TG^{2}M: Trajectory Generator and Guidance Module for the Aerial Vehicle Control Language AVCL

A. Barrientos *. J. Colorado. *
P. Gutierrez. *

**Robotics and Cybernetics Group. Universidad Politécnica de Madrid, CP 28006 Spain (Tel: 913363061; e-mail: jcolorado@etsii.upm.es).*

Abstract: This paper presents a novel framework for high-level complex mission planning – AVCL and its built-in trajectory generator module – TG^{2}M. For the mission planning we present a language capable of describing the missions and capabilities of a heterogeneous group of vehicles, which includes its interpreter, a definition of a base-vehicle, and a Mission Planner (MP) that uses GIS as the data-model for the world. This MP is not tied to a particular set of vehicles, sensors or commands, which means that at any given time new functionality can be loaded and displayed to the human operator as new options and commands, allowing to control and display *N* mission at the same time. In addition for low-level mission guidance, the TG^{2}M addresses the feature of generating complex trajectories within mission-specific constraints, improving the typical civil system which use basic trajectory-generation algorithms, capable only of *linear* waypoint navigation, with little or non-existent control over the trajectory. Final experiments will test the TG^{2}M mathematical framework for trajectory generation showing the AVCL capabilities for the mission planning and control of the UAV.

1. INTRODUCTION

As the operational capabilities of UAVs are developed there is a perceived need for a significant increase in their level of autonomy, performance, reliability and integration with a controlled airspace full of manned vehicles (military and civilian). As a consequence researchers working with advanced UAVs have moved their focus from system modeling and low-level control to mission planning, supervision and collision avoidance, going from *vehicle* constraints to *mission* constraints (Barrientos *et al*., 2006). This mission-based approach is most useful for commercial applications where the vehicle must accomplish tasks with a high level of performance and maneuverability. These tasks require flexible and powerful trajectory-generation and guidance capabilities, features lacking in many of the current commercial UAS. For those reasons, we focus on two aspects: first of all, a state-of-the-art Mission Planner – MP capable of managing all the mission requirements and second, an embedded robust trajectory module generator that allows the UAV autonomously maneuvering during flight. If we check motion-planning methodologies from specialized literature (Herwitz, 2007, Alison *et al*., 2003, Frazzoli, 2002), all of them agree that a planning algorithm should provide feasible and flyable optimal trajectories that connect starting with target points, which should be compared and valued using specific criteria. These criteria are generally connected to the following major concerns, which arise during a plan generation procedure: feasibility and optimality.

The first concern asks for the production of a plan to safely *move* the UAV to its target state, without taking into account the quality of the produced plan. The second concern asks for the production of optimal, yet feasible, paths, with optimality defined in various ways according to the problem under consideration (LaValle, 2006). Even in simple problems searching for optimality is not a trivial task and in most cases results in excessive computation time, not always available in real-world applications. Therefore, in most cases we search for suboptimal or just feasible solutions. The simplest way to model an UAV path is by using straight-line segments that connect a number of waypoints, either in 2D or 3D space (Moitra *et al*., 2003, Zheng *et al.,* 2005). This approach takes into account the fact that in typical UAV missions the shortest paths tend to resemble straight lines that connect waypoints with starting and target points and the vertices of obstacle polygons. Although waypoints can be efficiently used for navigating a flying vehicle, straight-line segments connecting the corresponding waypoints cannot efficiently represent the real path that will be followed by the vehicle due to the own kinematics of the traced path. As a result, these simplified paths cannot be used for an accurate simulation of the movement of the UAV in an optimization procedure, unless a large number of waypoints are adopted. In that case the number of design variables in the optimization procedure explodes, along with the computation time. If we analyze the presented state-of-the-art UAV complex mission planners almost all of them share one important limitation: their software architecture is tightly coupled to one vehicle and the capabilities of its low-level controller. Civil applications require open and extendable software architectures capable of *talking* to vehicles from different suppliers. The AVCL addresses those limitations, allowing to model different vehicles into a single common

language (vehicle-independent missions). In the same fashion the described vehicles show that complex and simple maneuvers could be a suitable solution depending on the kind of mission to fulfill. For this reason the AVCL is extended with the TG^{2}M framework, capable of generating simple and complex 3D paths with the necessary vehicle constrains. Basically the TG^{2}M is a software tool capable of generating complex six-degrees-of-freedom trajectories in 3D space with velocity, acceleration, orientation and time constraints. The TG^{2}M is an extension module to the Aerial Vehicle Control Language (AVCL), a software architecture and interpreted language specification that address the issues of mission definition, testing, validation and supervision for UAVs (Barrientos *et al*., 2006). The AVCL platform incorporates a 3D visual simulator environment that uses a Geographic Information System (GIS) as the world's data model. The GIS backend means all objects are geo-referenced and that several official and commercial databases may be used for mission planning (roads, airports, power lines, crop fields, etc.). The language specification contains a wide set of instructions that allow the off/on-line supervision and the creation of vehicle-independent missions.

The section 2 of this paper introduces a brief description of the *AVCL architecture*, describing its components, modules, and the main features provisioned, addressing a novel way of human-mission planning definition and testing. The section 3 introduces the AVCL built-in *TG^{2}M framework*, describing the different techniques used for the trajectory planning and guidance of UAVs, as well as the mathematical treatment of these methods (analytical functions and polynomial interpolation). On the other hand, simulation-based results (see section 4) using a mini-helicopter simulator embedded into the AVCL environment will show the capabilities of the TG^{2}M while flying aggressive and simple maneuvers, comparing a new methodology for UAV guidance, which uses the Frenet-Serret formulas (Garret *et al.,* 1995) for setting smooth UAV 3D-orientation profile as a function of the velocity and acceleration of the vehicle during flight. Last but not least, *Final Observations* in section 5 includes comments about the TG^{2}M framework and some discussion about the methodology used.

2. THE AERIAL VEHICLE CONTROL LANGUAGE

The AVCL is not just a language capable of describing the missions and capabilities of an heterogeneous group of vehicles, it is part of a bigger framework that includes its interpreter, a definition of a base-vehicle, and a Mission Planner that uses GIS as the data-model for the world. The Mission Planner – MP (see Fig. 1) is not tied to a particular set of vehicles, sensors or commands. At any given time new functionality can be loaded and displayed to the human operator as new options and commands. This means that the MP tool is to be extended through Vehicle and Command Libraries without recompiling, and those new capabilities and better vehicles can be added easily. The Mission Planner is a great tool for simulation and direct comparison of various trajectory trackers, UAV models and controllers, because it can display *N* missions at the same time.

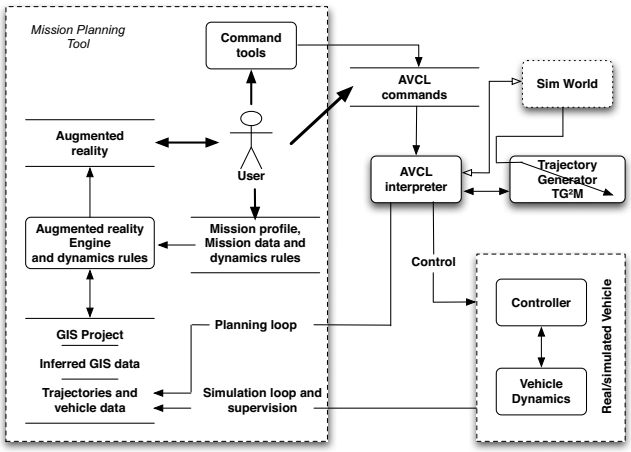

Fig. 1. The AVCL simplified diagram for mission planning.

When considered just as a language the AVCL concept is the abstraction layer that allows the human supervisor to create missions that are vehicle and payload independent, promoting code reuse. At the same time the AVCL statements and commands hide device specific software/hardware, and serve as mission definition and storage. As an example of the code used to define operations within a mission:

```
uav.Sensors(0) = parser.loadObject ('camera.lib')
uav.Sensors(0).LookAt (p1)
uav.Sensors(1) = parser.loadObject ('laser.lib')
uav.Sensors(1).TurnOn()
uav.doLine (way_points = {p1, p2, p3, p4}, vel = 0.9 m_s)
```

The AVCL and its interpreter provide several advantages: intuitive handling of different systems of units; its use of the object-oriented-programming paradigm; facilities for inter-vehicle communications; run-time definition of relations between vehicles, sensors and other equipment; it may be extended easily through C, C++ or C# code; the interpreter is a light-weight application written in C++, therefore it may be deployed in many SW/HW architectures. Before the development of the TG^{2}M module the AVCL framework relied on a simpler guidance module to connect waypoints with straight-line segments, and while the language could describe complex maneuvers and mission constraints the framework lacked the capacity to *fly* a vehicle through complex paths.

3. TRAJECTORY GENERATION AND GUIDANCE MODULE – TG^{2}M

The TG^{2}M is designed to model 3D cartesian paths with parametric constraints. The system uses two types of mathematical descriptors for trajectories: analytical functions and polynomial interpolation. The two main contributions of the module are its geometrical representation of the trajectory and its parametric definition. Simple maneuvers like lines and circumference arcs are created with analytical functions that constrain the geometry of the desired path; then the parametric constraints are *applied*. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. More complex

maneuvers are described with polynomial interpolation and fitted to the critical control path points, meeting desired position, time and velocity constraints. These polynomial functions are based on third and fourth order splines with fixed boundary conditions (for example initial and final velocities), which join all control points with a continuous and smooth path (Jaramillo-Botero *et al.,* 2004).

As shown in Fig. 2, two main modules compose the TG^{2}M framework: the geometrical trajectory generation and the online closed-loop guidance. The parameters and mandatory data (e.g. the desired maximum speed, percentage of the UAV acceleration during maneuvering, etc) are provided by the user via the AVCL interpreter.

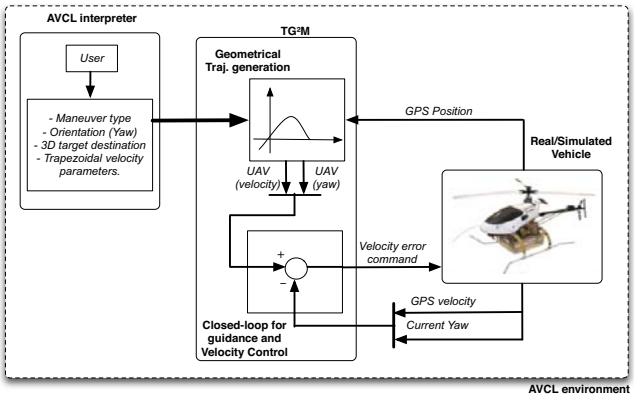

Fig. 2. The TG^{2}M framework.

3.1 The Geometrical Trajectory Generation module.

As mentioned before, the system uses two types of mathematical descriptors for trajectories: polynomial interpolation and analytical functions. For complex maneuvers, the 4D-spline interpolation will be used. The fundamental idea behind this spline interpolation is based on the definition of smooth curves through a number of points. These curves are represented by a polynomial function (in this case forth-grade) defined by some coefficients that determine the spline used to interpolate the numerical data points. These coefficients *bend* the line so that it passes through each of the data points without any erratic behavior or breaks in continuity.

$$S(t) = e_i t^4 + a_i t^3 + b_i t^2 + c_i t + d_i \tag{1}$$

Normal 3D-splines with fixed boundary conditions allow the definition of smooth curves across a number of knot-points. Nevertheless, two basic problems must be taken into account: the 3D spline only allows the user to establish the initial and the final velocities of the whole trajectory, limiting the user to have total control over the other points. The second problem is the smoothness of the acceleration curves (linear). To solve these problems, the 4D-splines address the user total control over the velocity profile across the whole trajectory and its results in additional *smoothness* for position, velocity and even acceleration curves. This could be more effective for UAVs tasks where the mission requires a strong control of the vehicle acceleration and velocities at each defined knot-point as shown in Fig 3.

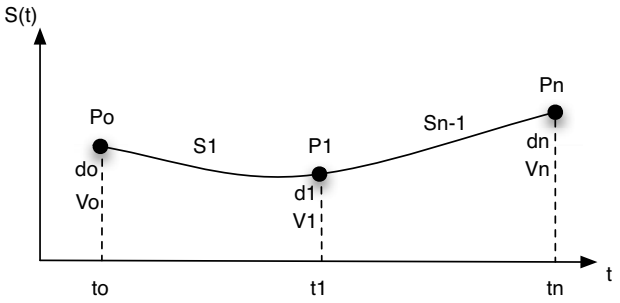

Fig. 3. Knot-points to interpolate with 4D-splines.

$S(t), S'(t)$ and $S''(t)$ will be continuous on the interval $[t_0, t_n]$. Since the curve $S(t)$ must be continuous across its entire interval, it can be concluded that each sub-function must joint at the data points, so: $s_i(t_i) = s_{i-1}(t_i)$.

$$A = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_1 & t_1^2 & t_1^3 & t_1^4 \\ 0 & 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 & t_2^4 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_1 & 3t_1^2 & 4t_1^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 & 4t_2^3 \\ 0 & 0 & 2 & 6t_1 & 12t_1^2 & 0 & 0 & -2 & -6t_1 & -12t_1^2 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 0 & 0 & -2 & -6t_2 & -12t_2^2 \end{bmatrix} \tag{2}$$

The polynomial coefficients are stacked into the $Y \in \Re^{5(n-1)}$ vector, and with the $f \in \Re^{5(n-1)}$ term, the total system is defined as:

$$Y = \begin{bmatrix} d_0 & c_0 & b_0 & a_0 & e_0 & d_1 & c_1 & b_1 & a_1 & e_1 \end{bmatrix}^T$$
$$f = \begin{bmatrix} f(t_0) & f(t_1) & f(t_1) & f(t_n) & V_0 & V_1 & V_1 & V_n & 0 & 0 \end{bmatrix}^T \tag{3}$$

To obtain a generalized solution of the system to solve from (2) and (3): ($A \cdot Y = f$) with $A \in \Re^{5(n-1)\,x\,5(n-1)}$, we start from the three-point case as depicted in Fig. 3. The polynomials for each trajectory segment as a function of time *t* are:

$$\begin{aligned} s_n(t_0) &= e_0 t_0^4 + a_0 t_0^3 + b_0 t_0^2 + c_0 t_0 + d_0 = f(t_0) \\ s_n(t_1) &= e_0 t_1^4 + a_0 t_1^3 + b_0 t_1^2 + c_0 t_1 + d_0 = f(t_1) \\ s_{n-1}(t_1) &= e_1 t_1^4 + a_1 t_1^3 + b_1 t_1^2 + c_1 t_1 + d_1 = f(t_1) \\ s_{n-1}(t_n) &= e_1 t_n^4 + a_1 t_n^3 + b_1 t_n^2 + c_1 t_n + d_1 = f(t_n) \end{aligned} \tag{4}$$

Taking the first and second derivatives from (4) (velocities and accelerations), we obtain:

$$
\begin{aligned}
s_1'(t_0) &= 4e_0t_0^3 + 3a_0t_0^2 + 2b_0t_0 + c_0 = V_0 \\
s_1'(t_1) &= 4e_0t_1^3 + 3a_0t_1^2 + 2b_0t_1 + c_0 = V_1 \\
s_{n-1}'(t_1) &= 4e_1t_1^3 + 3a_1t_1^2 + 2b_1t_1 + c_1 = V_1 \\
s_{n-1}'(t_n) &= 4e_1t_n^3 + 3a_1t_n^2 + 2b_1t_n + c_1 = V_n
\end{aligned} \tag{5}
$$

The second derivatives of (5) yield a set of accelerations. Equaling the acceleration functions for the intermediate points (t_1 for each case) and setting to zero the initial and final acceleration of the path segment yields:

$$
\begin{aligned}
s_0''(t_0) &= 12e_0t_0^2 + 6a_0t_0 + 2b_0 = s_{n-1}'(t_1) = 12e_1t_1^2 + 6a_1t_1 + 2b_1 \\
s_0''(t_1) &= 12e_0t_1^3 + 6a_0t_1 + 2b_0 = s_{n-1}''(t_n) = 12e_1t_n^2 + 6a_1t_n + 2b_1
\end{aligned} \tag{6}
$$

Complex maneuvers use (1) as shown in Fig. 3 to interpolate the desired knot-points that define the flight path. Likewise, for simple maneuvers generation, the TG^{2}M framework also supports the definition of straight-lines and circumferences via analytical functions that constrain the geometry of the desired path. These constraints are typically kinematic: constant velocity and acceleration, trapezoidal curves to accelerate or stop, etc. This kind of parameterization is useful when the mission requires the UAV stops at the desired end-point of the trajectory effectively, due to the user-control of the acceleration slope tilt level. For both (lines and circumferences) the desired set of velocities must fulfill the following trapezoidal velocity profile:

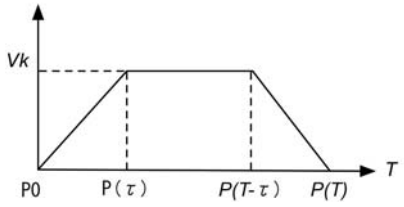

Fig. 4. Trapezoidal velocity profile used for simple UAV maneuvers.

The three-segment equation that compose the line-type motion is defined in (7) as:

$$
f(t) = \begin{cases}
\dfrac{V_k}{2\tau\left\|P(\tau) - P_0\right\|} t^2\left(P(\tau) - P_0\right) + P_0 & 0 \le t \le \tau \\
\dfrac{V_k}{\left\|P(T-\tau) - P(\tau)\right\|}(t-\tau) * \ldots & \\
\left(P(T-\tau) - P(\tau)\right) + P(\tau) & \tau < t \le T-\tau \\
\dfrac{(t-T+\tau)}{\left\|P(T) - P(T-\tau)\right\|}\left(-\dfrac{V_k}{2\tau}(t-T+\tau) + V_k\right) * \ldots & \\
\left(P(T) - P(T-\tau)\right) + P(T-\tau) & T-\tau < t \le T
\end{cases} \tag{7}
$$

In the first section (from P_0 to $P(\tau)$) the initial velocity is set $V_i = 0$ and progresses toward a final velocity V_k. The second segment (from $P(\tau)$ to $P(T-\tau)$) is traced at constant maximum velocity V_k. Finally the last segment (from $P(T-\tau)$ to $P(T)$) drives the vehicle from V_k to zero velocity. The two intermediate points of the trapezoidal curve: $P(\tau)$, $P(T-\tau)$ are calculated as:

$$
\begin{aligned}
P(\tau)_{x,y,z} &= \frac{1}{2\left\|P(T)_{x,y,z} - P_{0x,y,z}\right\|} \frac{V_k}{t} t^2\left(P(T)_{x,y,z} - P_{0x,y,z}\right) + P_{0x,y,z} \\
P(T-\tau)_{x,y,z} &= \frac{V_k(T-2t)}{\left\|P(T)_{x,y,z} - P_{0x,y,z}\right\|}\left(P(T)_{x,y,z} - P_{0x,y,z}\right) + P(\tau)_{x,y,z}
\end{aligned} \tag{8}
$$

The same approach is applied to generate circumferences with the trapezoidal profile used for the straight-lines. Three knot-points define the circumference path and the objective is to find the trace angle across the trajectory.

3.2 The Closed-loop Guidance module.

The geometrical representation of a UAV trajectory has been presented in the previous sub-section. Complex trajectories may be described using fourth-order degree splines, or simple maneuvers may be generated using common lines functions with some parametric features defined by the end-user.

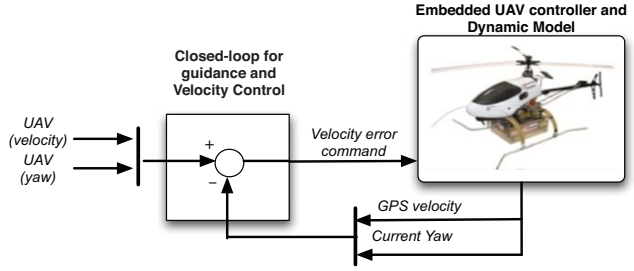

Fig. 5. The TG^{2}M guidance scheme.

From Fig. 5, the UAV velocity commands are generated using (5) and the derivative of (7) depending on the kind of trajectory to define. For the case of complex maneuvers, fixed-boundary constrained 4D-splines allow the generation of smooth paths with a certain set of accelerations. To extend this criterion to generate smooth UAV orientation commands (roll, pitch and yaw references), we introduce a new method based on the Frenet-Serret formulas for setting the UAV Euler angles as a function of the velocity and acceleration of the air-vehicle.

The built-in AVCL control module (see Fig. 1) is capable of receiving *velocity* and *yaw* angle orientation commands from the TG^{2}M module and generating the needed commands for the attitude controller that governs the vehicle's roll and pitch. The TG^{2}M's guidance module uses a simple PD controller for smooth *Euler* angle transition during the flight. For *roll, pitch* and *yaw* angles calculation, the following frames of references are used:

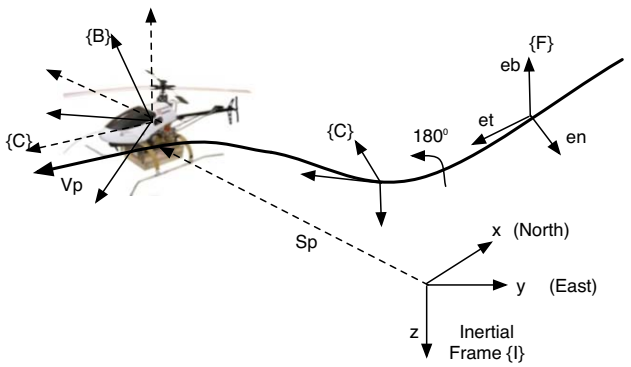

Fig. 6. The inertial, the Frenet-Serret, the vehicle and the curve Frames.

From Fig. 6, the following vectors can describe the motion of the UAV vehicle in 6 DOF:

$$\begin{aligned} \eta &= \left[\eta_1, \eta_2\right]^T = \left[x, y, z, \phi, \theta, \psi\right]^T \\ v &= \left[v_1, v_2\right]^T = \left[u, v, w, p, q, r\right]^T \end{aligned} \tag{6}$$

In (6), η_1 denotes the position of the center of mass CM of the vehicle and η_2 its orientation described by the *Euler angles* with respect to the inertial frame {I}. The vector v_1 refers to the linear velocity and v_2 to the angular velocity of vehicle frame {B} with respect to inertial frame {I}. In order to express the velocity quantities between both frames of references (from {B} to {I} and vice versa), the following transformation matrix is used.

$$\begin{aligned} \eta_1' &= R_B^I v_1 \\ \eta_1' &= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\theta s\phi \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} v_1 \end{aligned} \tag{7}$$

The body-fixed angular velocities and the rate of the Euler angles are related through:

$$\eta_2' = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{bmatrix} v_2 \tag{8}$$

The position and the magnitude of the velocity vector at a point *P* on the trajectory are given by:

$$\begin{aligned} S_p &= \left[x, y, z\right]^T \\ V_p &= \left\|S_p'\right\| = \sqrt{x'^2 + y'^2 + z'^2} \end{aligned} \tag{9}$$

The method to define the Euler angles is based on the Frenet-Serret theory (Angeles, 1997). To every point of the curve we can associate an orthonormal triad of vectors (a set of unit vectors that are mutually orthogonal) namely the tangent e_t, the normal e_n and the bio-normal e_b (see Fig. 6). The Frenet-Serret theory says that by properly arranging these vectors in a matrix $\in \Re^{3x3}$, we obtain a description of the curve orientation due to the position, velocity and acceleration of the UAV while tracing out the path. The unit vectors are then defined as:

$$e_t = \frac{S_p'}{V_p}, \quad e_b = \frac{\left(S_p' \times S_p''\right)}{\left\|S_p' \times S_p''\right\|}, \quad e_n = e_b \times e_t \tag{10}$$

In the definition of a frame associated with the curve the original definition of the Frenet-frame for counterclockwise rotating curves is used; in the case of a clockwise rotating curve, the *z*−axis of the Frenet-frame points in the opposite direction upwards than the inertial {I} frame. So in order to define small relative rotation angles for the orientation of a vehicle rotating clockwise and having its z_b axis pointing downwards, we define a reference frame associated with the curve as previously, but rotated with respect to the Frenet by an angle of 180 degrees about the *x*-axis of the Frenet frame (see Fig. 6). Collectively we denote the Frenet and the rotated frame as the "curve" frame {C}. we can express the coordinates of a vector given in the curve frame {C} to the {I} frame with the matrix:

$$\begin{aligned} R_C^I &= \left[e_t \quad e_n \quad e_b\right] \\ R_I^C &= R_C^{I\,T} \end{aligned} \tag{11}$$

For a counterclockwise rotation: $R_C^I = R_x\left(180^\circ\right)\left[e_t \quad e_n \quad e_b\right]$. Likewise, the rotation of the {B} frame from the {C} frame to the reference {R} frame can be expressed using customary aeronautical notation by considering the sideslip angle β and angle of attack α, (Siouris, 2004):

$$\beta = \sin^{-1}\left(\frac{v_R}{V_p}\right), \qquad \alpha = \tan^{-1}\left(\frac{w_R}{u_R}\right) \tag{12}$$

The vector v_R refers to the *y*-axis velocity component in the reference frame and w_R, u_R to the *z* and *x* - axis respectively. The overall rotation is composed by a rotation about body z_B axis through the angle β, followed by a rotation about the body y_B through the angle α, which is expressed as:

$$R_C^R = R_y^T\left(\alpha\right) R_z^T\left(-\beta\right) \tag{13}$$

Finally, the *roll, pitch* and *yaw* angles can be deduced as follows:

$$\begin{aligned} R_I^R &= R_C^R R_I^C \\ \phi &= \text{atan2}\left(r_{23}, r_{33}\right) \\ \theta &= \text{atan2}\left(-r_{13}, \sqrt{r_{23}^2 + r_{33}^2}\right) \\ \psi &= \text{atan2}\left(r_{12}, r_{11}\right) \end{aligned} \tag{14}$$

Where $r_{i,j}$ represent the components of the rotation matrix $R_I^R \in \Re^{3x3}$.

4. TG²M RESULTS

As shown in Fig. 1 the Mission Planner (MP) has two similar loops for mission planning and simulation/supervision. The difference is that in the Planning Loop the interpreter sends the projected waypoints back to the MP's Enhanced Reality, while in the Simulation Loop the interpreter commands the simulated vehicle, which in turn sends the simulated positions to the MP. Our research group has developed a Simulink-based model of a UAV helicopter named *Vampira*, which includes a position controller and is capable of real-time simulation. This simulator has been used with the Mission Planning and Simulation tool to test the TG²M. For Mission Supervision the AVCL commands would be sent to the real vehicle, and its position plotted alongside the projected and/or simulated paths. The *Vampira* helicopter was design and built within the framework of the project: "Guidance and Control of an Unmanned Aerial Vehicle" DPI 2003 01767 (VAMPIRA), granted by the Spain Ministry of Science and Technology, and it will be used for the real-world tests of the built-in TG²M framework. Figure 7 shows the AVCL user interface, which use GIS database for planning the mission within the real environment virtual representation. Once the user has planned the mission and established the necessary targets, requirements and constrains, the TG²M module computes the trajectory due to the specified set of knot-points. Once the trajectory is calculated, the built-in dynamics simulator (including the controllers) verifies that the trajectory is viable.

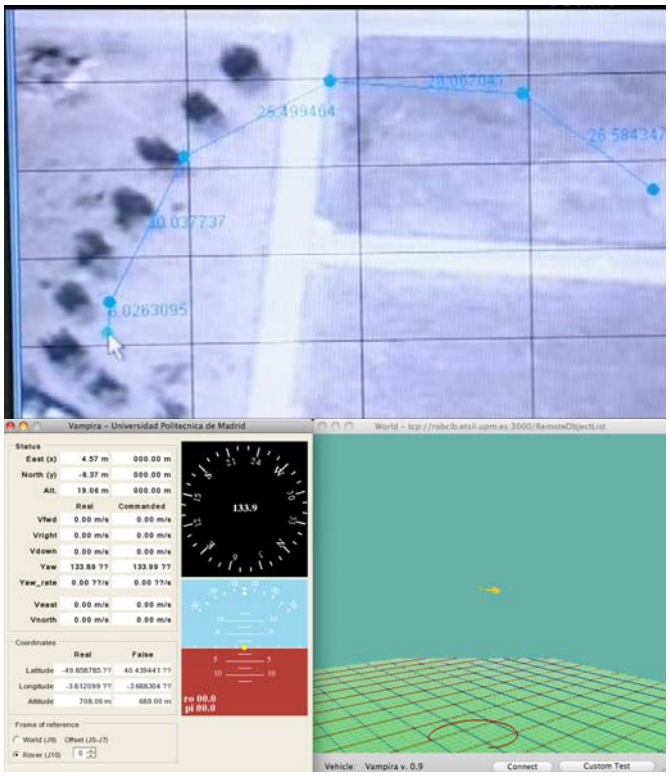

Fig. 7. The AVCL User Interface for mission planning and simulation.

In this case, we use the Vampira's UAV dynamics model, which has been obtained, identified and validated in previous works (Valero, 2005). The TG²M module has been coded using the C++ programming language, which is embedded into the AVCL, which has been coded using C++, C# and Java. However, the dynamics equations of motion and the controllers are embedded into a Simulink-Matlab block, which it is connected to the AVCL. Likewise, Figure 8 shows the *Vampira* prototype, which includes: a GPS, Wi-Fi link, IMU, and a PC104 computer for the low-level control (main rotor and tail servos). The Vampira's dynamics model has been obtained, identified and validated in previous works (Valero, 2005), (del Cerro *et al*., 2004).

Fig. 8. The Vampiras's UAV prototype.

Three test scenarios showcase the TG²M validation process. These tests involve the whole methodology previously presented in the other sections of this chapter, as well as the numerical simulation results using the AVCL environment and the embedded dynamics and control algorithms for the Vampira's helicopter. Two complex maneuvers are presented using 3D and 4D splines respectively and a simple last test using analytical function to generate a parameterized circumference motion.

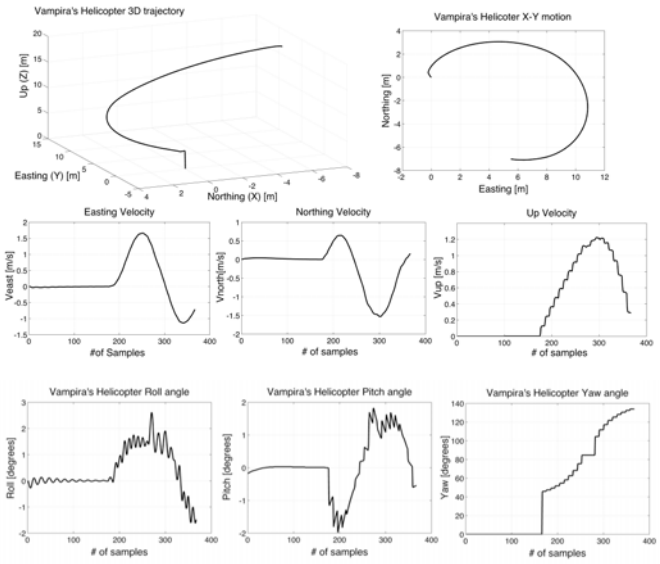

Fig. 8. Test #1: 3D-spiral flight using third-grade splines.

Figure 8 shows the test #1, "*Semi-spiral using 3D splines for the velocity profile generation and the Frenet theory for UAV orientation":* In this first test, we used a 3D-spline to joint three knot control points: (P_0(0, 0, 0), P_1(3, 5, 10), P_2(6, -7, 20)) at the desire time (given in seconds) for each point: (t(0,

10, 20)) and the desire initial and final speed (given in m/s): (V_0(0, 0, 0), V_f(0, -0.2, 0.4). The UAV started from initial point located at (0, 0, 0) coordinate and finished its trajectory at (6, -7, 20). Visual simulation depicted in Fig. 7 shows smooth motion across the trajectory due to the 3D-spline approach. Nonetheless, 3D-splines just allow the user to define the initial and final velocities of the motion, lacking of velocity control for the rest of the knot-points. To solve this problem, the following test (see Fig. 9) introduces a more complex trajectory generation using 4D-splines, addressing total user control of the UAV velocity profile.

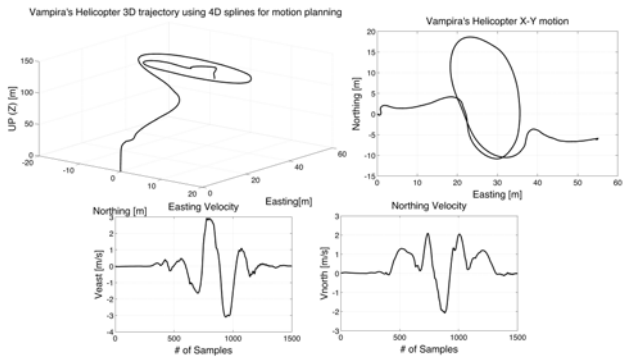

Fig. 9. Test #2: Complex motion using 4D-splines.

Figure 9 shows the test #2: *Complex trajectory using 4D splines for the velocity profile generation and the Frenet theory for UAV orientation*. This trajectory includes different kind of maneuvers joined into a single polynomial function (take-off, circumference-type motion and slow down in spiral-type motion). This test includes UAV long-endurance to high altitude (150 meters above ground) and a maximum easting displacement about of 60 meters. The following knot-control points (given in meters) have been defined: (P_0(0, 0, 0), P_1(0, 0, 20), P_2(0, 0, 40), P_3(0, 0, 60), P_4(10, 2, 80) , P_5(20, 4, 110) , P_6(25, -7, 130) , P_7(30, -10, 150), P_8(35, -5, 140) , P_9(30, 16, 125) , P_{10}(20, 5, 130) , P_{11}(33, -10, 145) , P_{12}(40, -5, 135) , P_{13}(55, -6, 125)). The advantage about using 4D-splines relies in the possibility of defining feasible paths that matches with the knot-control points defined (with less match error percentage than the 3D polynomial splines). In addition, the user is able to define the set of velocities for each of the knot-points during the motion. The set of velocities (given in m/s) are: (V_0(0, 0, 0), V_1(0, 0, 0.8), V_2(0, 0, 1), V_3(0, 0, 1.2), V_4(0.5, 1, 1.4) , V_5(0, 0.5, 1.7) , V_6(2, 1.5, 2.5), V_7(3, 2.2, 3), V_8(2, 1.2, 2) , V_9(1, 0.5, 1.5) , V_{10}(-0.5, -1, 0.8) , V_{11}(-3, -2, 0.4) , V_{12}(-1, -0.5, 0.8), V_{13}(0, 0, 0)).

Finally, Fig. 10 shows the test #3 "*Simple arc-type maneuver with trapezoidal velocity profile parameterization":* for the analytical AVCL feature of trajectory planning, the TG^{2}M module supports straight-lines and circumferences motions. An arc defined by: P_0(0, 0, 2), P_1(10, 5.5, 2), P_2(20, 0, 2)) with a maximum velocity of 0.5m/s is tested using the AVCL interpreter that allows the user to define the trapezoidal velocity profile configuration. For this case, the acceleration slopes of the curves have been set to the 30% of the total motion.

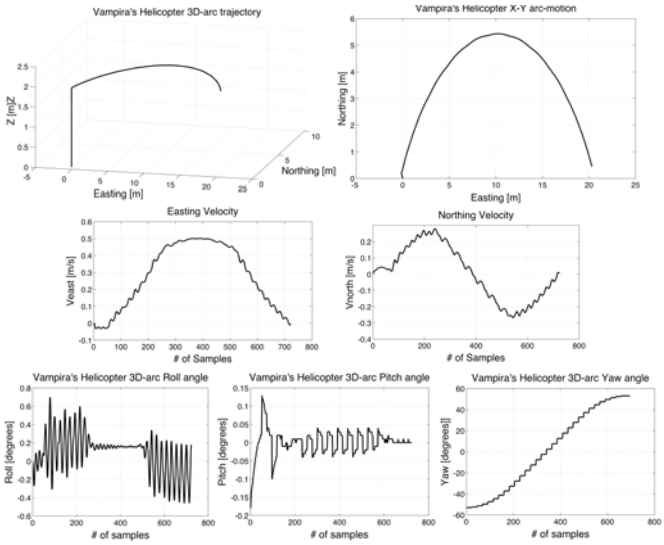

Fig. 10. Test #3: Simple Circumference maneuver.

5. FINAL OBSERVATIONS

For modeling continuous cartesian trajectories within the AVCL, analytical functions and polynomial interpolation methods are available; all of which can be used in any combination. The TG^{2}M module handles the definition of trajectories using knot control points as well as the incorporation of path constraints. It also supports the definition of complex tasks that require the construction of trajectories made up of different primitive paths in any combination of analytical and interpolated functions. The user-designed spatial trajectories can be visualized in three dimensions on the display window or plotted versus time using the embedded plotting library.

Simulation results have shown that the TG^{2}M module works perfectly for the definition and testing of wide kind of smooth trajectories, allowing the user a high-level control of the mission due to the AVCL interpreter. The three different scenarios used for testing, allowed verifying that the mathematical framework used for the trajectory generation and guidance was really working during flight. Percentage errors during maneuver execution were minimal, maintaining the UAV at the desired velocity limits and within the established path. We also incorporated velocity error fixing during flight. For high altitude tests, the velocity of the wind plays a mandatory role as a main disturbance external force. The TG^{2}M module includes wind perturbation compensation. The Guidance module fixes the velocity commands in real-time flight maneuver, decreasing the error position tracking.

On the other hand, the Frenet-Serret formulas included for the UAV orientation also presented a good approach in order to obtain smooth UAV rotation rate during flight. The use of simple trigonometric theory to obtain and define the UAV orientation profile (*Yaw angle*) is not convenient for complex maneuvers. Splines sometimes require a lot of know-points for feasible trajectory guidance, hence, using these polynomial equations, the Frenet approach allowed smooth angle changes between knot-points, which it had not been

obtained with the simple trigonometric angle calculation.

REFERENCES

Alison A. P., Bryan G., Suresh K. K., Adrian A. K., Henrik B. C. and Eric N. J. (2003). Ongoing Development of an Autonomous Aerial Reconnaissance System. *Georgia Tech UAV Laboratory, School of Aerospace Engineering*, Georgia Institute of Technology.

Angeles J., (1997). *Fundamentals of Robotic Mechanical Systems. Theory, Methods, and Algorithms*, Springer, New York.

Barrientos, A., Gutiérrez, P., del Cerro, J., San Martin, R. (2006). Mission Planning and Simulation of Unmanned Aerial Vehicles with a GIS-based Framework, *AIAA Guidance, Navigation and Control Conference and Exhibit*, Denver, EEUU.

Coifman, B., McCord, M., Mishalani, M., Redmill, K., (2004). Surface Transportation Surveillance from Unmanned Aerial Vehicles. *Proc. of the 83rd Annual Meeting of the Research Board.*

Frazzoli, E. (2002). Maneuver-based motion planning and coordination for single and multiple UAVs. *AIAA's 1st technical conference and workshop on unmanned aerospace vehicles*, University of Illinois at Urbana-Champaign, il 61801. S. Portsmouth, Virginia.

Garret J. Etgen. John Wiley & Sons. (1995) *Salas and Hille's Calculus -- One and Several Variables,* Seventh Edition. p. 896.

Held, Jason M; Brown, Shaun and Sukkarieh, Salah. (2005) Systems for Planetar y Exploration. *I5th NSSA Australian Space Scienc e Conference*, pp. 212-222, ISBN: 0864593740. RMIT University, Melbourne, Australia.

Herwitz, S. (2007). Developing Military COEs UAV applications, *UAV Applications Center – NASA Ames Research Center*, Moffett Field, CA.

Jaramillo-Botero A., Correa J. F., and Osorio I.J., (2004) Trajectory planning in ROBOMOSP, *Robotics and Automation Group GAR*, Tech. Rep. GAR-TR, Univ. Javeriana, Cali, Colombia.

LaValle, S.M. (2006). *Planning Algorithms*, Cambridge University Press.

Moitra, A., Mattheyses, R.M., Hoebel, L.J., Szczerba, R.J., Yamrom, B. (2003). Multivehicle 5 reconnaissance route and sensor planning, *IEEE Transactions on Aerospace and Electronic Systems*, p.37.

Price, I. C, (2006). Evolving self organizing behaviour for homogeneous and heterogeneous swarms of UAVs and UCAVS, *PhD Thesis, graduate School of Engineering and Management, Air Force Institute of Technology (AU),* Wright-Patterson AFB, OH, March.

Rysdyk, R. (2003). UAV path following for constant line-of-sight, *In 2th AIAA "Unmanned Unlimited" Conf. and Workshop and Exhibit, San Diego, CA.*

Siouris, G. M. (2004). *Missile Guidance and Control Systems*. Springer, New York.

Valero, J. (2005). Modelo dinámico y sistema de supervisión de vuelo de un helicóptero autónomo, *Proyecto de fin de carrera. ETSIIM-UPM.*

Zheng, C., Li, L., Xu, F., Sun, F., Ding, M. (2005). Evolutionary Route Planner for Unmanned Air Vehicles, *IEEE Transactions on Robotics,* **21**- 609–620**.**