Dynamic Collision Avoidance using Local Cooperative Airplanes Decisions

Augustin Degas, Arcady Rantrua, Elsy Kaddoum, Marie-Pierre Gleizes, Françoise Adreit

Dynamic Collision Avoidance using Local Cooperative Airplanes Decisions

Augustin Degas[1,2], Arcady Rantrua[1]
Sopra-Steria Group
Colomiers, France
Email: [1]{firstname.surname}@soprasteria.com

Elsy Kaddoum[2], Marie-Pierre Gleizes[2],
Françoise Adreit[2]
IRIT, Université Toulouse 3
Toulouse, France
Email: [2]{firstname.surname}@irit.fr

***Abstract*—Air Traffic Control (ATC) of the future will have to cope with a radical change in the structure of air transport [1]. Apart from the increase in the traffic that will push the system to its limits, the insertion of new aerial vehicles such as drones into the airspace with different flight performances will increase the heterogeneity level. Today's research works aim at increasing the level of automation and partial delegation of the control to onboard systems. In this work, we investigate the collision avoidance management problem using a decentralized distributed approach. We propose an autonomous and generic multi-agent system to address this complex problem. We validate our system using state of the art benchmarks. The obtained results underline the adequacy of our local and cooperative approach to efficiently solve the studied problem.**

***Keywords*—Trajectory optimization, Automation strategies, Conflict resolution, Self-separation, Multi-Agent system, Self-organization**

I. Introduction

Contrary to a clear majority of motion planning problems, the difficulty of motion planning in air traffic management do not yield in finding a trajectory for one aircraft. Airspace is rarely cluttered by obstacles, except for the weather, thus finding a trajectory is pretty much straightforward. The difficulty of motion planning for aircrafts reside in finding a feasible trajectory for each aircraft (*i.e* respecting the capabilities of the aircraft), collision free with other aircraft, globally optimal (*i.e.* optimal for all the airplanes), and resilient to changes and uncertainties, in a wide configuration space [2].

In todays air traffic management, airspace is divided into several zones each under the supervision of air controller. In order to help air controllers to manage real time traffic and avoid collision, the traffic is regulated upstream. With the increase in the traffic and the insertion of new aerial vehicles such as drones into the airspace with different flight performances, the air traffic control must evolve by increasing the level of automation and introducing partial delegation of the control to on-board systems.

In this work, we are interested in this particular kind of motion planning in a clear environment, with dynamic constraints, with multiple vehicles and an objective of optimality. This kind of motion planning are mostly about following the desired trajectory while avoiding collisions with other entities. We propose a generic approach and do not focus on the navigation of multiple robots to a single destination or the control of an swarm formation like in [3]–[5].

We propose to address the collision avoidance management problem using a decentralized distributed approach : the AMAS theory. It aims at solving problems in dynamic environment by a bottom-up design of autonomous agents, where cooperation is the engine of the self-organization process [6]. AMAS approach has been successfully used to solve different problems, as anomaly detection in maritime environment [7], control and optimization of heat-engine [8] , or context learning [9].

In the case of the collision avoidance problem, representing each airplane by an autonomous cooperative agent with local interactions, brings a natural decentralized solution to the complexity of the global problem. The system we propose, can be used for several issues:

- Simulation: the system can be used to measure the consequences of the introduction/modifications of airplanes trajectories.
- Learning: the openness of the system, allows real-time interactions with end-users. This can help for education purpose. Indeed scenarios can be defined where the control to avoid collision is given to air traffic controller in specific sectors and then taken back by the system with the modified trajectories. The solution proposed by the controllers can also be compared to the one proposed by the system.
- On-board: the system can be used by air traffic controller or directly on-board as a decision support system to avoid collisions.

The remainder of this paper is structured as follows. Section II briefly reviews related work. Section III presents our general approach for collision avoidance. Section IV describes the application of the approach to ATC, the experiments and the results. Finally, Section V summarizes our findings and conclude this study.

II. Related work

A. Problem formalization

Motion Planning, planning of trajectory of a set of mobile entities, is a widely studied field, from the planning of a path of a simple robot in a unknown environment to the planning of trajectories of a set of mobile entities with constraints in a known environment. By mobile entity, we mean every possible entity that can move, from a robot arm, to cars, Unmanned

Aerial Vehicles (UAV) or airplanes. Our concerns in this vast field of motion planning lies in collision avoidance of a set of mobile entities, with dynamic constraints, and an objective of optimally.

We based our formalization on the one proposed in [2] that introduces the notion of Configuration space. This notion was first introduced for path planning of a simple mobile, in order to represent the space of every possible state in which the mobile entity can be. It is easily generalized for motion planning of multiple mobile entities or a mobile entity with multiple parts, it only had parameters related to the state vector (position, speed, direction). A configuration from the configuration space is then a vector containing a state for every $M_i \in M$ (or every part of M_i). In the remainder of this paper, the configuration space is noted C, a configuration from the configuration space is noted q. In C, some configurations q are in collision with an obstacle. We note the space of configurations with collisions with obstacles C_{Obs} and the space without collisions C_{free}, thus $C = C_{Obs} \cup C_{free}$.

Our problem can be formalized as follow:

- A set of heterogeneous mobile entity, noted *M*, $M = \{M_i\}_{0<i\leq m}$
- A set of obstacles, O. An obstacle can be fix or can move in time.

Each mobile entity is characterized by:

- A state vector, containing the position vector of M_i, it's velocity vector, and the 3 rotation angles (Euler's angles).
- A predetermined trajectory, noted τ_p, is a function of time.
- Capacities, containing it's ability to accelerate, decelerate, turn and so on.
- A configuration space C_i, in which other M_j are obstacles with determined trajectory.

The problem consists in finding a trajectory τ_i for each mobile entity from a starting point, to a goal destination, while respecting τ_p and avoiding collisions with mobile and stationary obstacles. In other terms, the goal is to find for each M_i a trajectory τ_i that lies in $C_{i,free}$.

B. State of the Art

The studied problem is combinatorial : the huge numbers of heterogeneous airplanes, the largeness of the airspace and the forth dimensions (Space+Time) make the configuration space C tremendous. The size of the configuration space has led most researchs to discretize the configuration space, by discretizing the maneuvers or the airspace, and explore it with graph search or evolutionary algorithms or by using heuristics to guide the exploration.

Meta-heuristics using discretization of maneuvers, like genetic algorithms [10] or ant colony algorithms [11], along with artificial intelligence algorithm as neuronal networks [12]. Those methods give interesting results still they scale poorly. Indeed, [11] and [10] are one of the few that handle more than 20 airplanes and find a global optimum.

Potential fields are also expensively studied, starting from the standard method, to its extension with navigation functions [13], the usage of more complex potential fields [14], or the combination between potential fields and swarming [15]. Those methods have the particularity of providing a proof of convergence. However those methods need to be tuned carefully to be effective. Finding generic rules to do so seems difficult.

Mixed-Integer Linear Programing (MILP) solvers are studied as well, in particular for solving a minimum weight maximum clique model [16]. Results are interesting, but they use important instantaneous heading or speed changes. Constraint programming has also show some interesting results [17], finding solutions proved to be optimal, but scale badly with the number of airplanes.

Geometrical approaches have also been studied. The idea is to detect collisions using velocity vectors, compute the minimal velocity vector change to avoid the collision and divide equally the minimal velocity vector change among the mobile entities [18], [19]. One algorithm in particular has been successfully applied to multi-robot [20] and some adaptations for aircrafts have been made [21]. Most of them have not been tested in dense situations. The last ones however are interesting in the way they decentralize the problem among the different entities and give interesting results with dense situation as long as the constraints on maneuvers are light.

Different studies move towards decentralization and multi-agent systems approaches [22], with collision avoidance based on velocity changes and departure delay. Those techniques allow a natural description of the problem, and have shown their adequacy to efficiently solve complex problems addressing dynamic and scaling issues. We believe that this decentralization may grow interest in the future as part of the responsibility for separation maintenance will be delegated to the aircraft [1].

III. Collision Avoidance using Adaptive Multi-Agent System Approach

In this part, we start by introducing the Adaptive Multi-Agent Systems (AMAS), then we present our general approach, called CAAMAS for Collision Avoidance Adaptive Multi-Agent System.

A. Adaptive Multi-Agent Systems Theory

Multi-Agent Systems (MAS) are composed of different entities called agents. An agent is a physical or software entity, that is autonomous, evolves in an environment with perceive, decide and act abilities. The agent has a partial perception of the environment, is able to communicate with other agents, has its own resources and capacities, and can offer services. An agent follows a life cycle composed of three steps, repeated indefinitely: Perception, Decision, Action. During the perception, it acquires new information about its environment. It then decides during the decision phase the next action to perform. Then it realizes the action decided in the decision phase. By the agents local interactions, a self-organization is established making the solution emerges.

In some cases, an agent may be noncooperative, which means it may bother other agents in their task, or it cannot

fulfill its goal and thus cannot help the group at all. In the AMAS approach, the cooperation among agents interactions is the engine of the self-organization. The AMAS approach aims to create MAS in which agents act cooperatively between themselves in order to maintain a cooperative behaviour. The AMAS theory identifies seven generic non cooperative situations [23] among the three steps of the life cycle (perception, decision, action). In such situations, the agent based on criticality information decides cooperative actions in order to solve difficult situations (conflict, concurrence, ambiguity, etc.)

B. The CAAMAS approach

We introduce in this section our decentralized Collision Avoidance Adaptive Multi-Agent System (CAAMAS) approach, for collision avoidance management for multiple heterogeneous mobile entities with dynamic constraints. In our model, every M_i is represented by an agent following the life-cycle described in Algorithm 1.

Algorithm 1 Life-Cycle of an agent M_i along its trajectory

repeat
  Perceive : Store the criticalities of mobile entities (neighbors) in its perception zone noted Zp_i;
  Decide : Compute the **criticality** of each possible action and decide cooperatively the action that minimizes the criticality of its neighbors and its own;
  Act : Perform the decided action and inform neighbors of its new criticality
until M_i is arrived

1) Perception phase: Every M_i perceives its local environment defined by its perception zone Zp_i of the airspace. In this zone, M_i is able to perceive other mobile entities called its neighborhood.

Different communication means can be used in order to exchange information among the mobile entities. In our model, we use messages, still other means can be easily added. In the perception phase, M_i receives messages from other mobile entities M_j ($i \neq j$). Note that every M_j perceived by M_i belongs to Zp_i ($M_j \in Zp_i$).

Those messages contain two important parts :

- The **current situation** of the mobile entity: its position and velocity vectors. They will be used to determine some criticalities, in particular the collision criticalities.
- The **criticality of the sender**, $Crit_j$. This criticality is the criticality computed for the action the mobile M_j is currently doing. It is used by M_i to determine its behavior regarding M_j.

During the perception phase, the agent only stores the perceived information and uses them in the decision phase.

2) Decision phase and Action phase: In the decision phase, the mobile entity decides cooperatively, based on its evaluation of the current situation, which **action** to perform at the action phase.

a) Action: Given its capacities a mobile entity M_i can realize at each step a set of finite actions in order to explore the configuration space C_i (cf. section II-A). In a forth dimension C (Space+Time), those actions can be for example to climb, descend, stay put, turn left, turn right, accelerate or decelerate. We note Ac_i the set of n actions that M_i can do, $Ac_i = \left\{Ac_{i,k}\right\}_{0<k\leq n}$. For each possible action, the agent associates a criticality.

b) Criticality: For an agent, criticality represents the degree of non-satisfaction of its own goal [24]. We note $Crit_i$ the criticality of the mobile M_i. $Crit_i$ might be a simple real, or a tuple of different measures. In our model, the criticality of a mobile entity M_i is a couple, $Crit_i = (Crit_{i,coll}, Crit_{i,traj})$:

- **Collision criticality**, noted $Crit_{i,coll}$ which represents the degree of non-satisfaction of M_i regarding the objective of avoiding collisions.
- **Trajectory criticality**, noted $Crit_{i,traj}$ which represents the degree of non-satisfaction of M_i regarding the objective of following its desired trajectory.

Those criticalities are computed for every possible action $Ac_{i,k}$ as shown in algorithm 2. The algorithm starts by evaluating the collision criticalities for each $Ac_{i,k}$ regarding each M_j in the perception zone, $Criti, j, k, coll$. The calculated criticalities are stored in a collision criticality list $collCL_{Ac_i,k}$ ordered by $Crit_j$ associated to a $Ac_{i,k}$. Then, the trajectory criticality if M_i performs $Ac_{i,k}$, $Criti, k, traj$, is evaluated.

To sort the criticalities, the algorithm first considers the collision criticality. The trajectory criticality is considered in case of equal collision criticality or in case no collision is detected.

Algorithm 2 Evaluation of the criticalities for each $Ac_{i,k} \in Ac_i$

Sort $Crit_j$ of the M_j stored at the perception phase
for all $Ac_{i,k} \in Ac_i$ **do**
  Initialize $collCL_{Ac_{i,k}}$ associated to $Ac_{i,k}$
  for all perceived $Crit_j$ **do**
    Evaluate $Crit_{i,j,k,coll}$ of M_i regarding M_j if M_i does the action $Ac_{i,k}$
    Store the criticality in $collCL_{Ac_{i,k}}$
  end for
  Evaluate $Crit_{i,k,traj}$ of M_i if it performs action $Ac_{i,k}$
end for

Evaluation of the collision criticality of an action $Ac_{i,k}$: The evaluation of the collision criticality is based on an extrapolation called nominal projection in the literature [25]. It is a simple extrapolation of position and the velocity vectors. The idea is to consider that the situation of a mobile entity M_j will evolve in the same way it does currently (same speed, same direction) and thus, to calculate the criticality of the possible occurrence of a collision if M_i realizes $Ac_{i,k}$.

Considering that, the collision criticality of M_i if it does $Ac_{i,k}$ regarding M_j, $Crit_{i,j,k,coll}$ is an ordered couple :

- $C_{1,k}$ representing the criticality regarding the minimal

distance than can be reached between M_i and M_j if M_i realizes $Ac_{i,k}$.

- $C_{2,k}$ representing the criticality regarding the time at which the minimal distance between M_i and M_j if M_i realizes $Ac_{i,k}$ occurs.

In order to calculate this couple, the algorithm determines the minimal distance ($d_{min,i,j,k}$) between M_i and M_j if M_i does the action $Ac_{i,k}$, and then the time $t_{min,i,j,k}$ at which the minimal distance occurs. In case a collision is detected, the interval $[t_{startCollision}, t_{endCollision}]$ of time during which the collision occurs is computed. In this case, the $t_{startCollision}$ is considered instead of the $t_{min,i,j,k}$.

In the following, we note $||.||$ the euclidean norm, and $d_{i,j,k}(t)$ the distance between M_i and M_j if M_i does the action $Ac_{i,k}$. With those notations and the previous hypothesis, we have:

$$d_{min,i,j,k} = \min_{0 \leq t}(d_{i,j,k}(t)) = \min_{0 \leq t} ||\overrightarrow{p_{i,k}}(t) - \overrightarrow{p_j}(t)||$$

With $\overrightarrow{p_{i,k}}(t)$ the vector position of M_i if it does the action $Ac_{i,k}$, and $\overrightarrow{p_j}(t)$ the perceived position vector of M_j.

Since the speed vector is considered as constant, $d_{min,i,j,k}$ is computed using:

$$d_{min,i,j,k} = \min_{0 \leq t} ||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}) - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})||$$

The value at which the derivative of $||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}) - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})||$ is null, is then $t_{min,i,j,k}$:

$$t_{min,i,j,k} = \frac{-(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})}{||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||}$$

Note that:

$$d_{min,i,j,k} = d_{i,j,k}(t_{min,i,j,k})$$

Based on both values, $C_{1,k}$ and $C_{2,k}$ are then calculated as follow:

$$C_{1,k} = \begin{cases} 100 - \frac{100}{2d_{coll}} d_{min,i,j,k} & \text{if } d_{min,i,j,k} < 2d_{coll} \\ 0 & \text{if } d_{min,i,j,k} \geq 2d_{coll} \end{cases}$$

Where d_{coll} is the distance at which two mobiles should be at least from one to another. Figure 1 illustrates the computation of $C_{1,stayPut}$ for M_1 regarding two mobile entities M_2 and M_3 in its perception zone.

$$C_{2,k} = \begin{cases} 100 & \text{if } t_{min,i,j,k} < t_{tr} \\ 100 - \frac{100}{2t_{tr}}(t_{min,i,j,k} - t_{tr}) & \text{if } t_{min,i,j,k} \in [t_{tr}, 3t_{tr}] \\ 0 & \text{if } t_{min,i,j,k} \geq 3t_{tr} \end{cases}$$

Where t_{tr} is the time required by the mobile to cross its perception zone.

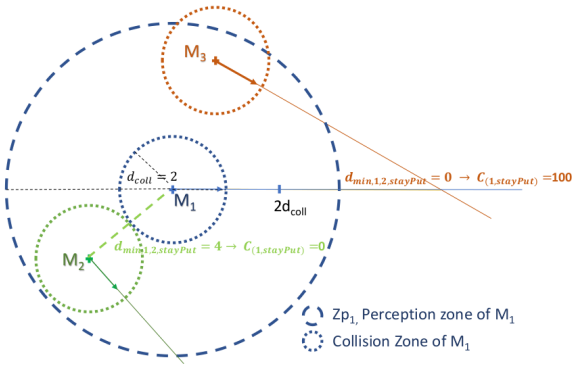

Fig. 1. Main notations used in criticalities computation

Evaluation of the trajectory criticality of an action $Ac_{i,k}$: The trajectory criticality is composed of three measures, $Crit_{i,k,traj} = (C_{3,k}, C_{4,k}, C_{5,k})$ where :

- $C_{3,k}$ is the distance to the predetermined trajectory τ_p at the next step (t+1) if $Ac_{i,k}$ is performed at the next step (t+1)

$$C_{3,k} = \min_{\overrightarrow{x} \in \tau_p} \{||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{x}||\}$$

- $C_{4,k}$ is the distance to the position of the destination $\overrightarrow{p_{goal}} \in \tau_p$ if $Ac_{i,k}$ is performed.

$$C_{4,k} = ||\overrightarrow{p_{i,k}}(t = \tau) - \overrightarrow{p_{goal}}||$$

- $C_{5,k}$ is the angle α_k between the predetermined trajectory τ_p and the speed vector if $Ac_{i,k}$ is performed,

$$C_{5,k} = \begin{cases} 0 & \text{if } \alpha_k < \frac{\pi}{2} \\ \frac{100}{\frac{\pi}{2}} \times (\alpha_k - \frac{\pi}{2}) & \text{if } \alpha_k \geq \frac{\pi}{2} \end{cases}$$

The three measures are used by the agent regarding the current situation. Two cases are to be distinguished:

- Previous trajectory modifications lead the agent to deviate from its predetermined trajectory τ_p keeping the same direction, the agent compares the trajectory criticality between two possible actions using first $C_{3,k}$, then $C_{4,k}$ (for equal $C_{3,k}$) and finally $C_{5,k}$ (for equal $C_{3,k}$ and $C_{4,k}$).
- Previous trajectory modifications lead the agent to deviate from its predetermined trajectory τ_p with opposite direction, the agent compares the trajectory criticality between two possible actions using first $C_{5,k}$, then $C_{3,k}$ (for equal $C_{5,k}$) and finally $C_{4,k}$ (for equal $C_{5,k}$ and $C_{3,k}$).

Decide which action to perform: After evaluating the criticalities of all $Ac_{i,k} \in Ac_i$ the agent cooperatively decides which action to perform in order to help the most critical agents. The decision process presented in Algorithm 3 distinguishes between two cases: a collision is detected or not.

In case of collision detection, the idea is to consider the criticalities of every $M_j \in Zp_i$ from the highest to the lowest, and determine, using the criticality collision lists $collCL_{Ac_{i,k}}$ of each $Ac_{i,k}$ computed in algorithm 3, which action(s) can help the most.

If several actions can be done to help most critical agents or no collision is detected, the agent decision is based on the trajectory criticality as defined above.

Algorithm 3 Decide

Create and initialize a list l_{act} with every $Ac_{i,k}$
if Collision detected **then**
  while $length(l_{act}) \neq 1$ and not all $Crit_j$ considered **do**
    Select M_j with the highest $Crit_j$
    Using $collCL_{Ac_{i,k}}$, find the set of actions $a \subset Ac_i$ that improve the most the criticality with M_j
    Remove from l_{act} every $Ac_{i,k} \notin a$
    Remove $Crit_j$ from not considered criticalities
  end while
end if
Choose the $Ac_{i,k} \in l_{act}$ that reduces the most the trajectory criticality, $Crit_{i,k,traj}$.

Note that, for a given mobile entity, the most critical agent might be an agent $M_j \in Zp_i$ or itself.

Once the action $Ac_{i,k}$ to perform is decided, the agent determines its criticality as

$$Crit_i = (\max(collCL_{Ac_{i,k}}), Crit_{i,k,traj}) \quad (1)$$

Action phase The action part for the agent is more or less straightforward. M_i does the action decided by algorithm 3, and sends messages containing its current situation and its criticality to the mobile entities inside its perception zone.

IV. Experiments and Results

For the experimentation, we apply our model to Air Traffic Management (ATM). Mobile entities are then airplanes each with realistic characteristics.

A. Air Traffic Control

We briefly introduce here some characteristics of the Air Traffic Control (ATC) which explains the instantiation of the CAAMAS approach.

In ATC, airplanes are separated for safety by a protection zone determined using different human and material factors. The protection zone is a cylinder oriented vertically with a height h of $1000feet$ ($1000ft = 304.9m$) and a radius r of 5 nautical miles ($r = 5NM = 9.26km$)

Airplanes fly from geographical points to geographical points, called Waypoint. Their trajectory from one airport to another is then reduced to a list of waypoints, called flightplan. Experience shows that they don't always follow their flightplan [26] for different reasons such as controller orders. In our study, we consider that their predetermined trajectory, τ_p, is approximated by a broken line.

Airplanes are increasingly capable of communicating data to each other such as their positions, heading or speed by means of messages. Messages can be transmitted using Automatic Dependant Surveillance-Broadcast (ADS-B).

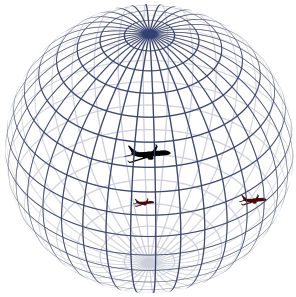

Fig. 2. Perception zone of the black airplane (at the center of the sphere), perceiving two other airplanes

B. Instantiating CAAMAS

a) Perception zone: The neighbourhood of the airplane is a sphere centered on it of radius r_{Z_p} of 100 nautical miles ($100NM = 182,5km$). We consider that every obstacle O_j (mobile M_j or stationary) in the sphere is perceived by M_i. This perception zone is illustrated by 2. In the conducted experimentation, only mobile obstacles M_j were considered.

b) Collision detection: In ATC, two airplanes are considered in collision whenever a violation of the protection zone is detected. Then in the following, a ATC loss of separation will be refereed as a collision, and is considered as a collision by CAAMAS. In order to take into account the fact that the vertical distance is not on the same scale as the horizontal distance, the collision criticality is calculated on the horizontal plan and the vertical plan. Thus, we have two values for $d_{min,i,j,k}$: $d_{min,i,j,k,h}$ and $d_{min,i,j,k,v}$, and two values for time $t_{min,i,j,k}$ (or an interval): $t_{min,i,j,k,h}$ and $t_{min,i,j,k,v}$ with every other M_j for every $Ac_{i,k}$. Then, the collision criticality $Crit_{i,j,k,coll}$ regarding M_j for action $AC_{i,k}$ is calculated as follow:

- $C_{1,k} = min(C_{1,k,v}, C_{1,k,h})$
- $C_{2,k} = min(C_{2,k,v}, C_{2,k,h})$

Where $C_{1,k,v}$ and $C_{2,k,v}$ being respectively the equivalent of the previous $C_{1,k}$ and $C_{2,k}$ for the vertical plan, and $C_{1,k,h}$ and $C_{2,k,h}$ for the horizontal plan.

c) Actions: Airplanes modify their speed, by accelerating or decelerating, but also change their heading, and modify their altitude as well. These actions have a fixed parameter like an acceleration rate or a turning rate. We consider that they can realize those three changes in the same time, which means that at every step of $\Delta t = 1s$ an airplane has to choose between 27 possibilities ($3 \times 3 \times 3$). These 27 possibilities of action results in 27 possible future positions represented in figure3. In this study, we only experiment the horizontal plan, so the number of actions available for the airplane is only of 9 (3×3).

Each airplane has a preferred cruise speed depending on general airplane performances and airline preferences, that we call v_{pref}. The airplanes are able to decelerate and accelerate within a speed range of $[v_{pref} - 6\%, v_{pref} + 3\%]$ which are considered as plausible values in real life [27]. It can accelerate and decelerate with speed modification of 0.33% at each step. In the experiment, we consider that airplanes have the same v_{pref} and that an airplane can modify its heading by $3°.s^{-1}$ [28, p. 18], which makes a complete 360° turn in 2 minutes.

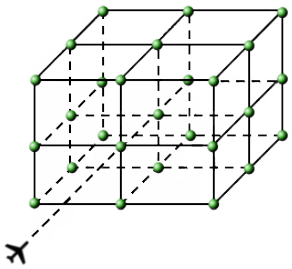

Fig. 3. The possible actions of the airplane (not to scale)

C. Benchmark

We experiment our approach with two benchmarks, a roundabout and a random case like in [29]. We experimented with several numbers of agents $AgNb$, for the roundabout $AgNb \in \{6, 8\}$ and for the random benchmark $AgNb \in \{12, 20, 40, 52, 60, 80, 100, 120\}$.

In the roundabout benchmark, we experiment on a disk of radius $R = 125NM = 231,5km$. At the beginning of the experiment, all the airplanes are placed at the edge of the disk, and they are all converging toward the center of the disk with an angle of $\frac{2\Pi}{m}$ between them. For this experiment, only heading changes are authorized, and speed is normalized.

For the the random experiment, airplanes are equally placed at each side of the square of side $l = 500NM$. They are placed randomly and have a precise point on the opposite side of the square as destination like in the figure 4. The arrival and start time must be at a distance $d = 2d_{coll}$ from each other.

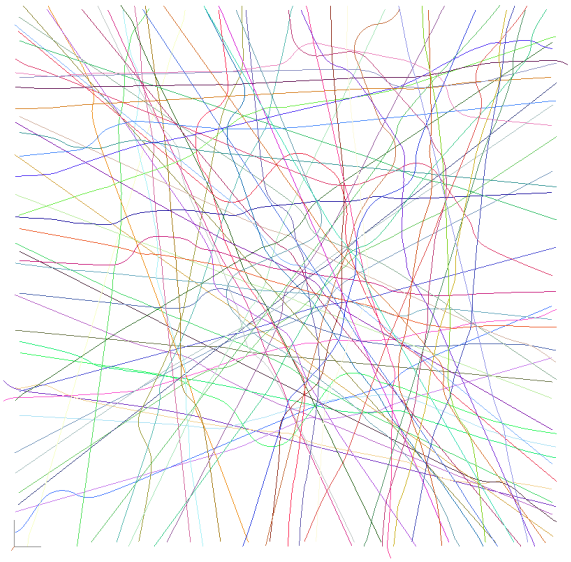

Fig. 4. Random experiment with 80 airplanes: 76 arrived and 0 collision

We evaluate our experiments regarding the number of predicted collision, the number of remaining collisions, the computation time, and the delays caused to airplanes. Comparison with the results obtained in [29] underline the advantages of our approach.

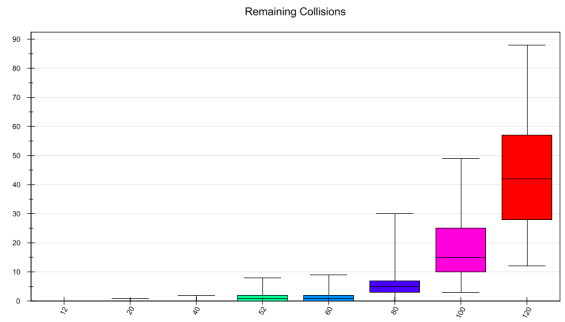

Fig. 6. The number of remaining collisions after the test

D. Results

In this section, separated results for the random and the roundabout experiments are presented. The tests were performed on a computer equipped with a 2.50GHz i7-6500 processor and 8 GigaByte of RAM. We implemented the algorithms in Java 8.

1) Random experiment: Our system has been tested 100 times per $AgNb \in \{12, 20, 40, 52, 60, 80, 100, 120\}$ on this random experiment. The evaluated metrics are presented each in separated figure as box-plot for every $AgNb$, with maximum value, third quartile, median, first quartile and minimum value. Table I summarizes the comparative study.

a) Predicted and remaining collisions: For each randomly generated scenario, we compute the number of predicted collisions (Figure 5), *i.e* the number of collisions that would occur if no change is made on the trajectories of the airplanes. We compare this number to the remaining collisions (Figure 6), that may be new collisions created by the system while avoiding others, or the same old collisions.

The obtained results show that in most cases more than 88.4% of the collisions has been solved (more than 99.12% for 40 airplanes). Still, detailed studies must be conducted in order to count the number of new collisions added by solving predicted collisions.

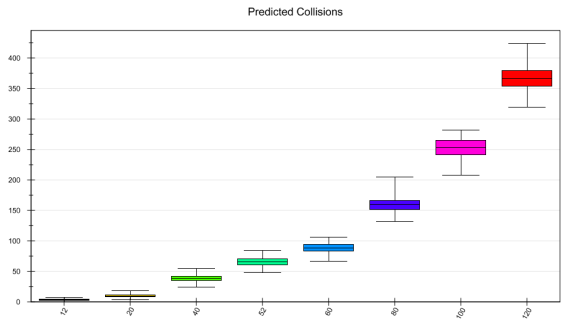

Fig. 5. The number of collisions predicted at the start of the test

b) Computation Time: Figure 7 shows the time the system uses to compute the trajectory of every airplane. Note that, for the benchmark with 120 airplanes, the algorithm takes less than 12s to compute the solution. In average, it takes 212ms, 418ms, 1305ms, 2048ms, 2826ms, 5072ms, 7660ms and 11891ms to compute the algorithm with respectively 12, 20, 40, 52, 60, 80, 100, 120 airplanes. Note that CAAMAS can be deployed on a distributed computation network which can drastically reduce the computation time.

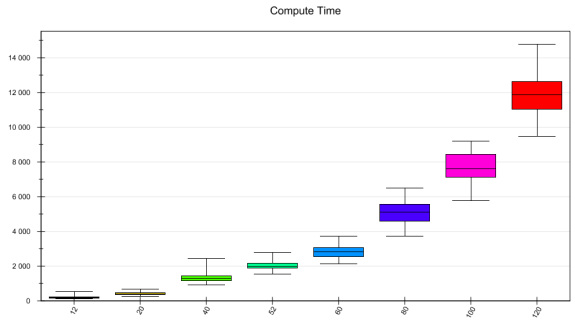

Fig. 7. Time to compute the solution in milliseconds

c) Delays caused to airplanes: Figure 8 shows the results concerning the delays caused to airplanes in order to avoid collisions. The results show than for each benchmark, more than 75% of the airplanes have very short delays. Still in high density benchmarks (*i.e.* 120 airplanes),some airplanes can be delayed significantly. This can be due to the limited 9 actions given to the airplane. Still, a detailed study will be conducted in order to understand the characteristics of such benchmarks (density, new generated collisions while solving predicted ones, etc.) and to be able to propose better trajectory for highly delayed airplanes.

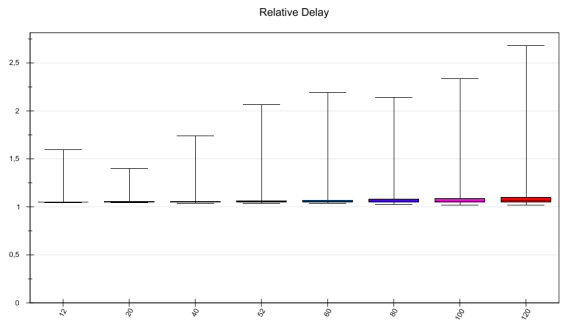

Fig. 8. Delays caused to airplanes

d) Comparative study: Table I summarizes the comparisons realized between CAAMAS and [29].

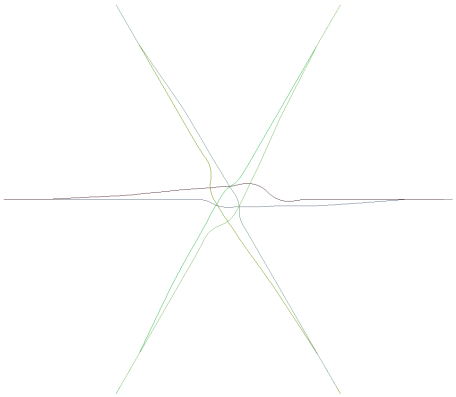

Fig. 9. Near optimal solution for 6 airplanes

	CAAMAS			Durand et. al. [29]		
nb acft	Rem Coll	mean delay	max delay	Rem Coll	mean delay	max delay
10	0	1.04504	1.58842	0	1.00094	1.00898
20	0	1.04745	1.56263	2	1.00431	1.03019
50	1	1.06049	1.91479	16	1.01376	1.03342
100	18	1.08122	2.05103	92	void	void
120	38	1.08897	2.39579	100	void	void

TABLE I. COMPARATIVE STUDY (DIMENSIONLESS MEASURES)

For the comparative study, the speed range has been set to $[v_{pref} - 5\%, v_{pref} + 5\%]$. Aircrafts can still accelerate and decelerate with speed modification of 0.33% at each step and modify their heading by $3°.s^{-1}$. We compare with the results of Table I and Table II from [29], with $s_{lat} = 0.05$ and $s_{long} = 0.05$.

For the slight scenarios (10 and 20 airplanes) presented in the Table I CAAMAS solves all the collisions, while the comparative algorithm is not able to solve some of them. When the number of airplanes increases too much (from 100 to 120), the number of collisions increases significantly for both, but the Durand et. al. [29] do not give delays since the algorithm cannot bring any aircraft to its destination. The last scenario with 120 airplanes could be considered as very overloaded because the collision number increases significantly. The max delay is quite high (from 50 to 140 %) compared to the Durand et. al. [29] or reality, and represent isolated aircrafts. Nevertheless the mean flight delay is not so high (10%).

2) Roundabout experiment: Figure 9 presents the obtained trajectories for the benchmark defined for 6 airplanes to simulate the roundabout experiment. Note that agents of CAAMAS only base their decisions on local information. The the usual pattern called roundabout emerges from their local interactions and cooperative behaviour.

V. CONCLUSION

The proposed collision avoidance system is a fully decentralized distributed approach based on adaptive multi-agent technology. We have shown its relevance from several criteria : efficiency of management even for dense traffic, limited amount of communication between airplanes and computation time. Moreover, it could eventually be implemented on board, removing the need to rely on ground equipment. Even significantly better of another one, CAAMAS is unable to solve

all conflicts for overloaded scenarios. And next research work will aim at reducing the number of conflicts by adding new behaviors to the adaptive multi-agent system.

We plan now to test our algorithm when some communications are lost, and adding in the scenarios non-cooperative obstacles, such as airplanes or weather. Testing other means of perception, like radar, would also be an interesting study case.

To deeply investigate the evaluation of CAAMAS performances, we intend to implement a standard global optimization method - used for solving similar problems - to solve the presented problem in order to compare the difference in terms of computation time and result optimality. As there is a priori no computed plan, we can assert that our method is fully adaptive face to unexpected event.

Eventually, CAAMAS could be used as a support decision system for air traffic controllers because it is able to work over different scales, time and space. This requires lot of experiments with data obtained from real air traffic.

ACKNOWLEDGMENT

The authors thanks Sopra Steria Group and the ANRT for their support in this research work.

REFERENCES

[1] M. Prandini, L. Piroddi, S. Puechmorel, and S. L. Brázdilová, "Toward air traffic complexity assessment in new generation Air Traffic Management systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. pp 809–818, Sep. 2011. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-01020894

[2] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[3] G. Antonelli, F. Arrichiello, F. Caccavale, and A. Marino, "Decentralized centroid and formation control for multi-robot systems," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3511–3516.

[4] C. Huang, X. Chen, Y. Zhang, S. Qin, Y. Zeng, and X. Li, "Hierarchical model predictive control for multi-robot navigation," *IJCAI*, 2016.

[5] Y. Tian and N. Sarkar, "Formation control of mobile robots subject to wheel slip," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2013, pp. 4553–4558.

[6] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos, *Self-organising software: From natural to artificial adaptation*. Springer Science & Business Media, 2011.

[7] N. Brax, "Self-adaptive multi-agent systems for aided decision-making: an application to maritime surveillance," Ph.D. dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2013.

[8] J. Boes, F. Migeon, and F. Gatto, "Self-organizing agents for an adaptive control of heat engines." in *ICINCO (1)*, 2013, pp. 243–250.

[9] J. Nigon, M.-P. Gleizes, and F. Migeon, "Self-adaptive model generation for ambient systems," *Procedia Computer Science*, vol. 83, pp. 675–679, 2016.

[10] D. Delahaye, C. Peyronne, M. Mongeau, and S. Puechmorel, "Aircraft conflict resolution by genetic algorithm and b-spline approximation," in *EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS*, 2010, pp. pp–71.

[11] N. Durand and J.-M. Alliot, "Ant colony optimization for air traffic conflict resolution," in *ATM Seminar 2009, 8th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2009.

[12] M. A. Christodoulou and C. Kontogeorgou, "Collision avoidance in commercial aircraft free flight via neural networks and non-linear programming," *International journal of neural systems*, vol. 18, no. 05, pp. 371–387, 2008.

[13] G. Roussos and K. J. Kyriakopoulos, "Completely decentralised navigation functions for agents with finite sensing regions with application in aircraft conflict resolution," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 7470–7475.

[14] L. Guys, S. Puechmorel, and L. Lapasset, "Automatic conflict solving using biharmonic navigation functions," *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 1378–1387, 2012.

[15] J. Maas, E. Sunil, J. Ellerbroek, and J. Hoekstra, "The effect of swarming on a voltage potential-based conflict resolution algorithm," in *submitted to the 7th International Conference on Research in Air Transportation*, 2016.

[16] T. Lehouillier, M. I. Nasri, F. Soumis, G. Desaulniers, and J. Omer, "Solving the air conflict resolution problem under uncertainty using an iterative biobjective mixed integer programming approach," *Transportation Science*, 2017.

[17] C. Allignol, N. Barnier, N. D., and J.-M. Alliot, "A new framework for solving en-routes conflicts," in *ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar*, Chicago, United States, Jun. 2013, pp. pp 1–9. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-00828736

[18] P. Machado and K. Bousson, "Automatic collision avoidance system based on geometric approach applied to multiple aircraft," in *6th International Conference on Research in Air Transportation (ICRAT 2014)*, 2014.

[19] C. E. Lin and C.-J. Lee, "Conflict detection and resolution model for low altitude flights," in *Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on*. IEEE, 2015, pp. 406–411.

[20] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics research*, pp. 3–19, 2009.

[21] C. Allignol, N. Barnier, N. Durand, and E. Blond, "Detect & avoid, uav integration in the lower airspace traffic," in *ICRAT 2016, 7th International Conference on Research in Air Transportation*, 2016.

[22] R. Breil, D. Delahaye, L. Lapasset, and É. Féron, "Multi-agent systems for air traffic conflicts resolution by local speed regulation," in *7th International Conference on Research in Air Transportation (ICRAT 2016)*, 2016.

[23] D. Capera, J.-P. Georgé, M.-P. Gleizes, and P. Glize, "The amas theory for complex problem solving based on self-organizing cooperative agents," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. IEEE, 2003, pp. 383–388.

[24] N. Bonjean, W. Mefteh, M. P. Gleizes, C. Maurel, and F. Migeon, "Adelfe 2.0," in *Handbook on agent-oriented design processes*. Springer, 2014, pp. 19–63.

[25] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on intelligent transportation systems*, vol. 1, no. 4, pp. 179–189, 2000.

[26] A. Rantrua, E. Maesen, S. Chabrier, and M.-P. Gleizes, "Learning aircraft behavior from real air traffic," *The Journal of Air Traffic Control*, vol. 57, no. 4, pp. 10–14, 2015.

[27] P. Averty, B. Johansson, J. Wise, and C. Capsie, "Could erasmus speed adjustments be identifiable by air traffic controllers," in *7th USA/Europe air traffic management research and development seminar (ATM2007)*, vol. 22, 2007.

[28] F. A. Authority, "Aeronautical information manual: Official guide to basic flight information and atc procedures," 2006.

[29] N. Durand and N. Barnier, "Does atm need centralized coordination? autonomous conflict resolution analysis in a constrained speed environment," *Air Traffic Control Quarterly*, vol. 23, no. 4, pp. 325–346, 2015.