This is an accepted version of the article: Vrba, P., Mařík, V., Přeučil, L., Kulich, M., and Šišlák, D. (2007). **Collision Avoidance Algorithms: Multi-agent Approach**. *Holonic and Multi-Agent Systems for Manufacturing - HoloMAS 2007*,

Collision Avoidance Algorithms: Multi-agent Approach

Pavel Vrba[1], Vladimír Mařík[1, 2, 3], Libor Přeučil[2], Miroslav Kulich[3], and David Šišlák[2]

[1] Rockwell Automation Research Center
Pekařská 695/10a, 15500 Prague, Czech Republic
{pvrba, vmarik}@ra.rockwell.com

[2] Department of Cybernetics, Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
{marik, preucil, sislak}@labe.felk.cvut.cz

[3] Center of Applied Cybernetics, Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
{marik, kulich}@labe.felk.cvut.cz

Abstract. The paper deals with the methods for detection and avoidance of collisions of autonomously moving vehicles utilizing the principles and techniques of multi-agent systems. Three different scenarios are discussed: (i) movement of AGVs in 2D space with fixed trajectories, (ii) movement of autonomous robots in an open 2D space and (iii) collision-free flights of unmanned aerial vehicles. For each category, an agent-based solution is proposed. Presented experiments show that the cooperative approach to detecting and avoiding collisions based on negotiation and goal sharing of agents, representing vehicles, seems to be highly efficient. The multi-layer architecture combining cooperative approach with algorithms based on dynamic no-access zones for avoiding non-cooperative vehicles provides good results in generating collision-free flight corridors.

1 Introduction

A robot is a computer controlled integrated system, capable of autonomous and goal-oriented interaction with real environment according to man instructions. The interaction requires that the robot is able to perceive its environment, make decisions and carry out actions in the outer world. There is an increasing number of applications that cannot be solved just by using a single robot, but require deployment of larger number of robots to ensure distributed perception, distributed action and distributed goal-oriented decision making. For instance, the assembly of highly complex machinery in hardly accessible terrain or space, activities carried out during nuclear power plant failures or rescue and military operations fall in these categories. Consideration of groups of robots makes sense also in situations where it is effective to physically and spatially distribute the perception, decision-making and action tasks – small specialized robots for instance carry out the exploitation of con-

taminated or hardly accessible areas, information is processed by other robots (often not able to move in space) while the physically intensive operation (barrier removal, pipe cut off or mine defusing) is usually performed by large powerful robots possibly in coordinated manner.

In all examples mentioned above there is a notion of *autonomous* or *unmanned vehicles* that have to be able to autonomously move in an unknown and dynamically changing environment while collaborating on common goals. This introduces a new category of theoretical and technical problems linked for example with communication, decision-making on the basis of partial knowledge, coordinated movement etc. This paper focuses on the mutual *collision detection and avoidance* problem. Collision can be considered as a state when the movement trajectories of two or more vehicles cross at the same time in a particular point in the environment. The collision detection means the ability of a vehicle to discover the possible collision beforehand. When discovered, the vehicles may apply different algorithms for re-planning their paths to avoid the collision point.

The paper discusses different categories of this problem from the viewpoint of the characteristics of the environment in which the vehicles move:

1. 2-D area with predefined paths that the vehicles must follow. Typical example of this category are the AGVs (Autonomous Guided Vehicles) deployed in factories and warehouses moving along predefined network of paths embedded in a factory's floor usually in form of magnetic stripes or rails.
2. 2-D area where the vehicles can move in any direction in two-dimensional space. The representatives of this large category are for instance robots cooperatively exploring unknown environment, robots playing soccer (http://www.robocup.org), AGVs with freedom in movement etc.
3. 3-D area where the vehicles can freely move in three-dimensional space. This category includes for example UAVs (Unmanned Aerial Vehicles) [4] and UUV (Unmanned Undersea Vehicles) [5] usually deployed in military missions. Routing of commercial jet planes is another example.

To effectively solve collision avoidance problem in a cooperative manner, robots have to be able to:

- communicate with each other using the common language, common negotiation mechanisms and common knowledge representation formats (ontology); in this case it is referred to as *cooperative collision avoidance*, otherwise when the robots cannot communicate for any reason, we talk about *non-cooperative collision avoidance*
- share knowledge about their location in the environment and anticipated direction of movement and keep this knowledge consistent and up-to-date in order to minimize the communication channels load in time critical moments
- reason about the movement plans of neighboring robots to detect possible collision situation and avoid it by re-planning the movement trajectory

2 Multi-agent approach to collision avoidance

Organizational structures and control algorithms of multi-robotic system are more and more often designed and implemented using the methods and principles known from the domain of *multi-agent systems* [9]. These two research domains have evolved relatively separately from each other, nevertheless the technological enhancements over the last few years, like smaller and cheaper robotic platforms, faster processor, more sophisticated sensors, wireless communication, new results in the multi-agent research domain, etc. lead to their convergence. The algorithms developed originally for highly distributed agent decision-making are now being applied for control of activities of robot groups. It is mainly the algorithms for distributed action planning, acquaintance models for social knowledge modeling [7] and negotiation mechanisms and protocols for inter-agent communication [8].

When designing a multi-agent system representing the community of robots, the agent should represent an independent functional part, usually the robot as a whole or its functional subsystem (in this case a single robot is represented by more agents). Subsequently, an appropriate software environment, so-called agent platform, where the agents exist have to be chosen. The agent platform should provide the white pages services holding information on registered agents and their addresses and the Directory Facilitator (yellow pages) where the agents register their services and search for suitable service providers. The agent negotiation is usually based on contract-net-protocol [2] or various auction mechanisms. An important aspect is sharing of semantic information, especially about surrounding environment in form of environment maps used by robot to localize itself, other robots, available paths or obstacles.

2.1 Collision avoidance in 2-D space with fixed trajectories

First category of collision detection and avoidance problem refers to the movement of AGVs (Automated Guided Vehicles) used in factories or warehouses for transportation of products or materials. This is a special case where there is a built-in network of magnetic or optical stripes or rails in the factory's shop floor which the AGV is able to observe and move along [10]. The network can be viewed as a graph with edges representing particular segments of network the AGVs travel through and vertices representing two different types of nodes: (i) a junction point where two or more segments cross and (ii) a work station representing source and/or destination place on factory floor between them the transportation is done. The work station is for instance a CNC machine, assembly machine, buffer storage station, etc.

Figure 1 shows an example of AGV transportation system with two work stations (`W1` and `W2`) and a network of segments designed to provide redundant transportation paths between the two work stations. It is supposed that the idle AGVs wait in a special dedicated segment(s) (`s5` in Fig. 1). Although the AGV can move through a particular segment in any direction, there obviously cannot be two AGVs moving in the same segment in opposite directions.

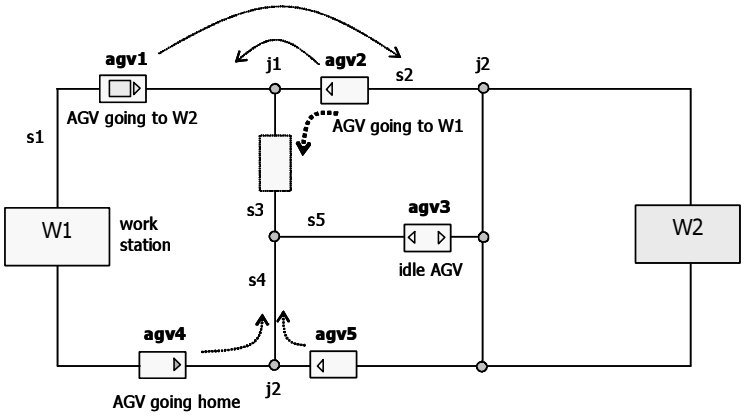

Figure 1. Possible collisions in AGV transportation system

A primary collision situation appears at the junction point, which only one AGV can go through at the same time. This is a case of `agv5` and `agv4` in Figure 1 that have to go through junction `j2` in a sequence – thus `agv4` have to wait until `agv5` enters segment `s4` and leaves the area around the junction. A secondary collision situation happens when two AGVs approaching a junction intend both to continue in segment occupied by the other one. As depicted in Figure 1, the collision is around the junction `j1`, where the `agv1` in segment `s1` wants to enter segment `s2`, which is occupied by `agv2` that intends to go to segment `s1`. A possible solution to avoid this collision is that one AGV avoids to third segment (`agv2` avoiding to `s3` in Fig. 1), waits until the second AGV passes the junction and then continues in the released segment as intended.

The proposed AGV agent is equipped with a complex behavior allowing it to negotiate with the work station agents about products transportation, navigate through the network to get to a particular work station as fast as possible, detect and avoid possible collisions with other AGVs and even dynamically detect and avoid obstacles in the paths. When instantiated, the AGV agent is given the XML description of the network comprising of the location of all nodes (X, Y coordinates), their type (work station, junction or curve) and their interconnection via segments. The AGV agent translates this description into an internal object model that is used to precompute all the possible routes to work stations from each junction node. This information is then used by the AGV to compute the length of an anticipated path between the source and the destination workstation and also for the navigation when determining the optimal direction at the junction point [13].

For the primary collision situation mentioned above, the first AGV agent that approaches a junction point declares itself as a *master*, while informing the second AGV that it became a *slave*. Additionally, the master AGV includes an estimation of the time period needed to go through the junction point. Then, the master AGV goes through the junction as the first one and, after the estimated period passes, the slave AGV moves through as the second one. It may happen that both AGVs reach the junction point at the same time and thus both declare themselves as masters. In

this case, the AGV agent with the lower priority of carried work piece or, if priorities are same, the one whose name is second according to alphabetical ordering freely gives up and becomes a slave.

Detecting and avoiding secondary type of deadlock is a bit more complicated issue. When the AGV approaches a junction node it determines what direction to choose, i.e. to which segment to continue to get to intended workstation by shortest path. However, a collision may happen in this segment if there is already another AGV moving toward the same junction. To detect such a situation the AGV should know in advance, if the segment is free or not. For this purpose, each AGV agent is equipped with knowledge about the position and direction of all other AGVs. This information is exchanged between AGV agents via messages – when a particular AGV leaves a segment and enters another one, it informs about this event all other AGVs. In situation depicted in Figure 1, `agv2` in segment `s2` that approached junction `j1` is aware that there is another AGV (`agv1`) in segment `s1` moving toward the `j1` junction. To detect if there is a collision or not, the first AGV asks the second one what segment it plans to go to and how long will it take to reach the junction. If collision state shown in Fig. 1 is recognized, the `agv2` avoids it by moving to another segment (`s3`) where it stops and waits until the colliding AGV passes the junction; then it continues as intended. In non-collision situation, the AGV just waits until the second one leaves the junction point. In both cases, the first AGV considers the waiting time period and decides if it would be more beneficial to take an alternative path to intended work station if waiting would be too long.

The AGV agent is also equipped with a mechanism for obstacle detection and avoidance. When the AGV hits an obstacle in particular segment and there is no other AGV in front moving in that segment in same direction, it marks the segment as blocked in its internal model of environment, turns back and informs all other AGV agents that the segment is blocked and should be avoided. However, if there is another AGV in front, the first AGV uses a ping-like mechanism to test whether the in-front-AGV is failed or not. If there is no response within some period of time (1s), the first AGV assumes that it hit the failed AGV – it turns back and again informs all other AGVs that the segment is blocked. When response is received (the asked AGV includes information about its own position in segment), the first AGV determines if it hit an obstacle or the in-front-AGV and reacts appropriately.

2.2 Collision avoidance in open 2-D space

Starting from graph-bounded collision avoidance, the more general problems incorporate situations with freely bounded vehicles. Solution of this class of tasks is typically required for systems providing large portion of autonomy and intelligence – self guided vehicles represented mainly by autonomous robots. On contrary to the previous case, these robots have no pre-determined motion trajectories in majority of cases as these are built up dynamically “on the fly” and dependent on the shape and structure of the operating environment at a time.

As the *Freely Moving Vehicles Avoidance (FMVA)* stand for a general case, previous feature of no predetermined motion trajectories allows us to classify the possible approaches to solutions of the problem into two base categories as:

1. *ExPost Collision Avoidance (EPCA)* methods which belong mainly to classical approaches of how to resolve collision situations after their appearance, or whenever their occurrence is predetermined by the vehicle behavior (and therefore can be recognized in advance). The EPCA methods are always used to correct existing and/or foreseen collision situations.
2. *Predictive Collision Avoidance (PCA)* methods are located on the edge of planning approaches, which are extended towards cooperation of planning procedures for separate vehicles. The predictive collision avoidance is typically capable of extinguishing collisions due to insufficiently coordinated plans for diverse autonomous entities. These methods are mainly applied prior to execution of the planned trajectory of the vehicle or robot.

The former EPCA methods rely of the knowledge of the robot path, which is typically predetermined either by technical constrains or pre-computed by a path planner. So far, the situation may be similar to the previous case with cable or rail-guided vehicles. The suitable collision resolution approaches can therefore be similar to the previous case. The advantage of the EPCA methods is seen in limited need of the vehicle to stick necessarily on the pre-planned trajectory (like on a guiding rail), but allows it to perform collision avoidance maneuver anywhere on its trajectory (if not constrained by other obstacles). This situation does not require relying exclusively on avoidance methods based on *scheduling* of vehicles on trajectory vertices. Moreover, performing the meeting maneuvers whenever needed allows the vehicle to achieve much higher efficiency of the collision avoidance (shorter paths, less traffic congestions in bottlenecks, no delays due to vehicle wait-states, etc.) and therefore improved efficiency of path planning procedures as well.

The EPCA approach sketched above requires executing the meeting maneuvers that can generally occur either in case of a vehicle colliding with an unexpected obstacle or in case of two independent, non-communicating vehicles. The former situation can be handled by a simple reactive behavior of the vehicle control system, for instance by executing a wall following procedure until the original path is reached. The latter case requires more sophisticated approach. Meeting of two independent autonomous vehicles brings together two autonomous behaviors which are bound together only through observations of the common environment (that can include also other entities). It is straightforward that design of such a behavior needs to avoid falling into an endless loop or a deadlock when trying to resolve the meeting maneuver. All previously mentioned situations denote so called *non-cooperative collision avoidance*.

An alternative approach bounds both the behaviors together by a communication link. This enables to perform proper negotiation procedure [6] to ensure correct behavior by a *cooperative collision avoidance procedure*. The most common solution of this kind incorporates the *master-slave* approach in which all the entities negotiate priorities and in a stepwise manner undertake top-down decisions and resolve possible deadlocks. Extension of the preceding situations for multiple obstacles and/or mobile entities is then easy.

On the other hand, the PCA approaches are bidding to prevent possible collisions already in the phase of a path planning. As this kind of behavior is generic for a standard path planner and builds an inseparable functional part of an autonomous

vehicle, execution of the path re-planning procedures during the autonomous vehicle mission differs. The dynamic re-planning in result tends to respond to the current state of the operating environment as well as to the current level of knowledge about the environment. Lack of these knowledge or a certain change in the environment status may then invoke re-planning, representing a goal oriented behavior. This can typically include also avoidance of collisions of expected and unexpected sort. If the collision is prevented by a proper plan, it has been the expected one (derivable from the current level of knowledge about the environment). The other cases cover the unexpected ones, which could not have been discovered from the so far existing data or behavioral intentions of the other entities present. The latter can be corrected by execution of the re-planning procedure with the goal to update the vehicles' behavior with respect to last discovered facts.

In general case, the PCA approaches can be used to fulfill reactive behaviors if executed in a fast loop, but also to resolve cooperative tasks and scheduling problems in long term. This is widely used to ensure cooperated and coordinated behaviors for multiple mobile entities.

Another class of tasks that belongs to the fundamental test-bench problems of intelligent robotics and that is similar to collision avoidance problems is the task of *cooperative exploration* of an unknown environment by multiple robots. One of the problems of exploration is the creation of the map of the unknown environment while optimizing available resources. The resources can be for instance the task execution time, total trajectories driven, minimum energy costs, number of resolved collisions, percentage of explored space overlaps, etc.

Having a common map containing actual shape and structure of the surrounding environment as well as positions of all the participating robots, following questions arise:

- Where the places of interest are – places, which visiting provides the best profit for understanding of the environment and which consideration in the following tasks can lead to best possible performance.
- How to assign the exploration of the places of interest to particular robots so that the possible collisions and overlapping of explored areas is kept to minimum in order to achieve the best efficiency of the system performance with respect to the overall cost function.

The cost function is in the frontier-based approach [15] usually expressed as the distance of a robot to selected place of interest and its distance to other candidate places selected by other robots [1]. If an observation gain of visiting the selected place by the robot is high, the profit of visiting the neighboring places by other robots is decreased. The assignment of robots to particular places of interest, called *frontiers*, can be viewed as multi-robot and multi-goal optimization problem, which is well-suited subject for the agent-based approach. In the proposed solution, each physical robot is represented by a single agent. The negotiation-based algorithm is initiated by a *triggering* agent recognizing significant event like detection of close-to-collision state, substantial change in the observed part of the environment or simply a time stamp. First, all the agents compute their utility values for all known frontiers and share this knowledge with the others. A single robot-frontier pair with the

best cost function is determined and excluded from the subsequent rounds of negotiations. The number of negotiation rounds obviously corresponds to number of participating robots. Although, the given procedure does not provide generally optimal solution to the problem (which remains *NP-hard*), experimental results prove a good quality of the proposed method.

2.3 Collision avoidance in 3-D space

The distributed collision avoidance problem extended to third dimension is characteristic for the domain of unmanned aerial vehicles (UAVs). The work presented in [12] suggests multi-layer collision avoidance architecture based on multi-agent technology. Each UAV is controlled by a single agent wrapping several different avoidance algorithms. Both cooperative and non-cooperative methods are optimally combined for planning runtime trajectory of the UAV with respect to series of time-specific way-points. There is no central planner providing collision-free flight plans – the plans for each UAV are individually held and modified in case of detected collision by the UAV agents themselves.

The multi-layer avoidance collision architecture contains the CSM (Collision Solver Manager) as a main controller able to combine all the available cooperative and non-cooperative collision solvers. Each collision solver is responsible for both collision detection and collision solving. Different solvers provide different quality of the avoidance solution, but require different amount of time to find a solution. Thus, based on the priority of solvers, the CSM assigns specific time slot to each solver so that the strict time constraints are met.

Cooperative avoidance is based on interaction between airplanes that exchange their flight plans. Two different collision solvers are implemented – *rule-based* and *utility-based*. The former one is implemented according to the Visual Flight Rules defined by FAA (http://www.faa.gov) – it determines the collision type on the basis of the angle between direction vectors of aircrafts and then applies a predefined maneuver to avoid the specific collision. This is done by both planes independently because the second aircraft also detects a possible collision with the first plane.

The utility-based avoidance mechanism finds a solution for a pair of airplanes. One of the airplanes is regarded as a master entity (usually the first one that identifies a collision) and the second one as slave entity. The slave planning agent is requested by the master to generate a set of flight plan changes using different combinations of seven parameterized maneuvers – straight flight (no change), turn right, turn left, turn up, turn down, speed and slow down. The set is ordered according to the utility value computed for each configuration considering for example total length of the flight plan, time deviations for mission way-points, fuel status, possible damage, etc. The set is sent back to the master aircraft that generates its own set of plans and tries to combine both sets together to find a collision solution selected from Cartesian product of generated plans. The candidates for solution are again ordered in increasing manner by product of utility quotients of flight plan pair. Each pair candidate is tested for a collision and if there is no collision between participants, candidate is selected as collision solution. The slave entity is then requested to change its flight plan according to selected collision solution.

Non-cooperative avoidance is applied when the communication between planes is not possible for instance due to broken communication module of the aircraft. The proposed algorithm is based on the *dynamic no-flight zones* computed for each object detected by the onboard radar. First, the collision point is determined as intersection of the current UAV's flight plan and the predicted flight trajectory of the non-cooperative object. Second, the collision point is wrapped by the no-flight zone, which shape is derived from object's possible future flight trajectory taking into account the minimal turning radius, maximal climbing/descending angle, etc. The UAV's planner module then tries to determine such flight path that doesn't intersect any detected no-flight zone using A* algorithm.

3 Experiments

The proposed agent-based solution of presented AGV transportation system is part of the Manufacturing Agent Simulation Tool (MAST) developed at Rockwell Automation Research Center in Prague [14]. This tool provides agent-based simulation and control solution for the material-handling domain, mainly the transportation of products and materials on the factory floor. Originally, the transportation using conveyor belts was considered – a library of agent classes representing basic material-handling components, like workstation, conveyor belt and diverter was developed. To enhance the capabilities of MAST to simulate AGV-based transportation, the AGV agent class, with capabilities described in Section 2.1, has been added.

Figure 2 shows a screenshot of the MAST tool with an example of AGV network connecting four workstations. When the "source" workstation agent needs to deliver a product to another "destination" workstation, it first queries the Directory Facilitator to obtain a list of AGV agent names. Second, it initiates the contract-net-protocol in which the AGV agents give their bids in terms of time estimate for moving from current position to the source workstation plus from source to destination workstation. The source workstation selects the best bid and awards the contract to selected AGV agent. Several experiments with different number of AGVs working under different conditions (blocked paths, failed AGVs) have been carried out. It has been shown that solving head-to-head deadlocks might become very complicated if there are multiple AGVs involved at the same time (not only two as described earlier) and that a simple, but efficient solution is to fix a direction in which the AGVs can move in each segment.

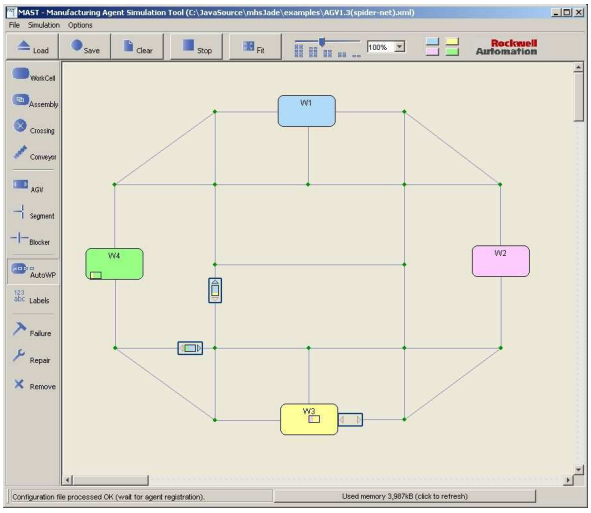

Figure 2: AGV-based transportation system with four workstations and three AGVs as part of the Manufacturing Agent Simulation Tool.

The presented robot exploration algorithm was implemented in the Gerstner Laboratory at the Czech Technical University in Prague and experimentally verified with a squad of four fully autonomous mobile robots (see Figure 3). RCS control architecture [3] was used to control the robots, while communication among them was performed by means of A-globe multi-agent platform [11]. Furthermore, SICK LMS 200 laser rangefinder stood as a main sensor for gathering information about the working environment and for collision detection and avoidance.

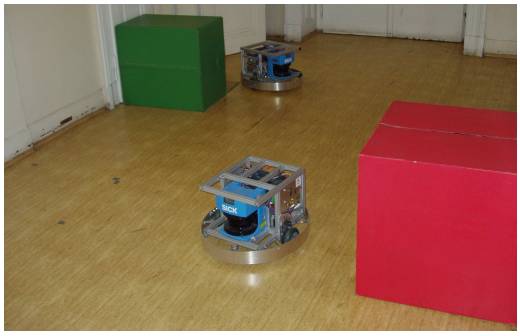

Figure 3: G2bots during exploration

The tests were performed in both simulated and real environments with different numbers of robots varying from two to four. Numerous frequencies of negotiation procedure (i.e. the process of generation, evaluation, and assignment of frontiers) were evaluated during these tests. The best results were obtained for frequency 5 seconds. Generated paths are smooth, while probability of collision is low for this value. Figure 4 shows a typical run – explored environment and robots' trajectories in an early stage of exploration (left) and the same after exploration completed. No-

tice that although the violet and yellow trajectories intersect several times, no collisions need to be solved because the robots moved to cross points in different stages of exploration.

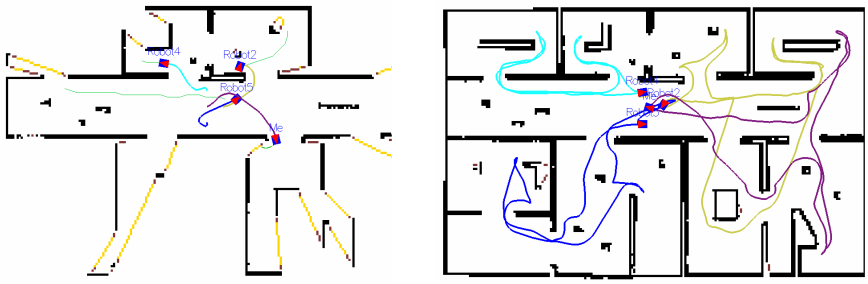

Figure 4: Exploration of an unknown space by four robots: detected obstacles (black), found frontiers (yellow), unreachable frontiers (brown), actual plans (green), already traversed trajectories (thick curves).

The multi-layer collision avoidance architecture presented in the section 2.3 has been evaluated on huge set of experiments. The experiments have been carried out using multi-agent system simulating air traffic built on to of A-globe platform [12]. The simulation system is extended to integrate several external data sources. One of them provides position of civil real aircrafts (10 minutes delayed) operating in selected area in the U.S. Randomly generated UAVs controlled by agents fulfill their missions in the same air space. Simulated UAVs are configured to use utility-based cooperative collision avoidance method with other agent-based UAV (can communicate with it) and use dynamic no-flight zones non-cooperative method against imported air-traffic (can use only position reported by on-board radar) at the same time.

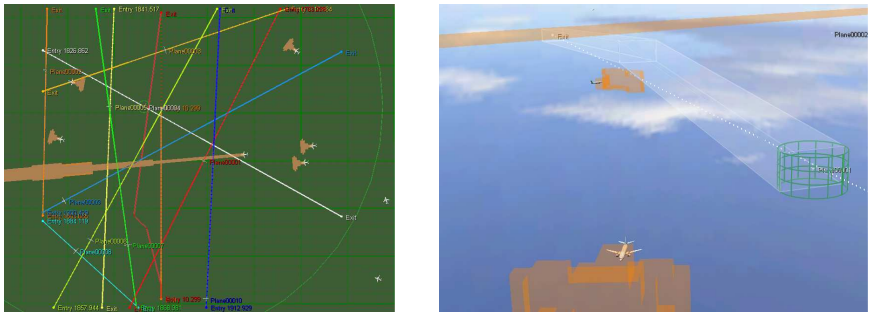

Figure 5: Operation of agent-controlled UAVs over LA with imported civil real air-traffic.

The screenshots from the simulation are shown in the Figure 5. Left screen is taken from the top view and provides small part of the simulated area. Right screen is three-dimensional view from one of the simulated aircraft. There is captured the situation when one of the agent-controlled plane monitors its local neighbor, builds

the prediction of future trajectory from the historical data and detects possible collision with civil plane. It just re-plan it flight corridor to not to collide with civil plane and re-negotiate new corridor with other cooperative (agent-controlled) UAVs.

4 Conclusion

Multi-agent technology provides good background for solving collision detection and avoidance issues. The collaborative approach seems to be highly desired – the agents representing vehicles can mutually inform themselves about their goals and intentions for future movement and thus can solve a possible collision much more efficiently than in non-collaborative case.

The following principles common for agent-based collision avoidance solutions can be identified:

1) *Master and slave*: when collision is detected, one of the vehicles is determined as a master while the second one is slave. Master then proposes a solution to avoid the collision and requests the slave to accept it.
2) *Collision metrics*: a utility value or cost of each candidate solution is computed and the most convenient one is adopted. Various negotiation protocols play an important role in exchanging this kind of information.
3) *Dynamic no-access zones*: an estimated collision point with each detected non-cooperative object is surrounded by dynamically updated no-access (no-flight) zone and the trajectory is planned to avoid all the zones.

As expected, avoiding collisions in 2D is much easier than in 3D. In the simplest case, the vehicle can stop itself and wait until the colliding vehicle passes the estimated collision point (as in case of AGVs described in Sect. 2.1); in a collaborative scenario, the master vehicle can command the slave to drive away.

The presented multi-layer architecture for generating collision-free flight corridors in 3D space is designed to optimally combine different cooperative and non-cooperative methods for planning and re-planning runtime trajectory of the UAV in the fully distributed manner. The cooperative methods are used to avoid the mutual UAVs collisions while the non-cooperative one based on dynamic no-access zones help to avoid other non-cooperative aircrafts. We envision that the application of this complex architecture also in 2D scenarios will offer even more sophisticated and efficient methods to detect and resolve collisions.

Acknowledgements

The support of the Ministry of Education of the Czech Republic, under the Project No. 1M0567 to Miroslav Kulich is also gratefully acknowledged.

Effort on three-dimensional collision avoidance is sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3073[1].

[1] The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions con-

References

1. Burgard W., Moors M., Stachniss C., and Schneider F.: Coordinated Multi-Robot Exploration. In: IEEE Transactions on Robotics, 21(3) (2005) 376-378.
2. Bussmann S, Jennings NR, Wooldridge M.: Multiagent Systems for Manufacturing Control: A design Methodology. In: Ishida T (ed), Springer-Verlag Berlin Heidelberg (2004)
3. Chudoba, J., Mázl, R., Přeučil, L.: A Control System for Multi-Robotic Communities. In ETFA 2006 Proceedings. Piscataway: IEEE (2006) 827-832.
4. Department of Defence, US: Unmanned Systems Roadmap 2005-2030 (2005).
5. Fletcher, B.: Autonomous Vehicles and the Net-Centric Battlespace. International Unmanned Undersea Vehicle Symposium, April 2000, Newport, RI, USA.
6. Fujimori, A., Teramoto, M., Nikiforuk, P., and Gupta, M.:Cooperative Collision Avoidance Between Multiple Mobile Robots. Journal of Robotic Systems, 17(7) (2000) 347-363.
7. Mařík V., Pěchouček M., and Štěpánková O.: Social Knowledge in Multi-Agent Systems. In: Multi-Agent Systems and Applications, LNAI 2086, Springer-Verlag, Heidelberg (2001) 211-245.
8. Mařík, V., Pěchouček, M., Vrba, P., Hrdonka, V.: FIPA Standards and Holonic Manufacturing. Deen, S. M. (Ed.): Agent Based Manufacturing: Advances in the Holonic Approach, ed., Springer-Verlag Berlin Heidelberg (2003) 89-121.
9. Ortiz C.L., Vincent R, Morisset B.: Task Inference and Distributed Task Management in the Centibots Robotic System. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM (2005) 860-867.
10. Reveliotis, S. A.: Conflict Resolution in AGV Systems. In: IIE Transactions, 32(7) (2000) 647--659.
11. Šišlák, D., Rehák, M., Pěchouček, M., and Pavlíček, D.: Deployment of A-globe Multi-Agent Platform. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (2006).
12. Šišlák, D., Rehák, M., Pěchouček, M., Pavlíček, D., and Uller, M.: Negotiation-based Approach to Unmanned Aerial Vehicles. In: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, Washington, DC USA, IEEE Computer Society (2006) 279–284.
13. Vrba, P. MAST: Manufacturing Agent Simulation Tool. In Proceedings of IEEE Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal, Volume 1 (2003) 282-287.
14. Vrba, P., Mařík, V.: Simulation in Agent-Based Control Systems: MAST Case Study. In Proceedings of 16th IFAC World Congress, Prague (2005).
15. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA (1997) 146-151.

---

tained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.