Integration of Autonomous UAVs into Multi-agent Simulation

Martin Selecký[1], Tomáš Meiser[2]

[1] *Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Technická 2, 166 27 Prague, Czech Republic*

[2] *Dept. of Computer Science, Faculty of Electrical Engineering, Czech Technical University, Technická 2, 166 27 Prague, Czech Republic*

Corresponding author: martin.selecky@agents.fel.cvut.cz

Abstract

In recent years, Unmanned Aerial Vehicles (UAVs) have attracted much attention both in the research field and in the field of commercial deployment. Researchers recently started to study problems and opportunities connected with the usage, deployment and operation of teams of multiple autonomous UAVs. These multi-UAV scenarios are by their nature well suited to be modelled and simulated as multi-agent systems.

In this paper we present solutions to the problems that we had to deal with in the process of integrating two hardware UAVs into an existing multi-agent simulation system with additional virtual UAVs, resulting in a mixed reality system where hardware UAVs and virtual UAVs can co-exist, coordinate their flight and cooperate on common tasks. Hardware UAVs are capable of on-board planning and reasoning, and can cooperate and coordinate their movement with one another, and also with virtual UAVs.

Keywords: Unmanned Autonomous Vehicles, multi-agent systems, deployment.

1 Introduction

The use of UAVs is growing nowadays thanks to the low cost of their deploying and maintaining them, and the possibility to operating them in areas inaccessible or dangerous for human pilots. To manage more sophisticated tasks such as area surveillance and monitoring or multiple target tracking, teams of multiple UAVs should be deployed. This requires more complex control, coordination and cooperation mechanisms. These mechanisms have already been studied and developed for virtual UAVs as a part of multi-agent systems, e.g. in the AgentFly system [8] or MAS, proposed by Baxter and Horn in [1] which would be very well suited for application in real hardware UAVs.

A further significant challenge in working with hardware air vehicles is that the design/test cycle is considerably longer than for ground robots or virtual entities. Flight experiments involve a greater level of risk, since even minor errors can lead to serious crashes. For this reason mixed reality simulations, which allow integration of real hardware UAVs and simulated virtual UAVs to co-exist in one environment, are helpful and necessary in the development process.

The principles of multi-agent systems have been used in [2], [3] and [6] to issue commands to hardware UAVs. In these works, however, central planning mechanisms are used, and the UAVs only follow precomputed plans.

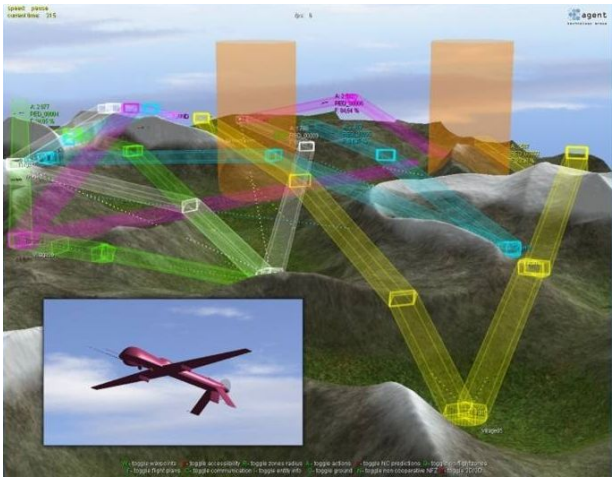


Figure 1: AgentFly multi-agent system.

Bürkle et al. [4] presented the deployment of control mechanisms for teams of hardware VTOL micro UAVs. These UAV teams are capable of on-board planning and some cooperation on mutual tasks, but they do not coordinate their flight in terms of collision avoidance. The system also does not allow the coexistence of hardware UAVs together with simulated UAVs.

We integrated two hardware fixed wing UAVs into the AgentFly multi-agent system [8], which is used for simulating UAVs and air traffic, allows complex coordination and cooperation of agents, and provides collision avoidance mechanisms. We modified the sys-

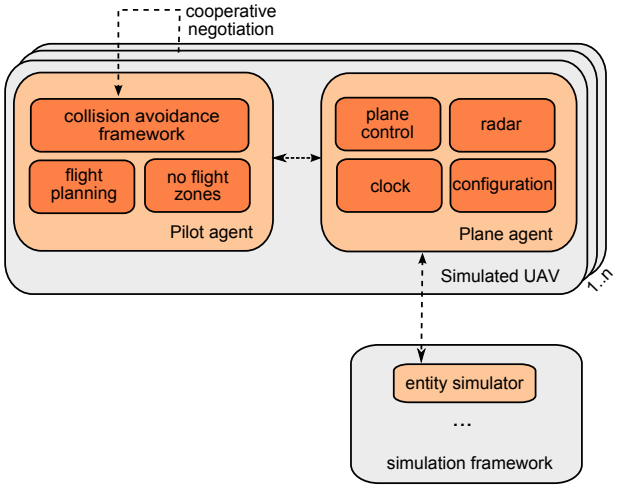


Figure 2: Design of a simulated virtual UAV.

tem to allow both hardware and virtual UAVs to act and interfere in a common mixed reality environment, and we equipped the hardware UAVs with a Gumstix computer to enable on-board planning, reasoning and communication with other hardware or software UAVs.

This paper is organized as follows. First, we will briefly describe the AgentFly system in Section 2 and UAVs and their computational and communication equipment in Section 3. Section 4 shows the modifications done to the AgentFly system and hardware equipment for deploying the UAVs, and in Section 5 we describe the results of field experiments. Section 6 concludes the paper.

2 AgentFly Multi-Agent System

AgentFly is a Java-based multi-agent system designed for simulation of air traffic and UAV missions (see Figure 1). It is built on top of the A-globe multi-agent platform [7], and it provides the simulated entities with a communication framework, trajectory planning, and collision avoidance and cooperation mechanisms.

Each simulated aircraft in the AgentFly system is composed of two software agents — the *Pilot agent* and the *Plane agent*. Pilot agent is responsible for the UAV's reasoning — it calls the trajectory planner and handles the communication with other UAVs for the purposes of cooperation and collision prevention. Plane agent provides Pilot agent with an interface for high-level plane control, and executes the flight plans. It also simulates the aircraft's on-board instruments, e.g. radar, GPS and clock.

The system provides simulated aircraft with mechanisms for peer-to-peer collision avoidance — either *cooperative* mechanism, where the UAVs exchange

Figure 3: Unicorn UAV by Procerus Technologies.

Figure 4: Gumstix Overo Fire on-board computer .

their flight plans and negotiate about avoidance manoeuvres, or *non-cooperative* mechanism, where the UAVs cannot exchange (e.g. because of incompatible protocols) or do not want to exchange their plans (for example hostile airplanes) and the collision situation is solved by flight trajectory prediction.

The flight plans consist of a sequence of flight manoeuvres — straight flight, left/right turn, up/down turn. They represent so called Dubins curves [5], which are based on the fact that any two positions (points with directions) in space can be connected by a sequence of a turn of a given radius, a straight segment and another turn. Each manoeuvre is represented by the starting point, turn angle or acceleration, speed and duration.

During the transition process from software simulation to mixed-reality we replaced two of these virtual aircraft with hardware commercial off-the-shelf fixed wing UAVs by Procerus Technologies.

3 Hardware UAV Platform

We selected the Procerus UAV (Figure 3) because of its relatively low cost, built-in cameras and sensors, ease of hardware integration and easy repairs, and the included Kestrel autopilot. The Kestrel autopilot is capable of waypoint navigation, thus removing the

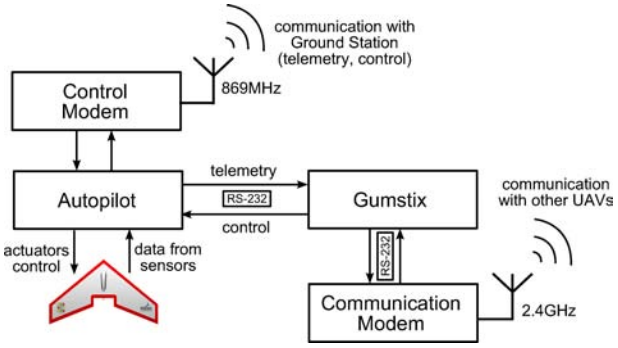


Figure 5: Scheme of hardware UAV design.

necessity for low level UAV control, and it provides telemetry and GPS info about the aircraft via an RF modem.

To provide computational capacity for on-board planning and communication, we equipped the UAV with a Gumstix Overo Fire computer on module (COM) (Figure 4). This computer is based on ARM Cortex-A8 architecture with 512 MB RAM running at 720 MHz with Linux OS and Java Embedded to enable running of the modified AgentFly multi-agent system.

Figure 5 shows the block design of the deployed hardware UAV. The Kestrel autopilot is connected to the sensors and actuators to provide low level UAV control. By the connected 869 MHz Microhard modem, the autopilot distributes telemetry and GPS info to the ground station and receives control commands in cases when manual control is required or when the autonomous on-board planning and control algorithms fail. The autopilot is also connected to the Gumstix Overo COM by RS-232 line. By this line it distributes the telemetry and GPS data to the on-board planner and receives standard control commands — mainly a sequence of waypoints in return. Communication with other hardware or virtual aircraft is carried out by an additional 2.4 GHz XBee modem.

The two modems have their counterparts on the ground station PC to pass the telemetry and communication to the simulation, and vice versa.

4 Modifications made to the multi-agent system

The software integration of the hardware UAVs into the multi-agent system is depicted in Figure 6. It can be seen that the hardware UAV entity consists of an on-board part responsible for planning, flight execution and other reasoning, and a part located on the ground station PC responsible for visualization and exchange of position and telemetry information (block in the figure denoted as 'radar') between the real and virtual entities.

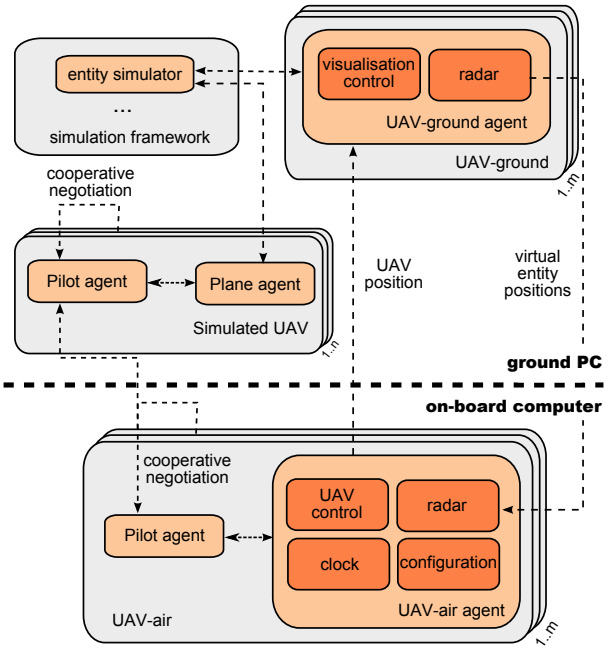


Figure 6: Design of modificated MAS for mixed hardware and virtual UAV simulations.

In order to integrate of the hardware UAV, some parts of the simulation and planning software had to be modified:

Waypoint navigation of the Kestrel autopilot was not compatible with the AgentFly's manoeuvre-like flight plan structure.

Wind in the real environment significantly influenced the precision of plan execution.

Position uncertainty caused by errors of sensors and actuators or environmental effects.

Unreliable communication and low bandwidths of the RF modems caused problems in the collision avoidance and cooperative mechanisms.

The simulation time needed to be synchronized, otherwise it caused problems in cooperative collision avoidance negotiations.

We will now describe the necessary changes that had to be applied to deal with these problems.

4.1 Waypoint Navigation

As was stated above, the Kestrel autopilot provides the operator with waypoint navigation capability. The waypoints are uploaded as GPS coordinates, so we had to sample the UAV's plan, which is represented as a list of flight manoeuvres. The manoeuvres are sampled according to the acceleration or turn angle — the more rapid the change in speed or angle, the more dense is the sampling (see Figure 7).

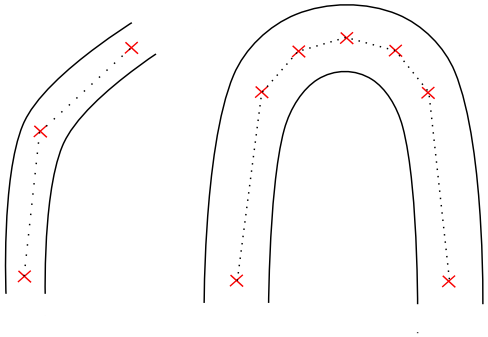


Figure 7: Flight manoeuvre sampling. The larger the turn angle, more dense the sampling is.

Another problem is that the planner works with Cartesian coordinates. The sample waypoints therefore have to be transformed to GPS coordinates using linearisation of the world at the point specified by the position of the ground station. This position represents the origin of a Cartesian coordinate system with the x-axis pointing to the east, the y-axis to the north and the z-axis upward (see Figure 8). This linearisation causes errors that are less than 10 cm in measurements of distances, and less than 8 m in measurements of altitudes at a distance of 10 km from the origin and 500 m higher.

4.2 Planning in Wind

Wind has a significant effect on the precision of UAV plan execution. Apart from gusts of wind that drift the aircraft and cause position uncertainty, see the next section, stable winds especially influence the minimal turn radius of the UAV — one of the main parameters of the trajectory planner. An aircraft flying against the wind is capable of turning with a much smaller turn radius than an aircraft flying with the wind. In these conditions, the original planner that was based on planning with Dubins curves (i.e. straight segments and turns with constant a radius) created trajectories could not be followed in presence of wind.

We modified the planner to use trochoidal curves. The trajectories can then be constructed as Dubins curves in a coordinate system that moves in the same direction and at the same speed as the wind (see Figure 9), and can be followed much more precisely, even in strong wind.

4.3 Position Uncertainty

Because of errors of the sensors and actuators and effects of the environment, e.g. gusts of wind, the aircraft's position estimation is not and cannot be precise. For effective coordination and control of UAVs, this uncertainty must be modelled and the

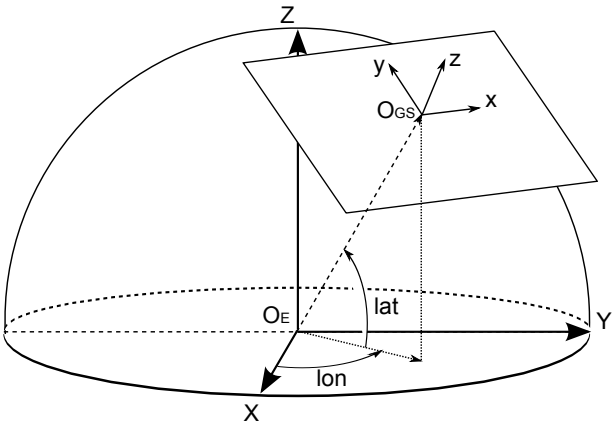
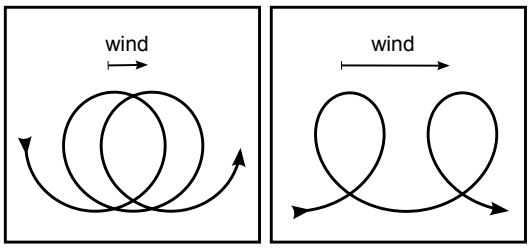


Figure 8: Linearisation of the coordinate system.



(a) $\frac{|\mathbf{wind}|}{|\mathbf{velocity}|} = 0.1$ (b) $\frac{|\mathbf{wind}|}{|\mathbf{velocity}|} = 0.3$

Figure 9: Effect of wind on the flight trajectory.

planning algorithms must take the uncertainty into account.

We distinguish two types of errors — Δ_T for *time-related errors* and Δ_D for *distance-related errors* (see Figure 10). The distance-related error is the deviation of the UAV from the planned spatial trajectory. It can be caused by sudden wind gusts, wind changes, high airspeeds or imprecise autopilot control when the aircraft is unable to follow the trajectory correctly. The time-related error is then the deviation of the UAV from the time plan. This is caused by imprecise autopilot velocity control, by accumulated small delays that emerge when the autopilot repairs small spatial deviations from the plan, or by high wind speeds when it is difficult to keep the desired velocity of the aircraft.

To handle these uncertainties, we use the so-called *safety zone* around the UAV. Generally, the worse the flight plan execution performance, the bigger the safety zone needs to be in order to keep the plane far apart from obstacles or other planes.

When specifying the dimensions of the safety zone we distinguish two different safety ranges — *safety time range* s_T, and *safety distance range* s_D (see Figure 11). The safety time range is given by the maximum allowed time-related error Δ_T, and the safety distance range is given by the maximum allowed distance-related error Δ_D. When the UAV gets outside any of these ranges, trajectory replanning is

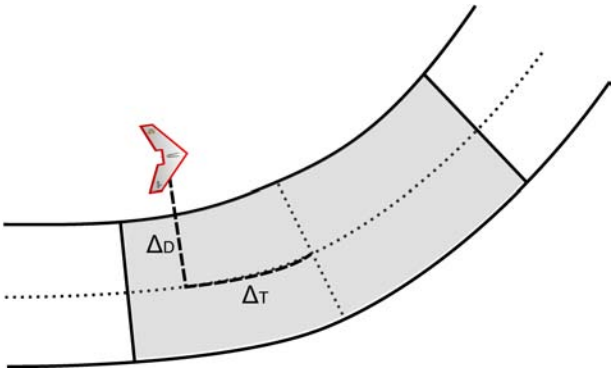


Figure 10: Two possible error types in plan following- Δ_T for time-related error, and Δ_D for distance-related error.

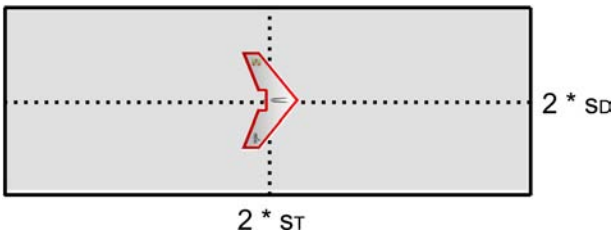


Figure 11: Two different safety ranges — *safety time range* s_T, and *safety distance range* s_D.

scheduled. The safety time range is bigger than the safety distance range, because it is generally more difficult for the aircraft to keep up with the plan in the time domain.

The safety zone is used during the planning procedure so that it is wrapped along the planned trajectory and it cannot cross any obstacle, no-flight zone or other planned trajectory in time and space.

4.4 Unreliable communication

Communication is one of the most crucial features of any multi-agent system. In the AgentFly system, all collision avoidance and cooperative negotiations, e.g. plan and position exchange and coordination commands, are conducted by message exchange. In a simulation the messages are transferred by the reliable TCP/IP protocol, but in a real environment UAVs use RF modems with limited bandwidth, with possible interference to their signal from other RF devices, and with significant signal attenuation with distance.

The largest portion of the communication bandwidth is used by plans exchange during cooperative collision avoidance. Figure 13 shows the required bandwidth for worst case collision avoidance negotiation in superconflict scenarios, where all the UAVs are flying against each other with one collision point in the middle of them (see Figure 12). We presume the initial distance of the UAVs to be 1.5km, with

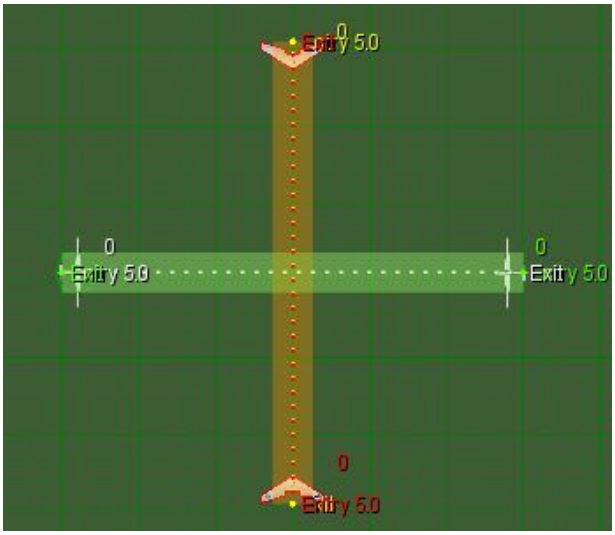


Figure 12: Superconflict collision avoidance scenario.

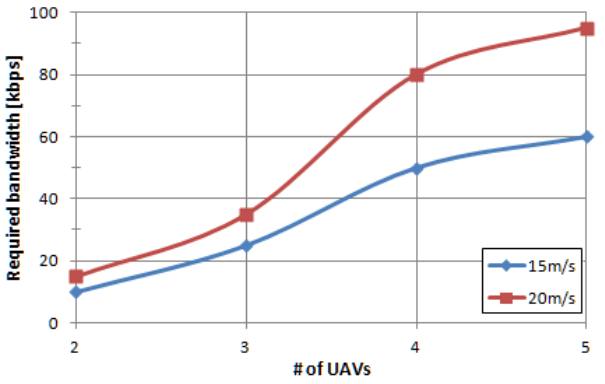


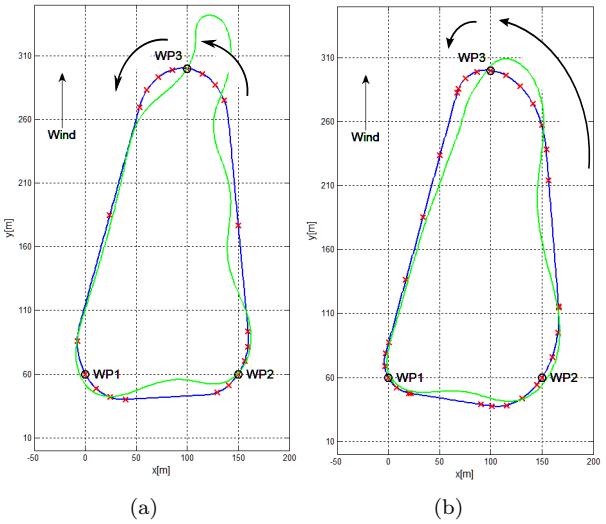
Figure 13: Required bandwidth for collision avoidance negotiation in superconflict.

the requirement that the conflict is to be resolved 10 s before the collision point.

Apart from plan exchange, additional bandwidth required is for manual safety control, telemetry broadcasts and communication management control. This needs 10–30 kbps of additional bandwidth. Commercial modems that are capable of communication at a distance of at least 1.5 km have maximum RF bandwidth around 115 kbps, which with the additional and safety bandwidth is not enough to handle even 4 UAV superconflict scenarios. Moreover, operating the modems at full speed requires a significant portion of CPU time. For that reason we decided to use two independent RF modems operated by Kesterl autopilot and Gumstix COM on-board CPU units, as shown in Figure 5.

4.5 Simulation time

Collision avoidance algorithms and other cooperation and coordination mechanisms need to have synchro-



(a) (b)

Figure 14: The effects of planning with (a) Dubins curves and (b) trochoidal curves.

nized time to work properly. There are two ways to synchronize simulation times.

Because all autopilot modules and also the ground station PC are connected to GPS modules, we can synchronize the clocks from the time information contained in GPS updates. This method can give very precise time information, but we have found out that Kestrel autopilot sometimes provides wrong time information in GPS updates that need to be recognized and taken out from the measurements.

The second way is to pass the ground station's simulation time during a registration process of the hardware UAVs to the system and then start to measure time from that moment. In this method there is a problem in the information transfer time — it can take up to one second to transfer the registration data along with the simulation time, and this can have a significant effect on time synchronization. However we decided to use this approach because the synchronization error is smaller than the time errors from the GPS updates.

5 Experiments

We performed several field tests to check and verify the modifications described above.

First, we compared the plan execution precision in cases with plans found by the original planning algorithm and with plans found by our modified algorithm for planning in wind. The tests were conducted with approximately 5 m/s wind, the UAV's airspeed was 15 m/s, and the scenario consisted of three waypoints placed as shown in Figure 14.

The blue line in Figure 14 is the ideal planned trajectory that starts at the first waypoint (WP1), then proceeds via WP2 and WP3 and ends again at WP1. The green line is the real recorded trajectory of the UAV, and the black arrows emphasize the different turn radii in individual cases. It can be seen that the trajectories created with Dubins curves with a constant minimal turn radius could not be followed at some points. On the other hand, the trajectories created with trochoidal curves that were adapted to the wind strength and direction were followed much more precisely.

Another experiment that we performed was a mixed reality collision avoidance test. We prepared a scenario with one hardware UAV and one virtual UAV flying against each other. The purpose of this experiment was to test the functionality of the collision avoidance mechanism between the real and virtual aircraft, and also to test the sufficiency of the RF bandwidth required for the communication. The experiment was successful: the UAVs needed less than 10 s to solve the collision situation, which with airspeeds of 15 m/s corresponds to 300 m distance flown from the beginning of the negotiation.

In future we would like to perform experiments with two hardware UAVs, later adding one and more virtual UAVs in order to study the scalability of the problem.

6 Conclusion

In this paper we have presented problems and solutions connected with integrating hardware fixed wing UAVs into an existing multi-agent system for UAV simulation. We are now able to verify the functionality of the collision avoidance and cooperative mechanisms in a real environment, and also to find possible bottlenecks and limitations, e.g. the maximum available RF bandwidths for communication, and the limited computational capacity of the on-board computers.

Our work is a first step toward full deployment of the AgentFly system on real hardware UAVs that could operate independently in coordinated teams during tactical missions.

In our future work, we would like to extend the tactical cooperation possibilities of UAVs and also to deploy autonomous VTOL quadcopters next to the fixed wings and provide them with mechanisms for mutual flight coordination and cooperation.

Acknowledgements

The project described in this paper was supervised by Ing. M. Rollo, PhD., FEE CTU in Prague, and was supported by Czech Ministry of Defence grant OVCVUT2010001.

References

[1] J. W. Baxter, G. S. Horn. Controlling teams of uninhabited air vehicles. In *Proceedings of the fourth international joint conference on Autonomous Agents and Multiagent Systems*, ACM, 2005, pp. 27–33.

[2] J. W. Baxter, G. S. Horn, D. P. Leivers. Fly-by-agent: Controlling a pool of UAVs via a multi-agent system. *Knowledge-Based Systems* **21**(3):232–237, 2008.

[3] R. W. Beard, T. W. McLain, D. B. Nelson et al.. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE* **94**(7):1306–1324, 2006.

[4] A. Bürkle, F. Segon, M. Kollmann. Towards autonomous micro UAV swarms. In *Journal of intelligent & robotic systems*, Springer, 2011, pp. 1–15.

[5] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics* **79**(3):497–516, 1957.

[6] P. Scerri, T. von Gonten, G. Fudge et al. Transitioning multiagent technology to UAV applications. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 89–96.

[7] D. Šišlák, M. Rehák, M. Pechouček. A-globe: Multi-Agent Platform with Advanced Simulation and Visualization Support. In *Web Intelligence*, IEEE Computer Society, 2005, pp. 805–806.

[8] D. Šišlák, P. Volf, Š. Kopřiva et al. AGENTFLY: A Multi-Agent Airspace Test-bed. In *Proceedings of 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, May 2008.