector and its derivatives, hence the system is differentially flat. So the optimal problem can be formulated in output space rather than control space. Thus, it is useful to find a sufficient description for the output space (trajectory profiles in our case) which makes the optimal problem more tractable.

III. Trajectory Description

NURBS curves are used to describe the trajectory profiles. A NURBS curve is a vector-valued piecewise rational polynomial function. The pth degree NURBS curve is given by:

$$P(\tau) = \sum_{i=0}^{n} R_{i,p}(\tau) C_i \tag{11}$$

$$R_{i,p}(\tau) = \frac{w_i N_{i,p}(\tau) C_i}{\sum_{i=0}^{n} w_i N_{i,p}(\tau)}; \quad a \le \tau \le b \tag{12}$$

where $R_{i,p}(\tau)$ are rational basis functions. The analytical properties of $R_{i,p}(\tau)$ determine the geometric behavior of curves [13], w_i are the weights, C_i are the control points, and $N_{i,p}(\tau)$ are the pth degree B-spline basis functions. There are many ways to represent B-spline basis functions, for computer implementation the recursive representation of B-spline basis functions is the most useful form [13]. Let $U = [u_0, u_1, \ldots, u_{m-1}, u_m]$ be a nondecreasing sequence of real numbers i.e, $u_i \le u_{i+1}$; $i = 0, 1, \ldots, m-1$, u_i called knots or breakpoints, and U is the knot vector that contain $m+1$ knots. So the ith B-spline basis function of p-degree (order $p+1$), denoted by $N_{p,i}(\tau)$ is defined as:

$$N_{i,0}(\tau) = \begin{cases} 1 & \text{if } u_i \le \tau < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$N_{i,p}(\tau) = \frac{\tau - u_i}{u_{i+p} - u_i} N_{i,p-1}(\tau) + \frac{u_{i+p+1} - \tau}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(\tau) \tag{14}$$

and $N_{i,p}(\tau) = 0$ if τ is outside $[u_i, u_{i+p+1}[$. The degree of the basis function p, number of control point $(n+1)$, and number of the knots $(m+1)$ are related by $m = n + p + 1$.

The knot vector can be realized in different forms, but it must be a nondecreasing sequence of real numbers. There are two types of knot vector, periodic and open, in two flavours, uniform and nonuniform [14]. In a uniform knot vector, individual knot values are evenly spaced. In practice, uniform knot vectors generally begin at zero and are incremented by 1 to some maximum value, or it can be normalized in a range between 0 and 1. A periodic uniform knot vector will give periodic uniform basis functions for which:

$$N_{i,p}(\tau) = N_{i-1,p}(\tau - 1) = N_{i+1,p}(\tau + 1) \tag{15}$$

Thus, each basis function is a translation of the other.

In an *open uniform* knot vector, the end knot values have multiplicity equal to the order of the B-spline basis functions $p+1$. NURBS basis functions have many useful properties [14]. For example, they are nonnegative, satisfy the portion of unity property, have a local support, remain in the convex hull of the control points, and all their derivatives exist in the interior of the knot span $[u_i, u_{i+p+1}[$, where they are rational functions with nonzero denominators. The recursive calculation of the NURBS basis functions makes them easily, efficiently, and accurately processed in a computer. In particular:

1) the computation of point and derivatives on the curves is efficient;
2) they are numerical insensitive to floating point rounding error, and
3) they require little memory for storage requirements.

A. Derivatives of B-Spline Curves

The derivatives of B-spline curves can be calculated simply by computing the derivatives of their B-spline basis functions. The kth derivative of $P(\tau)$, $P^{(k)}(\tau)$, is given by:

$$P^{(k)}(\tau) = \sum_{i=0}^{n} N_{i,p}^{(k)}(\tau) C_i \tag{16}$$

where $N_{i,p}^{(k)}(\tau)$ is the kth derivative of B-spline basis functions which can be calculated recursively:

$$N_{i,p}^{(k)}(\tau) = p\left(\frac{N_{i,p-1}^{(k-1)}(\tau)}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}(\tau)}{u_{i+p+1} - u_{i+1}}\right) \tag{17}$$

IV. Local Trajectory Optimization

The optimal local trajectory profiles can be achieved by finding values of design variables that minimize a defined cost function and satisfy all constraints.

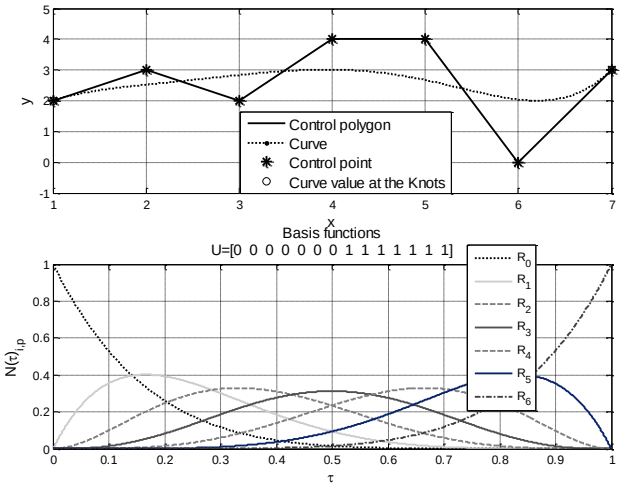

Fig. 1. Bezier curve (top), and its basis functions (bottom)

A. Bezier Curve

Bezier curves represent a special case of NURBS where all the weights are equal to unity, i.e. $w_i = 1$, and the knot vector is $U = [0,0,0,0,0,0,0,1,1,1,1,1,1,1]$ (for $p = 6$). In this case the basis functions are called Bernstein basis functions. A 6^{th} order Bezier curve ($p = 6$) has been used to represent the aircraft local trajectories. Using (12)–(14), the 6^{th} order Bezier curve basis functions are

$$\begin{aligned} R_0 &= (1-\tau)^6, & R_1 &= 6\tau(1-\tau)^5, \\ R_2 &= 15\tau^2(1-\tau)^4, & R_3 &= 20\tau^3(1-\tau)^3, \\ R_4 &= 15\tau^4(1-\tau)^2, & R_5 &= 6\tau^5(1-\tau), \qquad R_6 = \tau^6 \end{aligned} \tag{18}$$

Figure 1 shows a Bezier curve ($p = 6$), and its basis functions (Bernstein basis functions). It can be noticed that the first basis function, R_0, has a significant effect on the start point of the curve, R_6 controls the end point of the curve and the remainder of basis functions have no effect on the start and end points. This is one advantage of Bezier curves, and this property reduces the computational time during trajectory optimization. The trajectory shape will vary with variation of the coefficients C_i. The control point that are used in Bezier curve shaping in Figure 1 are $C = \{(1,2),(2,3),(3,2),(4,4),(5,4),(6,0),(7,3)\}$.

B. Trajectory Profiles Description

The speed profiles in forward (u), lateral (v), and vertical (w) axes can be written by using polynomial functions (6^{th} order Bezier function):

$$\begin{aligned} u(\tau) &= c_0^u R_0(\tau) + c_1^u R_1(\tau) + \cdots + c_6^u R_6(\tau) \\ v(\tau) &= c_0^v R_0(\tau) + c_1^v R_1(\tau) + \cdots + c_6^v R_6(\tau) \\ w(\tau) &= c_0^w R_0(\tau) + c_1^w R_1(\tau) + \cdots + c_6^w R_6(\tau) \end{aligned} \tag{19}$$

Using 6^{th} order polynomial functions to describe the speed profiles gives a good flexibility over the design horizon with acceptable number design variables (the polynomial coefficients) [15]. Calculation of acceleration, jerk, and position profiles can be done by taking the first derivative of (19) for the acceleration profiles, the second derivative of (19) for the jerk profiles, and integration for the position profiles. In order to do so, a relationship between the curve parameter τ and time t must be defined. A fixed time horizon (t_h) is used so t can be represented by $t = t_h.\tau$. Hence the acceleration profiles can be calculated:

$$\dot{u}(\tau) = \frac{1}{t_h}\left(c_0^u \frac{dR_0(\tau)}{d\tau} + \cdots + c_6^u \frac{dR_6(\tau)}{d\tau}\right) \tag{20}$$

and

$$\frac{d^2u}{dt^2} = \frac{1}{t_h^2} \cdot \frac{d^2u}{d\tau^2} \tag{21}$$

The acceleration and jerk profiles for lateral and vertical axis can be calculated in a similar way. The position profile is driven by integration of the basis function with respect to time t. This can be done by substituting $\tau = t/t_h$ in (18) and then integrating the basis functions with respect to time:

$$R_i^{int} = \int_0^{t_h} R_i(t)dt; \quad i = 0,1,\ldots,6 \tag{22}$$

The receding horizon trajectory profiles are discretized into n steps within the period $0 \le \tau \le 1$ to evaluate the cost function at each step during the optimization process. Discretized trajectory profiles can be calculated by discretizing the basis functions into n steps, so the resulted discrete basis functions can be written as matrices as follow:

$$\boldsymbol{R} = \begin{pmatrix} R_0(\tau_1) & \cdots & R_0(\tau_n) \\ \vdots & \ddots & \vdots \\ R_6(\tau_1) & \cdots & R_6(\tau_n) \end{pmatrix} \tag{23}$$

The same procedure can be applied to calculate $\boldsymbol{R}'$, $\boldsymbol{R}''$, and $\boldsymbol{R}^{int}$, all these matrices can be calculated off-line, hence the on-line trajectory profiles calculation is reduced to simple matrix multiplication:

$$\begin{aligned} u &= \boldsymbol{C}^{u^T}\boldsymbol{R}, & \dot{u} &= \frac{1}{t_h}\boldsymbol{C}^{u^T}\boldsymbol{R}', \\ \ddot{u} &= \frac{1}{t_h^2}\boldsymbol{C}^{u^T}\boldsymbol{R}'', & x &= x_0 + \boldsymbol{C}^{u^T}\boldsymbol{R}^{int} \end{aligned} \tag{24}$$

where $\boldsymbol{C}^{u^T}$ is the vector of coefficients for forward axis:

$$\boldsymbol{C}^{u^T} = [c_0^u \quad c_1^u \quad \cdots \quad c_6^u] \tag{25}$$

The trajectory profiles for the lateral and vertical axes can be similarly calculated.

By parameterizing the output profiles by the Bezier functions, the optimal control problem is converted into an optimization problem with the design variables being the polynomial coefficients. Hence there are 21 coefficients to be determined (seven for each axis).

C. Initial Conditions

The current aircraft state can be measured by a sensing unit and used as the initial boundary conditions that guarantee a smooth transition from the current state to the target state. Substituting $\tau = 0$ in the trajectory profile equations (19), (20), and (21) gives:

$$c_0^u = u_0, \quad c_1^u = \frac{t_h}{6}\dot{u}_0 + c_0^u, \quad c_2^u = \frac{t_h^2}{30}\ddot{u}_0 - c_0^u + 2c_1^u \tag{26}$$

where u_0 is the initial forward speed, $\dot{u}_0$ is the initial forward acceleration and $\ddot{u}_0$ is the initial forward jerk. Thus the first three coefficients for each trajectory profile can be determined and the number of design variables reduced from 21 to 12. In order to reduce the computational time of the optimization problem, the aircraft and the obstacles constraints have been augmented in the cost function by using a penalty function method.

D. Aircraft Constraints

In order to ensure that the resulted optimal trajectory will be achieved without exceeding the aircraft performance and control limits (i.e. ensure $\boldsymbol{U} \in \mathcal{U}$, $\boldsymbol{X} \in \mathcal{X}$), the cost function is augmented with additional penalty function terms. The Yukawa potential function [16] is used :

$$C^p = A_p \frac{e^{-\alpha_p d_p}}{d_p} \tag{27}$$

where C^p is the aircraft performance constraint term added to the total cost function, A_p is the scaling factor, α_p is the decay rate and d_p is the performance margin given by:

$$d_p(\%) = 100 - 100 \left(\frac{\text{current state value}}{\text{state max\textbackslash min value}} \right) \tag{28}$$

To avoid a zero value of d_p, a minimum performance margin value d_{min} must be defined so that:

$$\text{if } d_p \leq d_{min} \text{ then } d_p = d_{min}$$

It can be clearly seen from (27) that when the performance margin decreases (i.e. the current state value is close to its limit), the potential function takes a huge value. Thus the total cost function will increase significantly, so the search algorithm tries to find another solution that keeps the aircraft state away from its limits.

E. Obstacle Constraints

The collision avoidance constraint, $\boldsymbol{Y} \in \mathcal{Y}$, can be achieved by either including constraints on the optimization process or by augmenting the cost function with a penalty function. The latter is used here so that the total computation time of the optimization process is reduced. As for the performance constraints, the Yukawa potential function is used to punish the cost function if the aircraft approaches an obstacle:

$$C^{ob} = A_{ob} \frac{e^{-\alpha_{ob} d_{ob}}}{d_{ob}} \tag{29}$$

where C^{ob} is a penalty term that represents the obstacle constraints, A_{ob} is a scaling factor, α_{ob} is the decay rate and d_{ob} is the distance between the nearest point on the obstacle and the point of interest.

Although using potential functions to describe the obstacle constraints complicates the cost function, it simplifies the search algorithm in the optimization process. Another advantage of using a potential function is that it handles the collision event in a manner which is closer to human behaviour. For example, avoidance manoeuvres can vary according to many factors such as aircraft speed, obstacle speed, aircraft manoeuvrability, and obstacle manoeuvrability. Additionally, due to the difficulty in generating a full 3D illustration for the obstacles that are detected by the on-board sensor unit the potential function approach does not need a 3D description of an obstacle, it just needs the distance between the aircraft and the nearest point in the obstacle [17].

F. Total Cost Function

The following cost function is thus used for the optimization process:

$$J = \sum_{i=1}^{n} \left[\lambda_p J_i^p + \lambda_s J_i^s + \lambda_{prf} J_i^{prf} + \lambda_{ob} J_i^{ob} \right] + \lambda_t J^t \tag{30}$$

where
J_i^p is the position cost function:

$$J_i^p = (x_i^d - x_i^a)^2 + (y_i^d - y_i^a)^2 + (z_i^d - z_i^a)^2 \tag{31}$$

J_i^s is the speed cost function:

$$J_i^s = (u_i^d - u_i^a)^2 + (v_i^d - v_i^a)^2 + (w_i^d - w_i^a)^2 \tag{32}$$

J_i^{prf} is the vehicle constraints penalty function:

$$J_i^{prf} = \sum_{j=1}^{q} A_p \frac{e^{-\alpha_p d_p}}{d_p} \tag{33}$$

J_i^{ob} is the vehicle constraints penalty function:

$$J_i^{ob} = \sum_{j=1}^{m} A_{ob} \frac{e^{-\alpha_{ob} d_{ob}}}{d_{ob}} \tag{34}$$

and

$$J^t = \lambda_h (\psi_n^d - \psi_n^a)^2 + \lambda_f (\gamma_n^d - \gamma_n^a)^2 \tag{35}$$

where λ are scaling factors, n is the number of points that will be evaluated across the design horizon, q is the number of performance constraints, m is the number of detected obstacles, ψ is the heading angle and γ is the flight path angle. The superscript a means the actual value, while the superscript d means the demanded value.

It can be seen that the cost function given by (30) provides a balance between the different terms; trajectory tracking terms (J^p, J^s, J^t) and constraints terms (obstacle avoidance term J^{ob}, performance constraint term J^{prf}). This balance can be controlled by changing the scaling factors λ. The scaling factors can be constants or they may vary according to the situation, in other words the priority of the cost function terms can be varied in order to allow the aircraft to fly safely in different flight scenarios. By augmenting the constraints in the cost function the optimal problem will be solved as an unconstrained optimal problem, thus the computational time will reduced significantly.

G. Avoiding Local Minima

Using a gradient-based method to solve the optimal problem introduces the local minimum problem. The performance constraints tend to act as an enclosing boundary around the entire search space, hence are less likely to result in local minimum. Thus, the obstacle constraints are the primary source of the local minima. When obstacles are detected this can

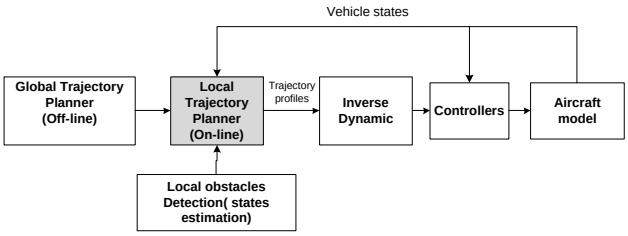

Fig. 2. System block diagram

have the impact of dividing the feasible design space into unconnected regions, therefore reducing the effectiveness of the solver of the optimal problem [17]. The possibility of getting trapped in local minimum is reduced by providing a mechanism for the search to jump to the different regions of the design space. This is achieved by generating a set of candidate trajectories then comparing the cost for each candidate then select the one that gives the minimum cost to initiate the optimal problem solver. The candidate trajectories are generated by applying maximum/minimum inputs to the vehicle model with the current vehicle states as initial states to ensure that the maximum performance manoeuvres in each axis are always available if required. In this case the input commands are:

$$\begin{aligned} \phi &= [\ \phi_{min} \quad \phi_c \quad \phi_{max}\ ] \\ T &= [\ T_{min} \quad T_c \quad T_{max}\ ] \\ n &= [\ n_{min} \quad n_c \quad n_{max}\ ] \end{aligned} \tag{36}$$

where ϕ_c, T_c, and n_c are the current values of the inputs, and $\phi_{min/max}$, $T_{min/max}$, and $n_{min/max}$ are the minimum and maximum values of the inputs which can be calculated from the vehicle specifications (the Aerosonde UAV [**?**] model and specifications are used here). This combination will produce $3^3 = 27$ candidate trajectories.

V. Simulation Results

This section demonstrates the method's effectiveness by showing simulation results of different scenarios. Figure 2 shows the system block diagram that is used in MATLAB/Simulink to produce the simulation results. For all scenarios, the global trajectory is level flight with constant speed $v = 30$ m.s^{-1} at 1000 m height, heading $\psi = 0$, the receding horizon time is $t_h = 100$ s and sampling time $t_s = 0.2$ s, the optimization process is updated every 10 seconds. The obstacle is represented as a sphere, and a 4D model of the moving obstacle is generated using a straight projection method [18], which assumes that the obstacle does not manoeuvre during the receding horizon time.

A. Trajectory Tracking and Pop-up Obstacle Avoidance

In this scenario the initial position of the UAV is higher than the global trajectory but with the same speed and direction. There is also a pop-up obstacle that the UAV must avoid. Figure 3 shows the simulation result of this scenario, it can be seen that the UAV is converging to the global trajectory then when the static obstacle appeared in its way, the UAV performed the necessary manoeuvre in order to avoid the obstacle. Then the UAV converged again to the global

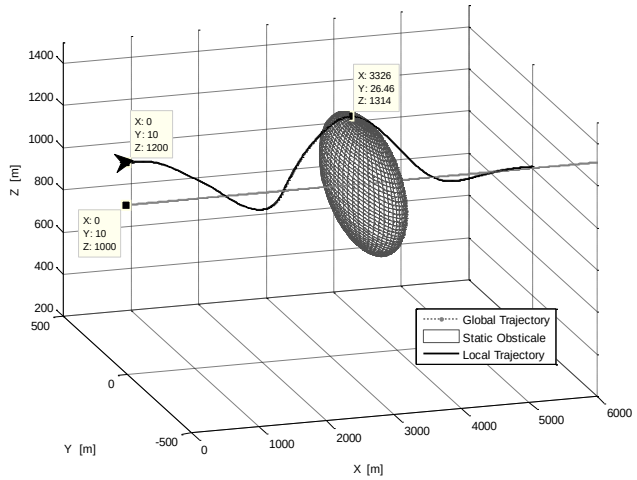

Fig. 3. Converging to the global trajectory and avoiding a pop-up obstacle

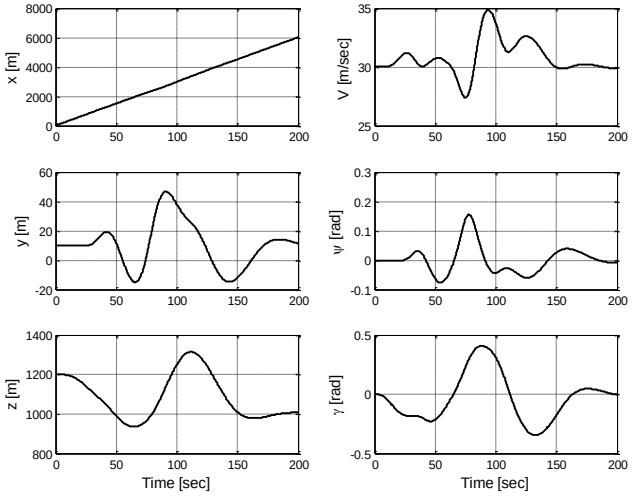

Fig. 4. Position, speed, heading, and flight path during the manoeuvre

trajectory after passing the static obstacle. Figure 4 shows time histories of some state variables of the UAV (position, speed, heading angle ψ, and flight path angle γ) during this scenario.

B. Global Trajectory Tracking with Two Moving Intruders

In this case the aircraft encounters two types of intruders so that there are two potential collisions, head-on and overtaking. The UAV has the following initial flight state (level flight at the initial position (0,10,1000) m heading $\psi = 0$, constant speed $v = 30$ m.s^{-1}). The first intruder (Intruder1) has the following state (level flight at initial position (2000,10,1000) m, heading $\psi = \pi$ rad, constant speed $v = 18$ m.s^{-1}). The second intruder (Intruder2) has the following initial state (level flight at initial position (2100,10,1000), heading $\psi = 0$ rad, constant speed $v = 15$ m.s^{-1}). So Intruder1 causes a head-on collision scenario, and then the UAV overtakes Intruder2. The protection zone around each intruder was chosen to be 200 m, so the distance between the UAV and the intruders should not become less than 200 m. Figure 5 shows the UAV trajectory during these scenarios. It can be seen that the UAV avoided both collision scenarios and returned to the global trajectory when it finished overtaking Intruder2. To clarify the

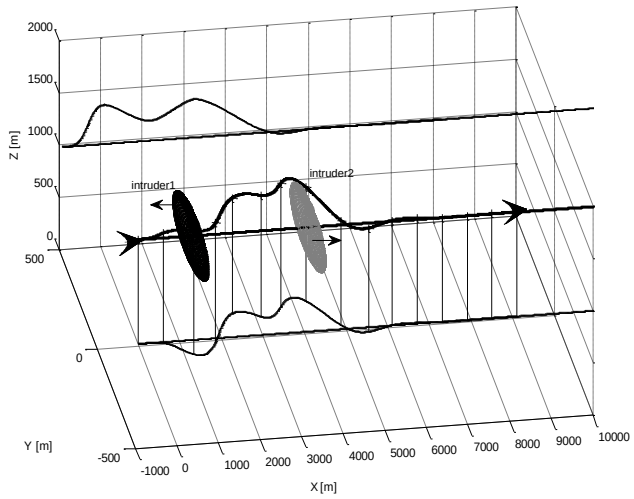

Fig. 5. Collision avoidance scenarios, head-on (intruder1), overtaking (intruder2)

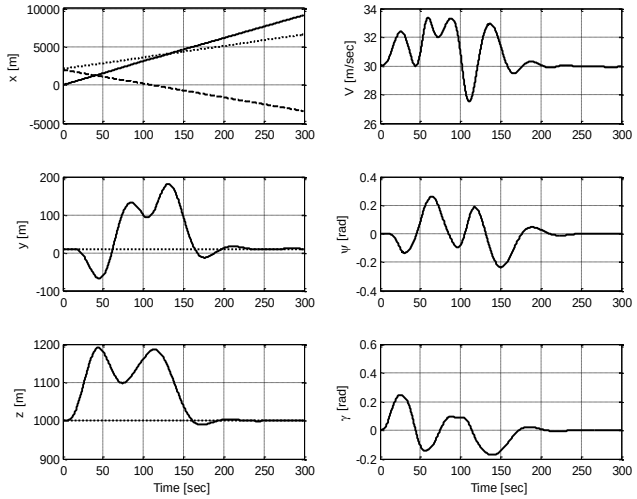

Fig. 6. Position, speed, heading, and flight path during the manoeuvre

performed manoeuvres, the projection of the UAV position on the horizonal and the vertical planes are included in Figure 5. The spheres that appear in Figure 5 represent the protection zones around the intruders when they and the UAV have the same position on the x-axis. Figure 6 gives the time histories of some UAV state variables (position, speed, heading angle ψ, and flight path angle γ) during these scenarios, and it also shows the position state of the intruders. The top-left subplot in Figure 6 shows the x distance time histories of the UAV (solid line), Intruder1 (dashed line), and Intruder2 (dotted line), while the other two subplots in the left column show the y and z distance time histories. It can be noted that when the UAV and one of the intruders have the same x distance, y and z will be at their maximum values, so the UAV is avoiding a conflict with the intruders.

VI. Conclusion

An optimal local trajectory generation by using B-spline is proposed for a real-time collision avoidance algorithm. Online avoidance maneuver generation, optimisation, and global trajectory tracking for different conflict scenarios are tested successfully in simulation environment (MATLAB/Simulink). Although the optimisation solver could be trapped in the local minima due to the obstacles existing, the coarse grid approach that is proposed in IV-G allows the solver to escape the local minima and ensure sufficient coverage of the overall design space. A computational time for the real-time collision avoidance algorithm is reduced significantly by using output space to formulate the optimal problem, and augmenting the vehicle/obstacle constraints in the cost function. The simulation results show that the proposed approach allows the UAV to track a predefined global trajectory as well as avoiding collisions with different types of conflict scenarios in real-time.

References

[1] C.-K. Lai, M. Lone, P. Thomas, J. Whidborne, and A. Cooke, "Onboard trajectory generation for collision avoidance in unmanned aerial vehicles," in *2011 IEEE Aerospace Conference*, march 2011, pp. 1–14.

[2] T. Hutchings, S. Jeffryes, and S. Farmer, "Architecting UAV sense and avoid systems," in *2007 IET Conference on Autonomous Systems*, Nov. 2007, pp. 1–8.

[3] T. Robinson. (2013, Jan.) Unlocking the skies. Aerospace Insight Blog. Royal Aeronautical Society. London, UK. [Online]. Available: http://media.aerosociety.com/aerospace-insight /2013/01/04/unlocking-the-skies/7633/

[4] I. Cowling, "Towards autonomy of a quadrotor uav," Ph.D. dissertation, Cranfield University, School of Engineering, 2008.

[5] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509 –521, Apr. 1998.

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, Mar. 1985, pp. 500–505.

[7] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, Sep. 1998.

[8] M. Y. Cho, A. J. Lichtenberg, and M. A. Lieberman, "Minimum stopping distance for linear control of an automatic car-following system," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 2, pp. 383–390, 1996.

[9] A. Bicchi and L. Pallottino, "On optimal cooperative conflict resolution for air traffic management systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 221–231, dec 2000.

[10] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "Direct method based control system for an autonomous quadrotor," *Journal of Intelligent & Robotic Systems*, vol. 60, pp. 285–316, 2010.

[11] H. Sira-Ramírez and S. K. Agrawal, *Differentially Flat Systems*. Marcel Dekker, 2004.

[12] M. V. Cook, *Flight Dynamics Principles*, 2nd ed. Elsevier, 2007.

[13] L. Piegl, "On NURBS: a survey," *Computer Graphics and Applications*, vol. 11, no. 1, pp. 55–71, jan. 1991.

[14] L. Piegl and W. Tiller, *The NURBS Book*. Springer Verlag, 1997.

[15] A. Berry, "Continuous local motion planning and control for unmanned vehicle operation within complex obstacle-rich environments," Ph.D. dissertation, University of Leicester, 2010.

[16] C. Cohen-Tannoudji, B. Diu, and F. Laloe, *Quantum Mechanics*. Wiley, 1977.

[17] A. Berry, J. Howitt, D. Gu, and I. Postlethwaite, "Continuous local motion planning & control for micro-air-vehicles in complex environments," in *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario, 2010.

[18] B. Albaker and N. Rahim, "Straight projection conflict detection and cooperative avoidance for autonomous unmanned aircraft systems," in *4th IEEE Conference on Industrial Electronics and Applications (ICIEA 2009)*, May 2009, pp. 1965–1969.