

Lightweight UAV Simulation for use in Multi-Agent Human-in-the-Loop experiments

Bram de Beer
Department of Aerospace Engineering
Delft University of Technology
Delft, Netherlands

Michael Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA

INTRODUCTION

Effective use of robotics for large scale applications such as wilderness search and rescue will depend on coordination of large teams. Automating an aerial search, for example, would not only require automating the piloting of planes but also leadership functions, support roles, and incidental forms of cooperation such as hand-offs and refueling. While automated run time coordination is an active research area, work up to now has failed to produce coordination algorithms that could be used to coordinate large numbers of heterogeneous actors in complex and dynamic environments.

Automated coordination involves many challenging problems including distributed task allocation, resource allocation, dynamic formation and re-formation of teams and coalitions, deciding when and what to communicate, how to coordinate for sensing, information fusion and coherent action, all of which have NP-complete or worse computational complexity.

If we look closely at the notions of large cooperating groups, it appears that the strict assumptions present in the theoretical teamwork model do not necessarily need to hold for most cases of coherent coordination. For many tasks, especially those with high degrees of uncertainty, such as the coordination for emergency response, large groups of cooperating agents need to coordinate only loosely. In other words, they do not need to adhere to the tight communications and maintenance

of accurate models of other agents and the state of the team. Yang et al. (2004) have recently proposed and implemented teamwork (Tambe 1997) algorithms that package communications as tokens and use small world properties of networks to provide highly efficient heuristic coordination at a fraction of the cost of prior approaches. This approach, however, is difficult to test in realistically complex domains because of the need to simulate the many entities being controlled.

In the absence of realistic simulation, the robustness of the algorithms with respect to noise, data loss, uncertainty, and the complexity of physically based calculations cannot be evaluated. An additional difficulty arises from the requirement of human controllability and interaction with such systems. This forces the simulations to run in real time as well as adhere to realistic physics and scale to large numbers of UAVs. The challenge of designing such a simulation is that it must be extremely lightweight so that 30+ UAVs can be simulated in real time on a single personal computer while maintaining sufficient fidelity to provide the uncertainties and complexities needed to evaluate coordination algorithms for actual applications.

SIMULATION ARCHITECTURE

To properly simulate a UAV, an aerodynamically correct aircraft model has to be used as the basis of the simulation. Aircraft models come in many different shapes and sizes. For modeling large numbers of UAVs, however, the most

important quality of the aircraft model is a low number of calculations. This means that the simulation must be lightweight enough for a regular personal computer to run several dozen of them simultaneously.

Traditional flight simulators, such as Microsoft Flight Simulator, FlightGear and X-Plane have very accurate aerodynamics models incorporated in their programs, but they need a lot of memory and computing time to accurately calculate the airplane's position and attitude. Running two or more on a computer will result in undesirable waiting time. Therefore, they cannot be used for a multi-UAV task.

A number of multi-UAV simulations have already been developed. These simulations can be divided into two categories; those that use MATLAB and Simulink, thus obtaining a very accurate aircraft model, or those that have only three degrees of freedom.

Programming a multi-UAV simulation using MATLAB and Simulink, although delivering a very accurate simulation, results in the computer using too much computing time and memory for scaling upwards to larger numbers. Therefore, the simulation can only be run for up to eight UAVs simultaneously (Rasmussen & Chandler 2002).

The simulations with three degrees of freedom are a lot more efficient for use in multi-UAV simulations. Unfortunately, by using only three degrees of freedom, a lot of details about the aircraft, as well as aerodynamic properties are left out of the equations. Therefore this strategy of using only three degrees of freedom may not be accurate enough to simulate the complexity and uncertainty needed to evaluate the coordination algorithms and the human controller.

Meeting the requirements for both accuracy and low computation in a

simulation design is very difficult, as they are for the most part mutually exclusive. More accuracy means more equations, which means that the simulation will require more computing time. Therefore, every decision taken in the development of our -UAV simulation was focused on achieving the right balance between accuracy and minimizing calculations.

Keeping the importance of reducing the number of calculations in mind, the choice was made not to use integrals in the calculations. Rather than that, linear estimations are used. Although this means a small sacrifice is made in accuracy (which is dependent on the sampling frequency), it results in a significant decrease in computations.

In order to keep the design simple and comprehensible, modular design was followed. Starting at the core, the aircraft model, modules can be added or subtracted at will, adding detail where needed. This way, it is easy to substitute modules with new modules or even replace modules with other programs.

For modeling the aircraft dynamics, one of many options must be selected. Traditional flight simulators use the linearized equations of motion. These can however only be used for steady, straight, symmetric and trimmed flight conditions. The conditions in which the UAVs are flown are significantly different and the linearized equations can therefore not be used. Furthermore, the equations are quite complex and therefore are not suited for a lightweight application.

A second option for modeling the UAVs is as a Rigid Body. Rigid Body equations assume that distance between any two points on the body is time independent. Although this is a simplification, it is acceptable in the case of most UAVs. The equations used are very simple and therefore the model can be lightweight.

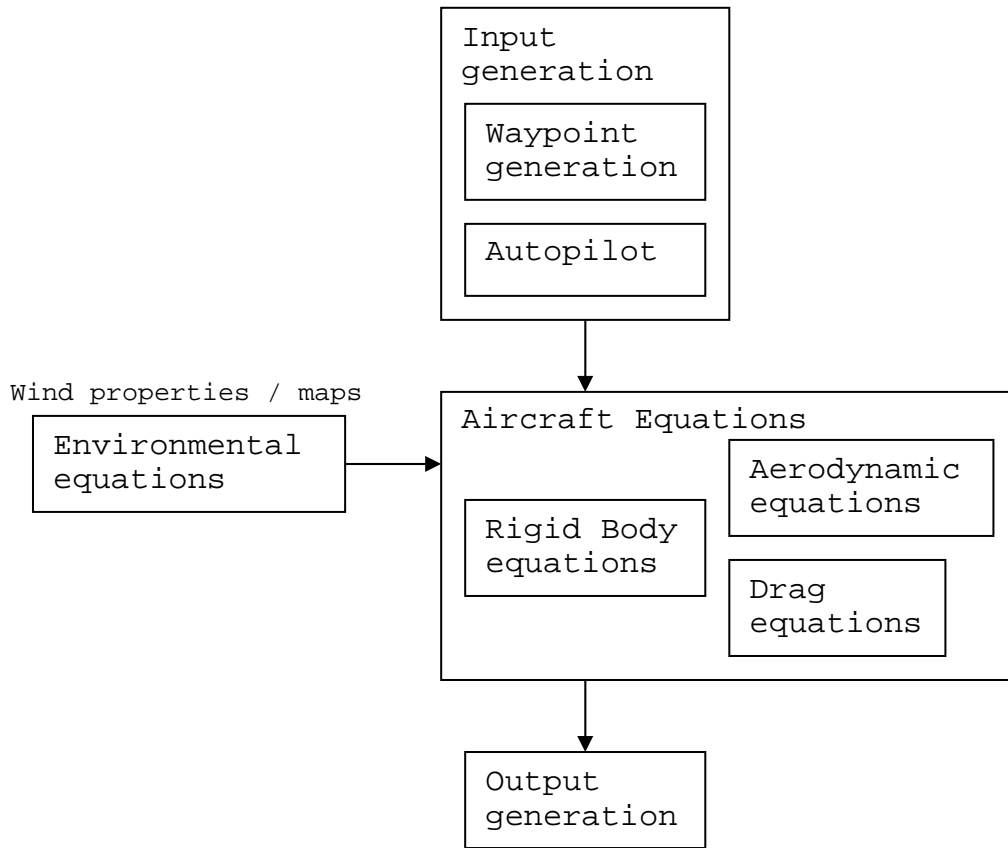


Figure 1. Setup of the simulation

The biggest downside to using a Rigid Body, as opposed to the linearized equations, is that all of the aerodynamic and drag effects have to be added by hand.

Rigid Body UAV Simulation

Using a Rigid Body to simulate an aerial vehicle may seem illogical, since it is so easy to implement the dynamics equations. The linear equations of motion of an aerial vehicle come in convenient state space form and are decoupled for symmetric and asymmetric motions.

In order to build a six degree of freedom simulation with the dynamics equations, however, non-linear equations of motion need to be used. These are six differential equations, which all need to be solved every update. Furthermore, the symmetric and asymmetric motions are coupled.

Using a Rigid Body instead of these equations means the only equations that need solving are three instances of Newton's second law for translations and three instances of Newton's second law for rotations. This can be simplified even further by taking linear estimates of the integrals instead of the integrals themselves. The price for this simplification is a decrease in accuracy, though this decrease should be small if the sampling frequency is high enough.

The big disadvantage of using a Rigid Body instead of dynamics equations is the lack of aerodynamic properties. Gravity, lift and drag forces need to be added to the equations. These are, however, simple equations compared to the differential equations.

RESULTS

By using a Rigid Body instead of the traditional dynamics equations, the amount of calculations required per update is decreased dramatically. Taking lines of code as a surrogate for computation our Rigid Body simulation is more than an order of magnitude shorter than its closest competitor among popular six degree of freedom simulations (Zipfel 2000).

Table 1.

Name of simulation	Lines of code
SRAAM6	5812
GHAME6	4726
FALCON6	1339
Rigid Body	116

To make performance comparisons we have chosen FlightGear (Perry 2004), a widely used open source flight simulator. FlightGear developers assisted us through answering questions and helping us set up the comparison runs. Our comparisons address two basic questions:

1. efficiency: what are the actual differences in computational costs between the two models?
2. accuracy: can the rigid body model approximate the behavior of a FlightGear model based on flight equations?

Efficiency

Computer-based simulations are discrete in nature, because of the time it takes for a computer to evaluate the equations that make up the simulation. This means that values are only known at specific points in time. Values between these “known values” can only be estimated with the assumption that the intermediate values have a certain relation to the nearest “known values”. This is, however, not always the case.

Based on informal experimentation a 20 Hertz simulation frequency seems to be the

best choice for this application. A lower frequency would result in too little accuracy (causing unreliable results) and a higher frequency would require more calculations per second (reducing the maximum number of UAVs that can be simulated simultaneously).

The main purpose of using a Rigid Body for the UAV simulation was the speed with which the new state can be calculated. After re-programming the simulation in C (to achieve the highest possible computing speed), the time needed for calculating one time step was measured. Since this time was not constant, the average of 100 calculations was taken.

The time needed for calculation of the Rigid Body equations is $1.83 \cdot 10^{-4}$ seconds. This means the simulation can be run 5400 times per second on a single computer. This also means that, using the previously determined simulation frequency of 20 Hertz, the simulation can be run for up to 273 separate UAVs simultaneously for a real-time application on a single computer.

The same calculation can be made for the FlightGear simulation. The average time needed for the FlightGear simulation is much higher than that of the RB UAV simulation. The variance of the calculation time is also a lot higher. FlightGear needs on average $8.1 \cdot 10^{-3}$ seconds for calculating the dynamics equations. This means that, when taking the same 20 Hertz simulation frequency, it can run only 6 simultaneous simulations on one computer for a real-time application. The RB UAV simulation performance therefore is comfortably in excess of our 30+ UAV target and the 6 UAV limit found for the FlightGear dynamics equations confirms the limitations of conventional approaches we were trying to overcome.

Accuracy

To determine whether the RB UAV model could approximate the behavior of a flight

equation based model, FlightGear, we flew both models over a series of paths. Figure 2 shows the flight paths of the FlightGear flight simulator (pale gray) and the Rigid Body UAV simulator (dark gray and black) for the same set of constants and a similar autopilot (because of the differences between the two simulations, small alterations had to be made to the autopilot algorithms).

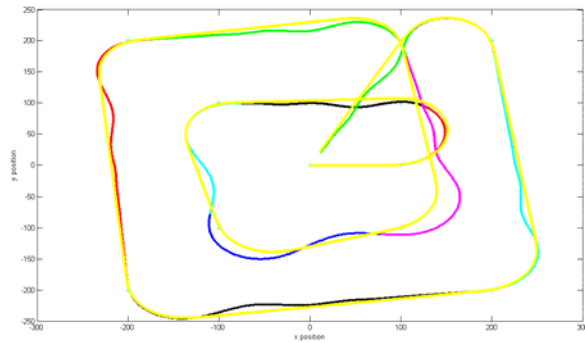


Figure 2. *Flight path FlightGear flight simulator (pale gray) and Rigid Body UAV simulator (dark gray and black)*

There are some obvious differences between the two flight paths. The FlightGear path is straighter than that of the RB UAV simulation. To determine what the reason for the difference was, the errors of both systems were compared.

Figure 3 compares the heading angle error and altitude error for both the simulations. The heading angle error shows that the FlightGear simulation has much more damping in its system than the RB UAV simulation. This results in errors being eliminated much faster. The overshoot and oscillations are therefore absent in the FlightGear graphs. This means the airplane followed a more direct path through the waypoints.

When comparing the altitude errors, there was also a large difference between the two simulations. In this case the error of the FlightGear simulation was much larger than the error of the RB UAV simulation.

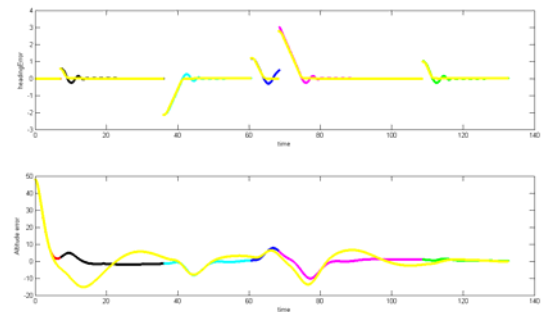


Figure 3. *Flight and altitude path errors FlightGear flight simulator (pale gray) and Rigid Body UAV simulator (dark gray and black)*

For purposes of approximation both heading and altitude of the simulations appear to agree closely enough to make the highly efficient RB UAV simulation a suitable substitute for use in the multiple vehicle control experiments for which it was designed.

Autopilot and Other Components

A simple controller was used for the autopilot, since low complexity of the controller reduces the number of calculations necessary and thus increases the speed of the simulation. For the current application, a very simple PID controller was used to steer the UAVs towards waypoints.

At present, no collision avoidance has been built into the simulation, as this higher level of control is meant to be performed by coordinating agents. Collision avoidance could be added later, however, as anything can be linked to the program, as long as it has the right inputs and outputs.

Figure 1 shows the setup of the simulation. The inputs of the aircraft equations are forces, which means the output of the autopilot and the environmental modules have to be forces. The output of the aircraft equations are position (in inertial frame), velocity (in vehicle frame as well as in inertial frame), attitude and angular

velocity. The inputs can come from any source, as long as they are in the right format. The same goes for the outputs. These can be used in any program or application that can interpret the numbers correctly (for example: OpenSceneGraph).

CONCLUSION

Our lightweight UAV simulation balances efficiency with fidelity by abandoning traditional flight equations in favor of rigid body modeling adding in effects of gravity, lift, and drag to the result. While conventional flight simulations are difficult to integrate with physics-based game engines that typically use OpenDynamics, Karma, Novodex or Havoc physics engines relying on rigid body modeling; enhanced rigid body models such as ours should prove more compatible.

For our present objective of simulating large numbers of UAVs in real time this lightweight model meets all of our design objectives.

ACKNOWLEDGMENTS

This work was supported by ONR MURI on Understanding and Measuring Macrocognition in Teams, University of Central Florida subcontract 92070.

REFERENCES

- Perry, A. The FlightGear Flight Simulator, Proceedings of the UseLinux Annual Technical Conference, 2004.
- Rasmussen, S and Chandler, P. MultiUAV: A multiple uav simulation for investigation of cooperative control, Proceedings of the 2002 Winter Simulation Conference, 869-877, 2002.
- Tambe, M. "Towards Flexible Teamwork" Journal of Artificial Intelligence Research, Volume 7, Pages 83-124, 1997
- Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M. and Sycara, K. An integrated token-based algorithm for scalable coordination, Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), 407-414, 2005.

Zipfel, P. Modeling and Simulation of Aerospace Vehicle Dynamics, American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2002.

BIOGRAPHIE

Bram de Beer is a master's candidate in Aerospace Engineering at Delft University of Technology. He was serving an internship at the University of Pittsburgh when this work was conducted.

Michael Lewis is a professor in the School of Information Sciences at the University of Pittsburgh and researcher in human-robot interaction.