AUTONOMY FOR UAVS: INTEGRATION OF MULTIPLE TECHNOLOGY DOMAINS

Brain A. White* and Antonios Tsourdos*

Abstract

This paper looks at the science technology required to develop an autonomous group of UAVs to perform a specified mission. It details a three level approach which describes the bottom level where UAV guidance algorithms are examined. The second level describes the path planning requirements to allow for area research object tracking and rendezvous. The top level is where mission planning takes place. The use of Kripke Temporal Logic is described which allows the UAV behaviours to be determined and verified. A simple example of a UAV as part of a UAV group is given.

Introduction

In order to develop uninhabited aerial vehicles which have some degree of autonomy in flight, several different methodologies need to be taken into account. The UAV needs to have some decision making capability to perform tasks such as area searching, obstacle avoidance, object tracking and cooperative behaviour with other UAVs and mobile sensor platforms. To perform such tasks, map building, path planning, path following and obstacle avoidance strategies need to be developed.

UAVs can be used for the surveillance of specific areas, as well as for identifying and tracking objects of interest (OOIs). A recent example is the use of UAVs to search an urban area for objects such as people or vehicles that could pose a threat of some sort. Fig.1 shows such an area obtained from Google Map. The map shows that there exists a road network that connects a set of buildings as well as providing a means of ingress and egress from the area. In order to identify and track activity within the area, several UAVs are required to overfly the area at an altitude sufficient to acquire images of the required resolution to identify traffic and people.

Autonomous Systems Functionality

For the UAVs to operate effectively and autonomously, several functions are required. The most useful formulation is shown in Fig.2. The figure shows three levels of functionality. The top level deals with the decision making function. This produces allocation plans for the UAVs, allocating specific areas to be searched and objects to be tracked, as well as defining the hand-over conditions for group tracking and surveillance. The middle level defines the search areas for each UAV and produces paths that cover the area. These will also define approach and search geometry for surveillance and tracking. The bottom level shows individual control functions for the UAVs. Such functions include autopilot design for flight control, as well as guidance and navigation algorithms for following prescribed paths.

Path Planning

Path planning is a form of guidance in the context of trajectory shaping. The solution to the path planning problem involves two steps: design and planning. The steps are interlinked as the design of paths deal with the generation of the path between two locations, and the planning confirms that the path meets the constraints and requirements of the UAV mission. The path can be designed in two ways: by optimizing performance requirement of the system such as minimization of fuel-consumption, reducing the distance of travel, or by using standard paths.

The coordinated path planning for co-operating UAVs involves the design of individual and jointly feasible paths between an initial and terminal position and pose. Each path has to meet UAV kinematical constraints which bound the minimum turn Radius of Curvature and by the maximum and minimum velocity of UAVs and lastly by mission objectives. Typical mission objectives are simultaneous arrival, reconnaissance, search and track and rendezvous. Accomplishing the mission objectives together with the physical and functional constraints of the UAV kinematics increases the complexity of solution to path
planning problem [1], [2]. For the UAV path planning problem, the most common objective is to complete the given mission (i.e. to arrive at the given target within a pre-specified time) while maximizing the safety of the UAVs [8].

Meeting most objectives relies on continuous and smooth path planning for a group of UAVs with kinematical constraints. The simultaneous arrival of a group of UAVs flying at a constant speed and at a constant altitude in free space is one of the main mission objectives. Around each UAV, a safety-circle of radius, \( R_s \), greater than the minimum turning radius \( \kappa_{\max} \) of the UAV and smaller than the radius of sensor range \( R_{\tau} \) is assumed. As long as this safety-circle area is not violated, safe flight for UAVs is achieved.

In [9] the problem of simultaneous arrival is solved using Dubins paths. The approach in [9] satisfies the requirements of coordinated arrival, collision avoidance and minimum travel time for the group of UAVs. The solution to the problem presented in this paper is based on the use of differential geometry of curves.

The coordinated path planning is required to satisfy three constraints:

- Maximum bound on curvature,
- Paths of equal length
- Inter-collision avoidance between UAVs.

The curvature constraint limits the minimum turning radius and, in turn, the yaw-rate of UAV. The Dubins curve is optimized to meet maximum bound on curvature. The other two are derived constraints to meet the mission objective and are related to collision avoidance between members of the UAV team and coordinated arrival. These are achieved by imposing the conditions:

- Minimum separation between the UAVs and
- Non-intersection of paths at equal distance.

The Dubins path is made up of curves parameterized by simple elements, a Curve, a Line and a Circle (CLC), so that the path is not computationally intensive and easy to implement. The line is a curve of zero curvature and the circle is a curve of constant curvature. When these curves are concatenated, there is a discontinuity in curvature and although this phenomenon is undesirable it can serve as a basis for the generation of a smooth curve which does not have curvature discontinuities.

CLC paths can be designed by constructing the internal and external tangents between the circular arcs. These can be further refined by defining a right hand curve (R) or a left hand curve (L) connected by a straight segment (S). RSR and LSL paths are constructed by drawing the external tangent between the circular arcs. Fig.3 illustrates these paths.

The procedure for construction of (CLC) Dubins paths is given by defining:

- \( P_s(x_s, y_s, \theta_s) \) - Initial Configuration
- \( P_f(x_f, y_f, \theta_f) \) - Final Configuration
- \( \rho_s \) - Radius of curvature of at starting turn with center at \( O_s \)
- \( \rho_f \) - Radius of curvature of at finishing turn with center at \( O_f \)
- \( \rho_s \leq \rho_f \)
- \( T_{eN} \) - Tangent exit point on starting arc
- \( T_{eN} \) - Tangent entry point on ending arc
- \( T' \) - Tangent entry point on secondary circle

The procedure to generate an RSR path is as then follows:

1. Find the centers of the circles \( O_s(x_{cs}, y_{cs}) \) and \( O_f(x_{cf}, y_{cf}) \) using Eq. (1).

   \[
   x_{cs} = x_s \pm \rho_s \cos (\theta_s \pm \pi) \\
   y_{cs} = y_s \pm \rho_s \sin (\theta_s \pm \pi) \\
   x_{cf} = x_f \pm \rho_f \cos (\theta_f \pm \pi) \\
   y_{cf} = y_f \pm \rho_f \sin (\theta_f \pm \pi)
   \]

2. Connect the centres of the circles, \( C_s \) and \( C_f \). The Euclidean distance between centres is:

   \[
   c = \sqrt{(x_{cs} - x_{cf})^2 + (y_{cs} - y_{cf})^2}
   \]
3. Draw a circle of radius, $| \rho_s - \rho_f |$ with center, $O_f$. This circle is the secondary circle.

4. With $| O_s O_f |$ as a hypotenuse, draw a perpendicular such that $\Delta O_s O_f T'$ forms a right angle triangle. The other two sides are $O_f T' (= | \rho_s - \rho_f |)$ and $O_s T'$. The angles $\angle ( T' O_s O_f )$, $\angle ( XO_f T_E X )$ and $\angle ( XO_T T_E N )$ are respectively $\phi_e$, $\phi_{ex}$ and $\phi_{en}$. The angles, $\phi_{en}$ and $\phi_{ex}$ are equal.

5. The angle $\phi_e$ is given by:

$$\phi_e = \arcsin \left( \frac{\rho_f - \rho_s}{c} \right)$$  \hspace{1cm} (2)

where all the angles are positive in an anticlockwise direction.

6. Draw the line $T_E X T_E N$ parallel to $O_s T'$. This is the external tangent line between the two circles.

7. Connect the path through the points, $P_s$, $T_E X$, $T_E N$ and $P_f$.

8. The CLC-RSR path is formed by concatenation of the start-circular-arc $P_s T_E X$, followed by the external tangent $T_E X T_E N$ and the finish-circular-arc $T_E N P_f$.

9. A CLC-LSL path can be constructed in a similar way as per the above procedure.

One element in constructing a CLC path is the calculation of the tangent exit point $T_E X$ on the circle $C_s$, and the tangent entry point $T_E N$ on the circle $C_f$. Once these points are determined, the CLC paths using external tangent can easily be generated. Table-1 details the formulae to calculate the tangent exit and entry points for both RSR and LSL paths.

### Guidance

Once the Dubins path has been defined, the UAV is required to follow it. In order to do this a guidance algorithm is required. Consider a UAV following a Dubins path, as shown in Fig.4. In inertial axes, we have:

$$V_I = R(\theta) V_b$$  \hspace{1cm} (3)

where $V_b$ is the UAV speed, and $\theta$ is the body axes rotation angle wrt the inertial axes. Hence:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V_b = \begin{bmatrix} V_{b_x} \\ 0 \\ 0 \end{bmatrix}$$  \hspace{1cm} (4)

If a set of path axes are defined that are attached to the UAV via the normal vector $n_z$, then the UAV position in inertial axes is given by:

$$p_I = p_c + R(\phi) p_u$$  \hspace{1cm} (5)

where

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p_u = \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix}$$  \hspace{1cm} (6)

### Table-1 : Angle of Rotation to Calculate the Tangent Exit and Entry Points on the Circles $C_s$ and $C_f$

<table>
<thead>
<tr>
<th>Start-Turn</th>
<th>Finish-Turn</th>
<th>$\phi_e$</th>
<th>$\phi_{ex}$</th>
<th>$\phi_{en}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right</td>
<td>Right</td>
<td>$\arcsin \left( \frac{\rho_f - \rho_s}{c} \right)$</td>
<td>$\phi_e + \frac{\pi}{2}$</td>
<td>$\phi_e + \frac{\pi}{2}$</td>
</tr>
<tr>
<td>Left</td>
<td>Left</td>
<td>$\arcsin \left( \frac{\rho_f - \rho_s}{c} \right)$</td>
<td>$\phi_e - \frac{3\pi}{2}$</td>
<td>$\phi_e - \frac{3\pi}{2}$</td>
</tr>
</tbody>
</table>
Differentiating wrt time, and re-arranging yields:

\[
RV = \begin{bmatrix}
V_b \\
0 \\
0
\end{bmatrix} + R_\phi \begin{bmatrix}
0 \\
d \\
0
\end{bmatrix}
\]

(7)

where

\[
R_\phi = \begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(8)

and

\[
R_V = \begin{bmatrix}
\cos (\theta - \phi) & -\sin (\theta - \phi) & 0 \\
\sin (\theta - \phi) & \cos (\theta - \phi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(9)

This gives rise to

\[
V_c = V_b \cos (\theta - \phi) + \phi d
\]

(10)

\[
d = V_b \sin (\theta - \phi)
\]

(11)

Equation (10) gives the speed of the path axes origin along the path. Eq. (11) gives the dynamics of the distance \(d\) from the path.

Now, the rate of change of \(\phi\), the curvature of the path \(\kappa\) and the speed \(V_c\) of the point on the path \(p_c\) are related by

\[
\dot{\phi} = \kappa V_c
\]

(12)

Hence

\[
V_c = V_b \cos (\theta - \phi) + \kappa V_c d
\]

\[
(1 - \kappa d) V_c = V_b \cos (\theta - \phi)
\]

\[
V_c = \frac{V_b \cos (\theta - \phi)}{(1 - \kappa d)}
\]

(13)

This becomes indeterminant when the distance from the path coincides with the turn centre of the arc of the path. This occurs when

\[
(1 - \kappa d) = 0
\]

\[
d = \frac{1}{\kappa}
\]

(14)

This is a condition where \(V_c\) becomes indeterminate. For this condition, the UAV passes through the centre of rotation of the arc of the Dubins path. The usual form of guidance is to assume a carrot at a distance of \(c\) further along the path. This is shown in Fig.5. For a zero curvature path, we have

\[
\tan \psi = \frac{d}{c}
\]

(15)

where \(\psi\) is the angle between the path and a line drawn between the carrot point and the UAV body axes origin. In order to drive the UAV onto the path, the velocity vector of the UAV needs to be controlled onto the vector of the carrot point from the body. Hence consider a feedback algorithm such that

\[
\theta = -K (\psi + \dot{\theta})
\]

(16)

Bringing the equations together, yields, for small angular error

\[
\begin{bmatrix}
\dot{d} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix}
0 & V \\
-k c & -K
\end{bmatrix} \begin{bmatrix}
d \\
\theta
\end{bmatrix}
\]

(17)

The characteristic equation takes the form

\[
|sI - A| = s^2 + K s + \frac{KV}{c} = 0
\]

(18)

This is stable with the natural frequency and damping determined by the speed of the UAV \(V\), the carrot distance \(c\) and the loop gain \(K\).

**Decision Making**

Formal methods for the development of autonomous behaviour for a group of UAVs exist, and the most useful formulation is that of Kripke Temporal Logic. For exam-
ple, for path planning, the length of the path coupled with the UAV velocity gives an estimated arrival time at all points along the path. This can easily be expressed in Kripke Linear Temporal Logic format that allows reasoning about the future position and state of the UAVs. Formal methods use mathematically rigorous techniques to model or guide the specification, design and verification process of the system. Formal methods can thus be usefully used in the design process to reduce potential design defects compared to empirical techniques, such as simulation and manually testing. In the past decades, formal methods have become a proven approach for assuring correct operation of complex interacting systems and are widely applied in the software or hardware industry [17], [18] Formal methods normally comprise the following parts [19]:

- The specification, which is a mathematical model of the system’s behaviour or systems properties.
- The implementation model, which is a mathematical model which represents the design structure of the system
- Mathematical expressions, which state the relationships between the models.

At a basic level, the first item can be used to describe the specification of the system. According to IAEA [20], up to 70% of defects are introduced in the specification and design stage of development. So even if the development process informal or implemented manually, the system design can still benefit from a formal description to minimise defects. A formal description can also be used to guide further development as well and the developers can also use proofs to confirm properties of the system that are required or assumed to be true. A lot of formal specification languages were developed for design and implementation. Some of the most well-known formal methods include specification language for recording a systems properties. For example, the Z notation [22], [21] creates a clear specification for the system and formalize the proof of the correctness of the resulting program behaviour. ISO have standardized the Z notation. Another formal method is the Vienna Development Method (VDM), which was specifically designed for the development of software. It was developed by IBM’s Vienna Laboratory in the 1970s, based on the VDM Specification Language (VDM-SL) and supports the modeling of object-oriented and concurrent system. Yet another formal method is the Formal Development Methodology (FDM) Backus-Naur Form (BNF) which is popularly used in formally describing programming language syntax.

The FDM specification can be used as the basis of proving the properties of the specification once it is developed. This means that the specifications are well-formed statements in a mathematical logic domain. The verification steps are rigorous deductions in that logic. Proving the correctness of a system can then be done by human-directed proof or by automated proof. The human-directed proof requires good mathematical sophistication and expertise to produce a good proof. Hand proofs inevitably lead to human errors even for the simplest systems as well. For the automated proof process, it is undoubtedly true that checking the process by computer can reduce the possibility of error. On the other hand, there is a possibility that the program that does the checking may itself contain some errors, or the basic underlying axioms may be inconsistent.

Automated techniques can be divided into some general categories:

- Model Checking: This is a static analysis technique which provides reliable result through finding defects by systematically exploring all possible states of a model.
- Automated theorem proving: In contrast to model checking, this is not a static analysis technique. It uses computer assistance and bookkeeping, and determines the validity of the system by deduction in a logic calculus. Automated theorem proving can handle unbounded domains naturally, but human interaction is normally required in proving.

Currently, powerful verification systems, such as ACL2, which its authors have been improving since they created it more than 30 years ago, and HOL, the general interactive theorem proving system based on typed higher order logic have been used in industry.

Historically, model checking was developed in the context of temporal logic formulae for finite Kripke models, called temporal logic model checking. Temporal logic is a powerful tool allowing expression of a great number of fundamental system properties such as safety, liveness and precedence. Pnueli was the first to use temporal logic for reasoning about concurrency [23], [24] and proved program properties from a set of axioms that described the behavior of individual statements. The first model checker
may be EMC model checker [40]. More details about the discussion of model checking can be found in [45]. Normally, the temporal logic model checking can be described as follows:

- Let M be a state-transition graph.
- Let be the specification in temporal logic.
- Find all states s of M such that M, s → .

This is illustrated in Fig.6.

**UAV Behaviours**

Consider a UAV that is responsible for obtaining images of potential objects of interest within a specified areas such as:

- Areas around buildings
- Areas around openings and walls.
- Rough terrain that is impossible for other vehicles to search

The UAV can also act as a communication relay to and from any other vehicles and a Ground Station. It will act in this role in a passive manner in that the communication relay will not monitor the communications traffic but will simply relay communications traffic between the Ground Station and other UAVs.

Hence the UAV behaviour can be described as:

- Inertial Path Following. The UAV will follow a path defined by a set of waypoints using its GPS sensors. These waypoints are joined by straight lines or constant curvature segments. The MAV will keep a record of the current segment that the vehicle is traversing and the waypoint that it is traveling towards for sending back to the Ground Station.
- Data and Image Communication. The UAV will send back packages of information to the Ground Station. These will include: (i) Vehicle Status. The position of the vehicle, current waypoint and the orientation of camera (ii) Image Data. Images will be captured either at regular intervals along the straight segments of the path and at prescribed locations. All images will be sent back to the Ground Station, tagged with the image sensor GPS position and pose.
- UGV Communication Relay. The UAV will relay the communications traffic between the Ground Station and the other UAVs.
- Communication Safety Channel Loss. If the communication channel to the Ground Station is lost, then the UAV will automatically fly towards the Ground Station in accordance with the safety document.

The Kripke model shown in Fig.7 describes the worlds and the connections between worlds, which gives a formalization representation for the behaviour of the UAV.

\[ M = (W, R, L) \]

\[ W = W_1, W_2, W_3, W_4, W_5, W_6, W_7 \]

\[ R = r_1 (W_1, W_2), r_2 (W_3, W_2), r_3 (W_2, W_3), r_4 (W_6, W_3), r_5 (W_3, W_6), r_6 (W_5, W_2), r_7 (W_3, W_4), r_8 (W_4, W_3), r_9 (W_5, W_3), r_{10} (W_4, W_5), r_{11} (W_3, W_7), r_{12} (W_4, W_6), r_{13} (W_6, W_2) \] (19)

The worlds are defined as follows:

- W1 - Launch The MAV is launched.
- W2 - Path Following. The UAV download waypoints from the Ground Station.
- W3 - Waypoint following. The UAV is following the waypoints given by the Ground Station.
- W4 - Position Identification. The UAV gets its GPS position.
- W5 - Broadcasting Position. The UAV report its GPS position to the Ground Station.
- W6 - Sending Image Data. The UAV send the images it captures to the Ground Station.
- W7 - Landing. The UAV performs the landing manoeuvre.

The relationships between the world, which dictate the transition conditions are defined as follows:

- (W1 - W2): After the UAV is launched, it will follow waypoints sent from the Ground Station.
  - r1 true if (the MAV is successfully launched)
  - r1 false otherwise
- (W2 - W3) and (W3 - W2): The UAV communicates with the Ground Station in a fixed interval to download new waypoints from the Ground Station.
  - r2 true if it is time for the UAV to download waypoints
  - r2 false otherwise
  - r3 true if the UAV gets new waypoints to follow
  - r3 false otherwise
- (W3 - W6) and (W6 - W3): The UAV may send images to the Ground Station in a fixed interval and then return to waypoint following mode after sending images.
  - r4 true if the UAV has sent images to the Ground Station
  - r4 false otherwise
  - r5 true if it is the time for the UAV to send images to the Ground Station
  - r5 false otherwise
- (W3 - W4) and (W4 - W3): The UAV needs to identify its current GPS position in real time. After identifying its position, it will return to waypoint following mode if there still are waypoints need to be followed.
  - r7 true if the UAV identifies its current position
  - r7 false otherwise
  - r8 true if the UAV has identified its position waypoints to follow
  - r8 false otherwise
- (W5 - W2): The transition happens when the UAV needs to communicate with the Ground Station to download new waypoints from the Ground Station.
  - r6 true if it is the time for the UAV to download waypoints
  - r6 false otherwise
- (W5 - W3): The transition happens after the UAV has sent its current position to the Ground Station and there still are waypoints need to be followed.
  - r9 true if the UAV has sent position to the Ground Station waypoints to follow
  - false otherwise
- (W4 - W5): The transition happens after the UAV has identified its GPS position.
  - r10 true if the UAV has identified position send position
  - r10 false otherwise
- (W3 - W7): The transition happens if the Ground Station issues the recalling command.
  - r11 true if the UAV has received the recalling command from the Ground Station.
  - r11 false otherwise
- (W4 - W6): The transition happens if it is the time for sending images to the GCS.
  - r12 true if it is the time for the UAV to send images to the Ground Station
  - r12 false otherwise
- (W6 - W2): The transition happens when the UAV needs to communicate with the Ground Station to download new waypoints
  - r13 true if it is the time for the UAV to download waypoints
  - r13 false otherwise

**Conclusions**

In order to develop autonomous systems of UAVs, several techniques are required. Three techniques have been described in this paper, which result in a flexible and efficient framework for defining behaviours that allow for searching an area and identifying and tracking objects of interest. The main problems associated with such an approach are the definition of the interfaces between each technique. The interface between guidance and path planning is the curvature and distance from a Dubins trajectory. This enables a guidance algorithm to be able to track a trajectory and localise along that trajectory. The interface between the path planning and the mission planning is via initial and final positions and poses to enable area search and rendezvous.

One of the remaining problems is that of specifying the behaviours and relationships between such behaviours that will produce a suitable set of activities that a group of UAVs need to perform in order to fulfill any mission. Kripke Temporal Logic and associated model checking will verify that the behaviours and their relationships contain no deficiencies, but will not test whether the behaviours are adequate for purpose. Hence a good understanding of the problem is essential to produce correct behaviours. The Kripke approach as well suite to such a description, however, so minimises the possibility of incorrect specification.
Developments in guidance using dynamic inversion and the inclusion of aerodynamic functions is possible but more complex. This will result in more robust, invariant trajectory tracking.

There are other forms of path planning that use smoother trajectory definition, such as Pythagorian Hodographs. These have the advantage of continuous curvature, as opposed to the discontinuity in curvature associated with Dubins curves. However, Dubins curves are very simple to implement and result in the shortest path between any two positions and poses. They can be used to develop PH curves that are close to the Dubins solution, giving the best of both worlds.

It is true to say that there is more research required to provide a unifying framework to cover the total system performance. This should be able to deal with both differential equations to define the dynamics, as well as trajectory definition to define the kinematics and the behaviours in logic form.

References


17. Christopher A. Rouff and Michael G.Hinchey., "Using Formal Methods and Agent-oriented Software Engineering for Modeling NASA Swarm-based Sys-


Fig. 5 Carrot Guidance

Fig. 6 Model Checking Framework

Fig. 7 Kripke Model for UAV Behaviour