BUILDING MULTI-UAV SIMULATION METHODS

Daniel Lluch
Senior Applications Engineer, Member AIAA

The MathWorks
Natick, Massachusetts

1 Abstract

The objective of this paper is to show a complete simulation of multiple Unmanned Air Vehicles (UAVs), with emphasis on the requirements needed and the benefits realized in making the simulation functional for an arbitrary number of vehicles. Cooperative control will be shown for an autonomous set of UAV in performance of some task, such as formation, reconnaissance, surveillance, search, jamming, decoy, or target attack.

The simulation is developed using a Commercial Off-the-Shelf (COTS) software package that allows for a hierarchical block diagram representation to include the control laws and vehicle models. The tools also allow for the use of finite state machines to be used for control law modes. Visualization is achieved by having the simulation drive a VRML (Virtual Reality Modeling Language) virtual world allowing the interactions of the vehicles in their environment to be seen as the simulation is running. The simulation also will allow for replacement of certain components; such as the control scheme that is used or the vehicles that are modeled.

The basis of this simulation comes from previous research performed in the study of aircraft formation flight in the context of cooperative game theory and the natural aggregate motion of flocking birds, schooling fish, and herding of land animals.

2 Introduction

A multiple vehicle simulation showing autonomous control will be made using COTS software tools. This simulation will show the benefits of making a graphical hierarchical model. The benefits will include ease of communication of the model and its components. It will also add such features as state machines and VRML visualization.

Multiple UAV sorties have a wide variety of applications, both military and commercial. There has been a fair amount of research over recent years in this area to further these efforts in such applications as reconnaissance, search and rescue, forest fire inspection, traffic reporting, and rapidly deployable communications networks among others. Many of these applications are concentrating on UAVs that fall into the category of Micro Air Vehicles (MAVs) [1]. Two major areas of research are vehicle construction [2] and collaborative/distributed control algorithm design [3]. Integrating these research areas will require testing the control algorithms on prototype vehicles. The dynamic capabilities of the vehicles will certainly affect algorithm performance. It will be necessary to have a quick way of implementing algorithms based on vehicle selection as well as on the number of vehicles. This effort will entail building models that exercise the algorithms on a variety of vehicles types and numbers, in order to establish success criteria or guidelines in mission tasks.

Graphical dynamic modeling software tools allow for certain advantages over compiled languages as well as scripting/interpretive languages. Although each approach has its strengths and weaknesses, the user's needs can dictate the appropriate path to take and in general a combination of all the above can lead to the best compromise between execution speed, ease of construction, flexibility in making changes, and ease of communication and sharing with others. Simulink is one such graphical block diagram hierarchical modeling tool. This example shows how to model a system with an arbitrary number of states. This translates into a simulation model that has the capability for handling varying signal widths, a state machine with an arbitrary number of states, and a VRML world with an arbitrary number of vehicles. The following secions specify the modeling approach that lead to the successful implementation of these goals.

3 Background

Existing research in the area of collaborative control of autonomous vehicles is used as a starting point. The existing work was

used as a base to make a model of the multiple UAVs using rules similar to that in collaborative game theory [3]. This work was originally done using only MATLAB scripts taking advantage of the interpretive aspects and matrix capabilities of the language. Some of the benefits of moving from a textual/scripting environment to a graphical environment will be shown. These benefits include, among other things, a hierarchical representation, state machines, and VRML visualization.

The vehicles are allowed to move individually yet each aircraft's motion is based on the positions and motions of all aircraft, i.e. collaborative control. This is the type of group dynamics exhibited in schooling fish, flocking birds, and herding land animals. The decisions that dictate each of the vehicle's motions are based on instincts for each vehicle. The instincts used in the previous work and reproduced here are

- Collision avoidance
- Obstacle avoidance
- Target acquisition
- Formation keeping

These instincts all weigh in to dictate what each vehicle will be commanded to do. By examining the maximum weight of the instincts the vehicle can be said to be in one of four modes, namely one of the four possible instincts. It is important to note that all instincts come into play when each vehicle decides how to maneuver; yet only one instinct will dominate at any particular time. For more information on the instincts and the calculations involving them refer to [3]. The vehicles that were used in the original research are similar to typical modern fighter jet aircraft.

As mentioned earlier, the original research was accomplished using MATLAB only. The simulation existed in a few scripts that modeled five vehicles through two example maneuvers. The first example was target acquisition and the second was obstacle avoidance. Both of these examples are replicated with the simulation made here, but in addition, other examples will be included, such as changing the number of vehicles available for a certain task. Methods will be shown on what can be done to make the process of changing these things easier.

Some of the benefits from migrating to the new simulation environment are:

- The use of a large variety of integration solvers and being able to change between them easily.
- Graphical block diagram model allows for ease of communication of model hierarchy, and lets the diagram be the documentation as well as the executable specification.
- Adding a state machine, which can greatly assist in the mode logic that can occur and will be needed in a multi UAV environment.
- Easily driving a VRML world while the simulation was executing which greatly enhances user feedback on vehicle motions and dynamics.

4 The Simulation

The simulation is broken into 3 major parts/subsystems.

1. The vehicle(s), which takes the commands as inputs and then outputs the state of each vehicle.
2. The instincts, which takes the states of all vehicles as inputs and then outputs the desired states for each vehicle based on the instinct calculations.
3. The control law, which takes the desired and actual states and computes the commands to drive each vehicle to the desired state.

The commands generated by the control law are then the inputs into the vehicles, thus forming an autonomous loop closure that constitutes the multi UAV simulation model, Figure 1.

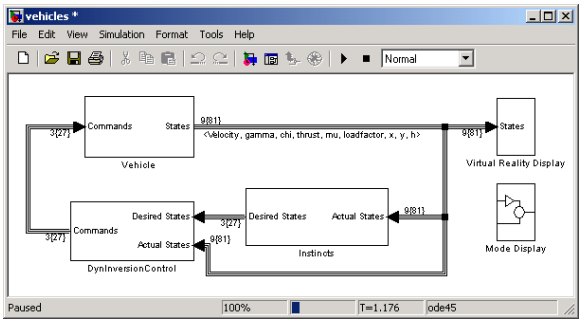

Figure 1: highest level of simulation

There are two other blocks on the highest level which are used for mode visualization using a state machine and for VRML visualization of the whole formation.

4.1 Mode Processing

How the vehicles process the instinct information is of relevance in understanding vehicle behavior. The scaling factors for each instinct are shown in Figure 2, which is how the information was presented in the previous research. These plots show the time histories of the weighting factors of all four instincts for each aircraft in the previous research. The time trace with largest magnitude is the instinct that is dominant at that time.

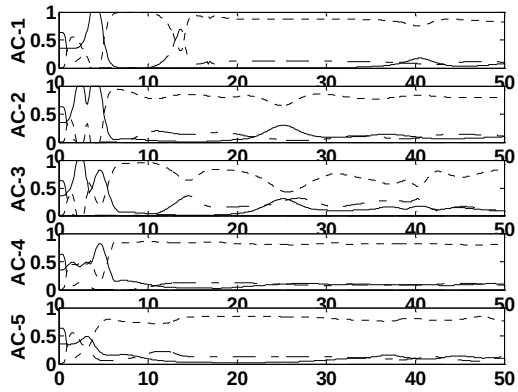

Figure 2: Instinct scale factors

This same type of information is also shown in a state machine, Figure 3. The state machine shows which instinct has maximum effect among all instincts for each vehicle at any given time. This chart can be animated during simulation and highlights the mode transitions for all aircraft as they occur.

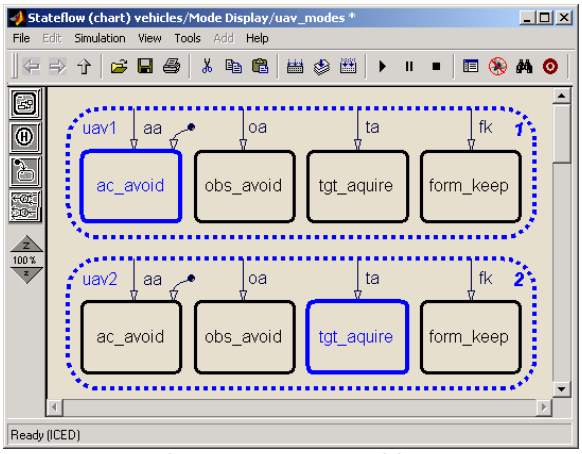

Figure 3: State Machine

This is useful for quickly understanding the interactions of the vehicles throughout the simulation. For example, it can be seen when particular aircraft are most concerned on not hitting their neighbor because the collision avoidance instinct has become dominant.

The use of a state machine allows for an environment where supervisory control can easily be implemented. One could enforce behavior to occur when vehicles are in a particular mode. For instance, enforce that only one vehicle can lock into a particular target at any given time or when a particular vehicle does lock onto a target perhaps turn off some other instincts such as formation keeping. This would allow for certain vehicles to leave a given sortie for possible reconnaissance/target flyby. As another example of adding conditionally executed statements based on mode, counters will be added on each aircraft that will keep track of what modes have been entered and how many times. This will be used in a later example.

4.2 Visualization

The VRML visualization subsystem allows for a virtual world to be driven by the simulation. All vehicles are represented in the VRML world with their position and orientation being driven by the simulation. This is extremely useful in being able to visualize the formation as a whole and get immediate user feedback as to how the vehicles are moving with respect to each other or obstacles/targets in their path. Figure 4 shows a side view of five vehicles approaching two obstacles.

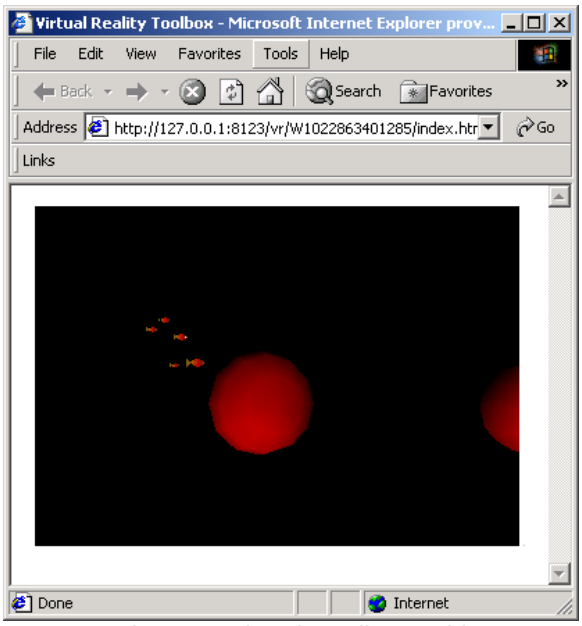

Figure 4: Virtual Reality World

It is this virtual reality world that allows choosing which vehicle to follow, or choose between different views such as side, rear, and top. This is in contrast to analyzing sets of time histories in order to reconstruct motions for each vehicle and how they moved relative to each other or to their environment.

4.3 Arbitrary Number of Vehicles Handling

As mentioned, one of the primary goals was to create this simulation with the capability of modeling an arbitrary number of vehicles. Certain modeling techniques were used in order to accomplish that. On a global sense, this translates to the ability of being able to make calculations that are independent or parameterized on signal dimensions within the block diagram. An example of this is the initial condition for each vehicle. The state vector initial condition is parameterized on the number of vehicles and the signal width for the one integrator in the model adjusts itself accordingly. This approach is cost effective in simulation execution time because all signal width checking and propagation are only called once, prior to simulation start.

The need for an arbitrary number of vehicles also meant an arbitrary number of states in the state machine and vehicles in the VRML world. Both the state machine and the VR world are generated automatically via respective Application Program Interfaces (APIs) using the MATLAB scripting language as an initialization stage to running the simulation. This is an example of how the scripting/interpretive language compliments the graphical modeling paradigm. These abilities allowed for the simulation to be parameterized on the number of vehicles, the number of targets, and the number of obstacles.

Another key feature that was utilized in this simulation was the ability to handle matrix signals. Throughout the model matrix signals exist and need to be concatenated or elements extracted from. Matrix math is critical in being able to keep arbitrary numbers of vehicles, targets, or obstacles and allows for single algorithm implementations to be valid for arbitrary signal dimensions.

There are quite a number of options the user may change in a simulation such as this. Previous results modeled include the target/obstacle positions and radii or whether obstacle avoidance or target acquisition instincts/modes are turned on. New options include the number of vehicles to be simulated, along with number of targets and/or obstacles. MATLAB has the capability to make custom graphical user interfaces (GUI). One was made that served as the front end to this simulation and allowed the user to see and change all the aforementioned options, Figure 5. This GUI is completely customizable and can be quite useful in presenting the user with pertinent information and a single front end in which to control the simulation and make changes to it. From this GUI, the proper timing can be enforced and automated in the execution of the appropriate Stateflow and VRML API calls, the two pieces that need to change as a function of the user entered parameters in the GUI. It is important to note that the main simulation blocks consisting of the vehicles, instincts, and controller, do not change at all with the above mentioned parameters.

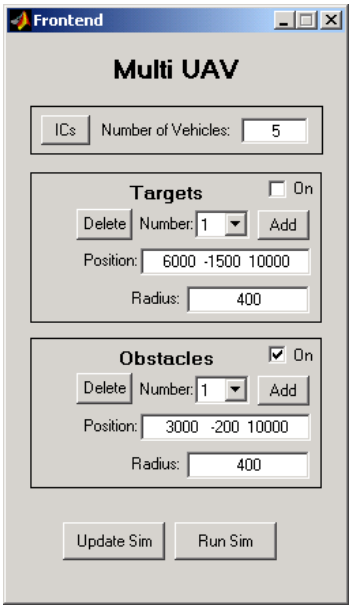

Figure 5: Frontend Control GUI

4.4 Other Benefits

Migrating the simulation to Simulink also resulted in a much faster execution time. The exact same target acquisition test case as in the first example from Reference [3] is used, including Euler integration at a .1 second time step. This resulted in an 84% reduction in simulation execution time from 5.3 seconds to .84 seconds. But even more importantly, there is now an easy way to choose integration solvers. With Simulink, the solver can immediately be changed to a variable step

solver which would shrink it's time steps when the dynamics of the system get faster (acquiring of targets), and larger time steps for the slower dynamics (formation flight). A variable step Domand-Prince ode45 solver showed an execution time of 4.7 seconds. This was also still faster than the original 5.3 seconds and shows that the dynamics of the problem dictated smaller time steps at some part of the simulation run and a more correct solution was obtained over using .1 second fixed time step Euler integration.

5 Trade-off study: Number of vehicles

With the existing simulation environment and the added functionality, the effect of number of vehicles will be shown on the instincts and the formation dynamics. Given a set of obstacles, the number of vehicles will be increased for a given sortie. The results are summarized in the following simulation outputs. The variable `mode_count` is a [vehicles x instincts] matrix whose values are how many times the particular instinct was entered by that vehicle. The instincts are repeated here in columnwise order

- Collision avoidance
- Obstacle avoidance
- Target acquisition
- Formation keeping

For 3 vehicles:

```
mode_count =
     1     0     1     0
     2     1     2     0
     2     1     2     0
```

For 5 vehicles:

```
mode_count =
     3     0     2     0
     4     1     4     0
     3     2     4     0
     2     1     3     0
     3     1     3     0
```

For 7 vehicles:

```
mode_count =
     2     0     1     0
     4     1     4     0
     4     0     3     0
     4     0     3     0
     4     0     4     0
     5     0     4     0
     3     1     3     0
```

For 9 vehicles:

```
mode_count =
     3     0     2     0
     4     0     3     0
     5     1     5     0
     3     0     3     0
     4     0     3     0
     5     0     4     0
     7     1     7     0
     5     0     4     0
     3     0     2     0
```

The obstacles for each run were fixed at the following positions:

```
>> obstacles.positions
ans =
        3000         -200        10000
        4500          800        10000
        3500          200        11000
>> obstacles.radius
ans =
   400
   300
   200
```

But this may more easily be visualized by the VR world rear view snapshot taken near the beginning of the nine vehicle case, Figure 6.

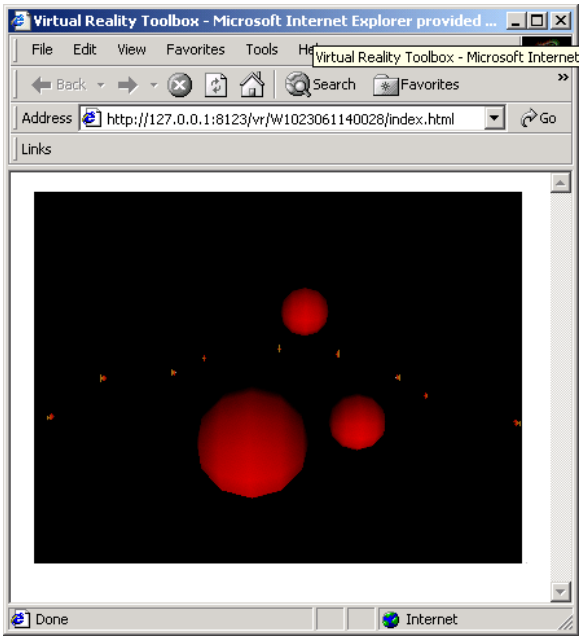

Figure 6: Trade off analysis

As the number of vehicles increased it can be seen that the instincts most at conflict with each other are the collision avoidance and target acquisition instincts. The target acquisition instinct also takes into account the

desire for each vehicle to acquire a particular inertial velocity vector, i.e. speed and heading. As the number of vehicles is increased, the obstacles take second precedence to having each vehicle avoid hitting its neighbor and also heading in the direction it is supposed to travel. It is also interesting that the formation keeping instinct never became a prominent mode for any of the test runs. This may be because the other instincts or influence of the obstacles never resulted in a vehicle becoming too separated from the rest where the formation instinct would be dominant. A future possible test case could include the same type of scenario but with targets as opposed to obstacles, or possibly both.

It was also noticed that the vehicles tended to climb as they approached the obstacles. The same test case as the nine vehicles above was ran, except all vehicles started 400 lower in altitude, which corresponded to the bottom of the first obstacle. The resulting `mode_count` matrix is shown here

```
mode_count =
     3     0     2     0
     4     0     3     0
     4     1     2     0
     1     1     1     0
     2     2     2     0
     2     1     1     0
     3     2     4     0
     5     1     5     0
     3     0     3     1
```

Here the obstacle avoidance instinct becomes a dominant instinct in the vehicles towards the middle of the group as the closest obstacle is directly in front of them. The groups tendency to climb place those in the middle of the group at greater risk of obstacle collision and the instinct is seen to do as would be expected.

6 Future Work

This work was originally started to show the above referenced control algorithm working with a model of a real micro UAV in order to test algorithm performance along with vehicle specific factors. At the time of this writing the model was not available, but because of the decisions made in making this simulation, once the model is available, this simulation will lend itself to its implementation. The vehicle is a micro UAV that has been under investigation by NASA Langley Research Center and Georgia Tech [4]. The simulation environment presented here will facilitate the trade-off analyses needed when testing various combinations of vehicles (type and number) and control algorithms.

7 Acknowledgement

I thank Dr Mark Anderson and Drew Robbins for providing a basis for this work and the initial push in getting the connections established between myself and NASA. I thank Marty Waszak at NASA Langley Research Center for his assistance and willingness in providing help and information for this project.

8 Simulation Files

All files needed to run this simulation are available on MATLAB Central http://www.mathworks.com/matlabcentral/ as long as the user has the appropriate tools installed on their platform. The tools used in this application are:

- MATLAB
- Simulink
- Stateflow
- Virtual Reality Toolbox
- Digital Signal Processing Blockset

All files are provided on an as is basis and are not officially supported by The MathWorks.

9 References

[1] Dynamics and Control of Biologically Inspired Flight Systems, Aerospace Vehicle Technology Program. NASA Langley Research Center.
[2] Ifju, P.G., Waszak, M.R., et al, “Flexible-Wing-Based Micro Air Vehicles”, Paper No. AIAA 2002-0705
[3] Robbins, “Formation Flight as a Cooperative Game”, Paper No. AIAA-98-4124, AIAA GNC, 1998
[4] Waszak M., Davidson J.B., Ifju, P., “Simulation and Flight Control of an Aeroelastic Fixed Wing Micro Aerial Vehicle”, AIAA-2002-4875, AIAA GNC, 2002