Neural Networks 22 (2009) 812–821

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2009 Special Issue

Extending the Evolutionary Robotics approach to flying machines: An application to MAV teams

Fabio Ruini *, Angelo Cangelosi [1]

Adaptive Behaviour and Cognition Research Group, Centre for Robotics and Neural Systems, School of Computing & Mathematics, University of Plymouth, Drake Circus, Plymouth PL4 8AA, UK

ARTICLE INFO

Article history:
Received 6 May 2009
Received in revised form 3 June 2009
Accepted 25 June 2009

Keywords:
MAV
Neural network
Multi-Agent System
Evolutionary Robotics
Distributed control

ABSTRACT

The work presented in this article focuses on the use of embodied neural networks – developed through Evolutionary Robotics and Multi-Agent Systems methodologies – as autonomous distributed controllers for Micro-unmanned Aerial Vehicle (MAV) teams. The main aim of the research is to extend the range of domains that could be successfully tackled by the Evolutionary Robotics approach. The flying robots realm is an area that has not been yet thoroughly investigated by this discipline. This is due to the lack of an affordable and reliable robotic platform to use for carrying out experiments, and to the difficulty and the high computational load involved in experiments based upon a realistic software simulator for aircraft. We believe that the most recent improvements to the state of the art now permit the investigation of this domain. For demonstrating this point, two different evolutionary computer simulation models are presented in this article. The first model, which uses a simplified 2D test environment, has resulted in controllers evolved with the following capabilities: (1) navigation through unknown environments, (2) obstacle-avoidance, (3) tracking of a movable target, and (4) execution of cooperative and coordinated behaviors based on implicit communication strategies. In order to improve the robustness of these results and their potential use in real MAV teams, a more sophisticated 3D model is presented herein. The results obtained so far using the two models demonstrate the feasibility of the chosen approach for further research on the design of autonomous controllers for MAVs.

1. Introduction

During the last decade several studies have been carried out on both wheeled and underwater autonomous vehicles driven by embodied neural network controllers (e.g. Baldassarre, Parisi, & Nolfi, 2006; Kodogiannis, 2006). But, as yet, the application of the same principles to flying robots has not been thoroughly investigated. With the only notable exceptions of the systems developed by Buskey, Roberts, Corke, Ridley, and Wyeth (2003), De Nardi, Holland, Woods, and Clark (2006) and Hauert, Zufferey, and Floreano (2009a) it seems that the current approaches to the development of autonomous controllers for aircraft mainly rely on techniques other than neural networks. Examples of the methodologies employed are behavior-based robotics (Dong & Sun, 2004), genetic programming (Barlow, Oh, & Grant, 2005; Richards, Whitely, & Beveridge, 2005), evolution-based path planning (Rathbun, Kragelund, Pongpunwattana, & Capozzi, 2002), modeling field theory (Deming, Perlovsky, & Brockett, 2005; Perlovsky, 2001), and graph search methods (Qu, Pan, & Yan, 2005).

In this study we use a combination of Multi-Agent System (MAS) (Wooldridge, 2009) and Evolutionary Robotics (ER) methodologies (Floreano & Mattiussi, 2009; Nolfi & Floreano, 2000) to develop controllers for Micro-unmanned Aerial Vehicle (MAV) teams for autonomous navigation. Test scenarios include obstacle-avoidance and target reaching experiments in unknown environments. Distributed control, intended as the process of coordinating the movements of a number of agents in order to make them perform a collective task without using a central controller, is generally considered a interesting problem from both technological and scientific perspectives (Nitschke, 2005). Good examples of the complexity involved in designing effective cooperative strategies for teams composed of many unmanned vehicles can be seen in the works made by Hussain, Montana, and Vidaver (2004), and Gaudiano, Bonabeau, and Shargel (2005).

In order to reduce this complexity level, many studies on the behavior of groups of Unmanned Aerial Vehicles (UAVs) have concentrated on flocking and swarming behavior (e.g., Bamberger, Watson, Scheidt, & Moore, 2006; Corner & Lamont, 2004). The simulations we describe here do not share all the principles of swarm systems. For example, in the MAV model proposed here the

* Corresponding author. Tel.: +44 (0) 1752 586288.
E-mail addresses: fabio.ruini@plymouth.ac.uk (F. Ruini), a.cangelosi@plymouth.ac.uk (A. Cangelosi).
[1] Tel.: +44 (0) 1752 586217.

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.06.032

individual agents cannot rely on the fundamental pre-defined rules of swarm and flocking behavior, such as separation, alignment, and cohesion (Kennedy & Eberhart, 2001).

Richards et al. (2005), reviewing the numerous approaches to the development of control systems for teams of unmanned vehicles, propose a useful classification of different methodologies. Even if their categorization is quite detailed, it is probably possible to reduce the family of methodologies identified to just two: (i) reactive strategies and (ii) deliberative strategies. The approach we have chosen for studying the emergence of cooperation in MAVs is based on reactive strategies. In other words, no pre-planned strategies for the teams are developed, since all the aircraft simply react to the sensorial input they can gather, directly or indirectly, from the environment. The cooperation emerges spontaneously, simply modifying the rules driving the individual behaviors. This method has several advantages with respect to those belonging to the other main category, which is deliberative approach. Deliberative approach strategies focus on developing a specific flight path for each aircraft belonging to a team (see for example Ablavsky, Stouch, & Snorasson, 2003) to follow. But generating fixed routes in advance implies that a very good knowledge of the reference environment is available to the central controller, whether it is a human or a computer system. UAVs relying on such a kind of deliberative controller could therefore be considered autonomous, in the sense that they will be able to autonomously follow a pre-planned flight path. But they would not have the ability of taking autonomous decisions, therefore resulting in a lack of intelligence (autonomy). This does not represent an issue for domains such as civilian aviation, where all the needed information is immediately available. The lack of flexibility related to the deliberative approach instead becomes problematic if we try to apply the same principles to dynamic or unknown scenarios. Attempts have been made to overcome these drawbacks by incorporating in deliberative approaches some elements of adaptive replanning. The implementation of this kind of improvement requires equipping the aircraft with a set of sensors that enables them to fetch previously unknown, non-accessible and/or non-existent information from the environment. The idea behind adaptive replanning is that a centralized controller generates a specific flight path for each UAV to follow based on the currently available information. UAVs strictly follow those paths until they detect some new elements through their sensors (e.g. an unknown enemy or an unexpected obstacle). When this happens, the sensor information gathered is sent back to the controller, which may then decide to generate new flight paths for the entire team (or just part of it) and transmit them to the UAVs. A good example of adaptive replanning can be seen by looking at the "UAV manager" concept elaborated by Rathinam, Zennaro, Mak, and Sengupta (2004). Despite the fact that the adaptive replanning approach looks promising, many issues remain to be addressed in deliberative strategies. For example, to decide when a replanning is required, and the amount of time needed to calculate and broadcast the new flight paths to the various UAVs are two non-trivial elements to consider. Scherer, Singh, Chamberlain, and Elgersma (2008) have recently identified a possible solution using two separated but interacting controllers that respectively act on a global and on a local level ("plan globally and react locally"). Even in this case, a good level of knowledge about the environment is still required. Generally speaking, we might argue that it is the need for a central controller that is highly problematic. As highlighted for example by Wu, Schultz, and Agah (1999), distributed control is generally preferable since its non-critical reliance on any specific element can in turn guarantee increased reliability, safety and speed of response to the entire system. In addition to this we believe that a distributed control system also has a better potential to produce adaptive and flexible solutions for the tasks we are interested in studying.

The main difference between the methodology we are following and a "standard" reactive strategy approach (as the one described in Richards et al., 2005) mainly consists in the employment of a neural network controller instead of a properly defined decision tree. In both cases the controllers are subjected to an evolutionary process and therefore the use of computer simulators for the training phase is compulsory. The basic principle we have adopted is to some extent similar to the ones proposed by Buskey et al. (2003) and De Nardi et al. (2006) for the autonomous control of unmanned helicopters. The controllers we use are in fact embodied neural networks whose outputs affect the aircraft's orientation and its direction of motion. However our approach introduces at least three elements of novelty. The first is that we aim to study the (simplified) dynamics of airplane-like UAVs rather than helicopters. Even employing streamlined simulation models, as the two described in this article, helicopters are much more flexible in adjusting their movements during flight when compared to airplane-like aircraft. If, for example, an unexpected obstacle is encountered, a helicopter could easily hover overhead, perform a 180 degrees yaw and then look for a different path to follow. When it comes to an airplane-like aircraft, this kind of behavior is not possible, so the on-line adjustments to the current route need to be extremely accurate.

The only work, to our knowledge, where neural networks are applied to the control of non-helicopters or blimp aerial vehicles is the one by Hauert et al. (2009a). Furthermore, another major novelty consists in our decision to implement a basic obstacle-avoidance mechanism, which represents an additional challenge to be addressed by the neural controller. Traditionally, obstacle-avoidance behavior has not been taken into account in studies regarding UAV path planning. As pointed out by Rathbun et al. (2002), this is mainly due to the fact that UAVs have usually been restricted to operating in areas that do not contain any other vehicles outside the control of the authority in charge of it. Rathbun's work, where an evolution-based path-planner is able to deal with movable and non-accurately estimated obstacles, constitutes one of the few meaningful exceptions to this trend. Finally, the controller we use is made of a single feed-forward neural network and not of different modules joined together, each of these dedicated to managing different sub-tasks as in Buskey et al. (2003) and De Nardi et al. (2006). The entire controller therefore acts as a single entity. The task of identifying a favorable decomposition of the controller into different dedicated modules is left to the evolutionary process.

2. The 2D simulation model

In this section we introduce the first of the two models that we have developed for our research. As previously mentioned, our approach requires the employment of a computer simulator for the evolution of the MAVs' autonomous controllers. The structure of the simulator is quite simple. A team is composed by four MAVs, each endowed with its own neural network controller, identical to the ones of its teammates. The task the MAVs have to perform is a classical "search and hit" situation. At the beginning of a test, an "enemy" target is deployed somewhere inside the environment. The simulated scenario is a rectangular area, with size 710 × 760 pixels (px), consisting of a 2D representation of the Canary Wharf financial district in London (Fig. 1). MAVs are represented as squares with a side length of 2px.

Starting from the four corners of the area and facing the center of the environment, the MAVs have to fly towards the target attempting to eliminate it. In order to neutralize the target, one of the MAVs needs to perform a self-detonation when it is close enough to it (2.48 px or less). A test ends when the target has been destroyed or no MAVs are still living. An MAV will die if it performs

Fig. 1. A screenshot of the 2D simulator. On the left it is possible to see the environment used in this model. The obstacles, corresponding to the tallest buildings present in the Canary Wharf area, have been highlighted.

a detonation, if it exits from the environment's boundaries, if it collides against a teammate, if it runs out of energy, or if it crashes against a building.

Automatic target acquisition (ATR) is not provided to the MAVs. In this way they do not need to execute such an intensive computational task (even if the job could be effectively tackled cooperatively, as demonstrated for example by Dasgupta, 2008). Our assumption is based on the presence of a satellite system that constantly monitors the target and broadcasts real-time information about its position to all the team members. In this way the MAVs, equipped with a GPS receiver, can easily calculate their distance from the target matching the two data sources gathered. A simple compass can also allow the MAVs to determine the relative direction in which the target is located. In our simulator each MAV is in fact given information about the distance between itself and the target, as well as the angle that separates the two agents based on the current MAV's heading. This information is received by the neural network (Fig. 2) controlling the aircraft's behavior by means of four input neurons: one encoding the distance (using values discretized according to the maximum distance possible inside the reference environment), the other three the angle (using a Gray Code representation of eight possible sub-spaces). The MAVs are also endowed with three ultra-sonic sensors (respectively orientated at $-20°$, $0°$, and $+20°$ according to the aircraft's heading), capable of detecting the presence of an obstacle. Categories of obstacles that MAVs can spot are the target, the teammates, the buildings, and the environment boundaries. This information is encoded using three continuous neurons, each of them activated with a value representing the distance from the current sensor and the closest obstacle perceived by it, if any are within a certain range. The seven input neurons are fully connected to the neural network's hidden layer, made of fifteen continuous neurons, activated through a tan-sigmoid function (slope 1.0), which output values are within the range $[-1.0; +1.0]$. The neural network's output layer consists of just two neurons, receiving incoming connections from all the neurons belonging to the hidden layer. One output unit controls the MAV's yaw ($+/-$ 20° in the time unit; this neuron has the same activation function as the hidden layer neurons, but the output value is translated into the range $[-20.0; +20.0]$); the other one is a Boolean neuron (activated through a step function with a 0 threshold) that, when it turns to 1, causes the MAV to carry out the detonation. It is worth highlighting how all the neurons employed in this network just use summation as the aggregation function. It means that the activation function for each neuron (referred to below with the letter g) can be formalized according to (1), where $w_n + 1$ is a parameter needed to take into account the biases, n is the number of neurons connected to the given one, x_i is the activation value of the i-th neuron, and w_i is the weight of the connection between the i-th neuron and the given one.

$$f(x) = g\left(w_n + 1\sum_{i=0}^{n} w_i x_i\right). \tag{1}$$

The fact that we are simulating an airplane-like motion implies the constraint, for the MAVs, of never being stationary. The speed is assumed as constant: during each time-step all of the simulated aircraft move 2 pixels along their heading direction before eventually performing a yaw rotation.

The evolution towards a controller able to perform the desired task is made possible through the use of a genetic algorithm (GA) (Mitchell, 1998; Nolfi & Floreano, 2000). An initial population of 100 teams is created with randomly assigned connection weights and biases ranging from -1.0 and $+1.0$. The genome (consisting of a vector of real values, directly encoding connection weights and biases values) is generated at a team-level. This means that all the MAV members of a certain team share the same genotype, i.e. they are driven by the same identical controller as their team-mates. Each MAV team is tested four times with the target deployed in randomly chosen positions within specific areas. Twice the target will be inside an "enclosed area" at the center of the environment, surrounded by buildings and with narrow entrances, twice it will be put outside this area. The MAVs start each test with an initial storage amounting to 5,000 energy units each, and they spend 2.14 energy units per time-step. At the end of each generation the 20 individuals that have performed the best scores according to the

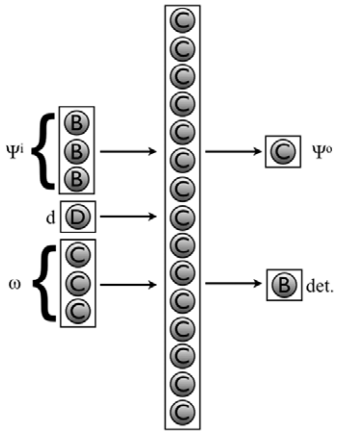

Fig. 2. Graphical representation of the basic neural network controller used for the 2D model. The input layer, on the left, contains three neurons encoding the horizontal angle (Ψ^i) and the distance (d) between the MAV and the target, as well as the ultra-sonic perception (ω) of obstacles. Between the input and the output layer there is a hidden layer made of 15 continuous neurons, activated by a tan-sigmoid function. The output layer, on the right, contains the neuron that generates the yaw (Ψ^o), plus the one dedicated to the detonation of the aircraft (det.) All the modules are fully forward connected. [B: Boolean, D: discrete, C: continuous].

fitness formula are selected for reproduction. The fitness formula used is the following:

$$f = -\alpha + \left(\frac{\beta}{50}\right) + (\sigma * 50) + (\varepsilon * 10), \tag{2}$$

where: α is the average distance (measured in pixels) between the target and the team member detonated closest to it, calculated based on the four tests performed; β is the average amount of energy retained by the MAV detonated closest to the target, again calculated based on the four tests; σ is the number of tests concluded by the given team with the elimination of the target; and ε is the total number of MAVs remaining alive after the four tests have finished (maximum 12).

Each of the selected teams generates 5 copies of its genome, on which the mutation operator is then applied. Each gene of the copied genome is modified, with probability 0.25, by a random amount between -0.5 and $+0.5$. The only exception is for the best individual of the current generation, which generates a copy of its genome without any modifications (elitism). The resulting 100 individuals will constitute the new population at the next generation. The evolutionary process lasts for 2,500 generations and it is repeated 10 times, with the results coming from all the different runs averaged, in order to obtain more robust data about the generated trends.

The results show how the elaborated set up can lead quite easily to the evolution of the desired behavior. At the end of the evolution, on average, we have the 93.46% of tests successfully concluded in empty environments and 87.18% when obstacles are present. It is interesting to consider how the fitness formula we have decided to use does not require taking into account any information about the environment, such as waypoints disseminated in specific places (as did for example by De Nardi et al., 2006; and Scherer et al., 2008). Navigation and obstacle-avoidance abilities emerge run-time as sub-tasks necessary for the completion of the main task, which is to neutralize the target.

2.1. Experiments using a movable target

In this experimental scenario, the target is able to detect an MAV approaching it. This new property of the target has been introduced to increase the complexity of the task and test the robustness of the model. The simulation starts as usual, with the target deployed in a random position within the environment and fixed on it. During each time step, if an MAV happens to be closer than 17 px to the target, the latter switches to a particular "MAV detected mode" with probability 0.5. In the event of detection, the target will then move away from the closest aircraft during each time step in order to maximize the distance from it. The decision on the best place to move is taken after the evaluation of 8 alternative locations. These positions are respectively located at north, north/east, east, south/east, south, south/west, west, and north/west, calculated on the basis of the current target's orientation. The distance of these locations from the target (equal for all of them) depends on its speed. The target will keep escaping from the aircraft until all of the detected MAVs die or the distance between the target and the closest aircraft rises above the 48 px threshold.

Five simulations have been carried out where we vary the escaping speed of the target. These different speeds in the various simulations respectively correspond to different fractions of the MAV's speed (M_s) : $M_s/2$ (Simulation A1), $M_s/3$ (Simulation A2), $M_s/4$ (Simulation A3), $M_s/5$ (Simulation A4), and $M_s/6$ (Simulation A5). The results obtained are summarized in Table 1.

Observing the outcome of these simulations (more details can be found in Ruini, Cangelosi, & Zetule, 2009) we can easily identify a kind of threshold. Simulations A3, A4 and A5 seem to perform equally well according to the various parameters measured. Simulation A2 produces a significantly worse performance for the average fitness, but could be considered as performing reasonably well if we take into account both the maximum fitness and the average percentage of tests concluded successfully. In simulation A1 the success rate of the MAVs drops instead.

Comparing these results with the ones obtained using a static target, we can notice a general performance decrement. As clearly shown in Fig. 3, the difference is mainly concentrated in the average values, while the maximum ones (i.e., the best individuals/controllers within a certain generation) tend to reach a similar level of performance. The main conclusion drawn from this experiment is that the algorithm setup can evolve MAV controllers able to navigate through unknown environments and autonomously reach and destroy a target, not only when the latter is fixed on a certain position, but also if it is able to move away from them. The only constraint is that, in order to keep a good success rate, the target should not be able to move faster than one third of the MAVs' speed. This is quite a reasonable assumption if we suppose that the target is not a vehicle, but a person instead. A typical MAV platform could easily reach a speed of 50 km/h. One third of this velocity roughly corresponds to 16–17 km/h. Considering that the speed of an average person moving within a crowded environment could be approximated to 4–7 km/h while walking, and 12–15 km/h while running (furthermore, this speed would be just maintainable for a short period of time), we might argue that the evolved controllers are able to accomplish their task with a good degree of confidence even against a movable target.

2.2. Experiments requiring a cooperative approach

The setup labeled as experiment B adds the constraint of requiring two MAVs to detonate against the target at the same time (i.e., within a limited maximum number of time-steps apart from each other, since the simulation works in discrete time) in order to neutralize it. The target begins each training epoch with the assigned status of "intact". When it happens that one of the MAVs detonates close enough to it (i.e., the same situation as in the previous setups would have provoked the elimination of the target), the target's status switches to "damaged". If a second MAV manages to detonate close enough to the target while this is still in the damaged mode, the latter will be eliminated. Otherwise, after 10 time-steps of damaged state, the target will restore its original

Table 1
Summary of the results for Simulations A.

Sim.	Av. fitness	Max fitness	Av. success (%)	Av. dist from the target (px)	Min. dist from the target (px)
A1	138.93	395.18	54.48	92.94	1.34
A2	198.08	409.67	78.28	107	1.09
A3	250.68	411.74	81.89	72.47	0.95
A4	258.72	409.9	83.3	70.03	0.98
A5	242.42	413.05	84.06	95.53	0.81

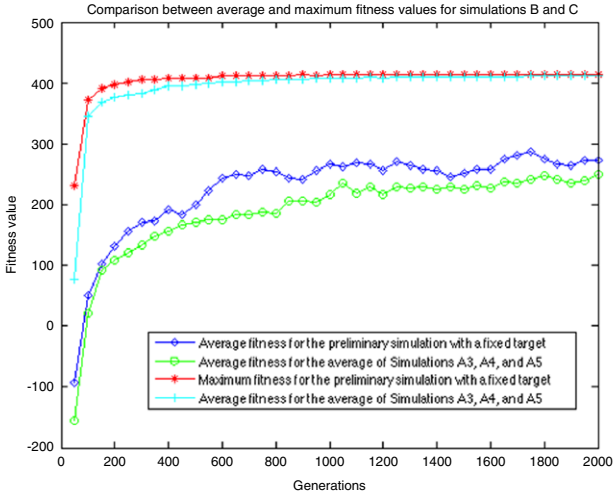

Fig. 3. Average and maximum fitness for simulations A3, A4 and A5 compared to the preliminary results obtained using a fixed target.

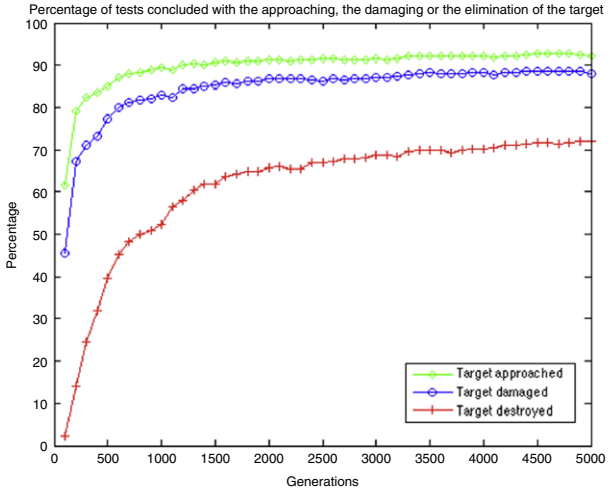

Fig. 4. Simulation B — percentages of tests respectively concluded with the approaching, the damaging and the neutralization of the target when it is not capable of moving and it has to be attacked cooperatively.

"intact" condition and the simulation will goes on as usual till the neutralization of the target or the failure of the entire MAV team.

In order to make the MAVs able to accomplish this task, we have provided them with the capability of gathering new pieces of information from the environment. Each member of the team is now able to detect both the status of the target (intact rather than damaged) and the presence of a teammate within a 30 px distance. We assume that the information about the target's status is provided to the active MAVs by the satellite system (see Section 2), following the disappearance of an aircraft when at a distance compatible with the damaging of the target. As in any real life scenario where a task has to be performed cooperatively, the agents involved in it need to be provided with the capability to communicate, explicitly (i.e., intentionally) or implicitly (i.e., non-intentionally), with each other. In this experiment we introduce a simple form of implicit communication, merely consisting in the observation of the other teammates (Pagello, D'Angelo, Montesello, Garelli, & Ferrari, 1999). In more detail, we define experiencing an implicit communication exchange as the ability to detect the presence of a teammate within a MAV's own sensorial space. This information is given as input to the neural controller through two additional Boolean neurons. These two neurons implement a kind of logical OR. Apart from being in the proximity of the target, in order to decide the proper moment at which to detonate, an MAV also needs to know that there is a teammate close to it (thus the second hit can presumably be delivered soon), or that the target has recently been struck (i.e., it is currently on the damaged status), or that both conditions (closeness of a teammate and target's damaged status) are true. Three neurons have been added to the hidden layer. Apart from these modifications the rest of the neural network maintains the same characteristics as before.

The fitness formula has been also modified in order to let the new desired behavior evolve. We have now introduced the concepts of "target approached" and "target damaged". At the end of a test, we define the target as approached if at least one MAV has detonated within a 63 px distance from it. The target is instead considered damaged if at least one MAV has managed to hit it. These modifications tend to recreate what we could call an incremental evolutionary process (though pursued in a different way than that done for example by Barlow et al., 2005). The MAVs initially learn how to perform the simplest sub-tasks (avoiding obstacles and approaching the target) and then progressively move towards the more complicated sub-tasks (damaging and neutralizing the target respectively), which in turns make the accomplishment of the overall task possible.

Putting this all together, the new fitness formula is:

$$f = (\gamma * \frac{\chi}{4}) + \left(\eta * \frac{\chi}{2}\right) + (\lambda * \chi) + (\varepsilon * 10) + \left(\frac{\beta}{50}\right), \qquad (3)$$

where: γ is the number of tests concluded with at least one MAV approaching the target in the sense we have defined above; η is the number of tests concluded with at least one MAV damaging the target; λ is the number of tests concluded successfully with $\chi = 50$ (χ is just a parameter arbitrary chosen in order to assign different specific weights to γ, η and λ). Parameters ε and β have similar meanings to those in (2), as they respectively represent the total number of MAVs surviving at the end of the all tests and the average amount of energy retained by the MAVs that had eventually neutralized the target. Consider that now every team is tested 12 times and the evolutionary process lasts for 5,000 generations.

Figs. 4 and 5 show the results obtained with this experimental setup, respectively with a fixed and a movable target. The simulations carried out using a fixed target have produced an overall good performance. On average, for the individuals belonging to the last generation, more than 70% of tests are successful, while 90% finish with the target hit at least once.

The strategy that the MAVs evolve is straightforward, but nonetheless very effective. They independently look for the way to the target as in the previous experimental setup, without

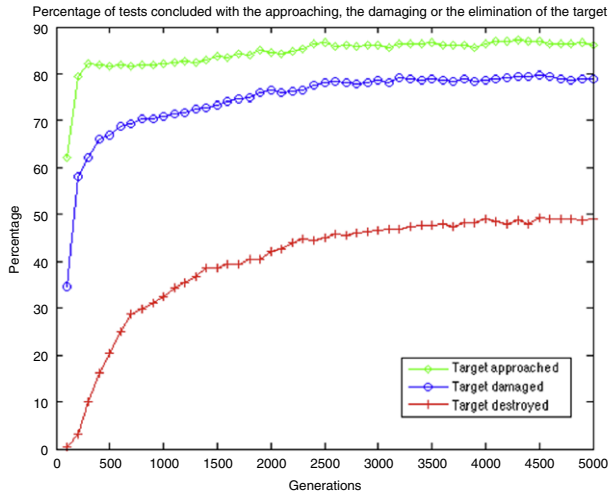

Fig. 5. Simulation B — percentages of tests respectively concluded with the approaching, the damaging and the neutralization of the target when it is able to move and it has to be attacked cooperatively.

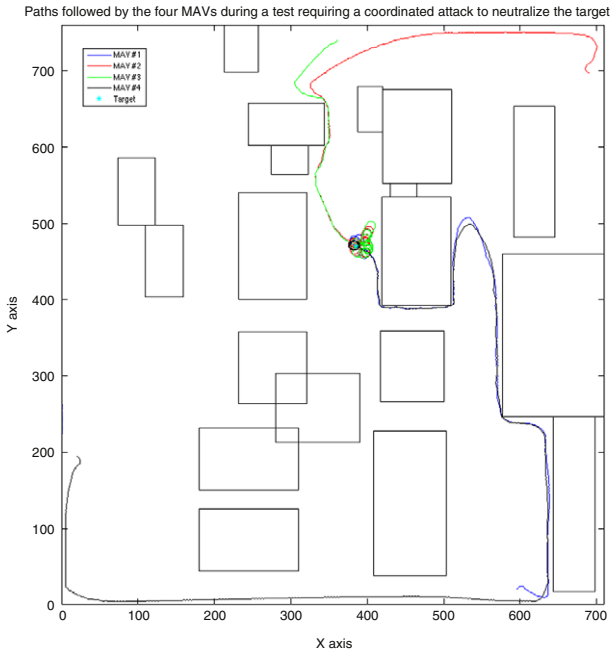

Fig. 6. Flight paths followed by the members of a team belonging to the last generation in order to reach the target and attack it cooperatively.

any interactions (if not purely generated by chance) with the teammates. Once the first MAV gets close to the target, instead of detonating it continuously flies in a circle around its position. Only when a teammate also arrives in the proximity of the target (thus being detected by the aircraft already there), does the first MAV carry out the detonation damaging the target. The second MAV, detecting the target as damaged, attacks it without waiting for any other team members to arrive, successfully concluding the test. An example of the evolved behavior can be observed in Fig. 6.

The performance of the teams dramatically decreases when the target is moving. In this experimental setup, only 50% of tests end with the neutralization of the target, even if the percentages of tests concluded both with the approaching and with the damaging of the target are comparable to those obtained in case of a non-movable target. Of course we have to consider that what we are illustrating are average results referred to an entire population. It means that, within this population, the likelihood of having MAV teams that are particularly good at performing the desired task (i.e. with an accuracy level close to 100%), is extremely high. Nevertheless, a target being able to move constitutes a major issue for the MAVs, since they find it more difficult to implement the strategy described above. The first aircraft approaching the target can have in fact a hard time flying around the target when it moves, for example, close to a building (when a test starts, the target is always deployed at a certain distance from both the buildings and the environment boundaries). The continuous target's movements could also provoke the attacker to incur some mistakes, for example with the first MAV misjudging the right moment in which to detonate and resulting in the second aircraft arriving too late at the target. Future investigations will focus on the introduction of forms of explicit communication between the MAVs that could positively affect the likelihood of successfully completing this kind of task.

3. The 3D simulation model

The extension of the previously described model to a 3D one requires an expansion of the number of degrees of freedom (DOF) involved in the evolutionary and neural control process. The new 3D model introduces two new DOF. In the 2D simulator described before, MAVs were only able to rotate their body clockwise or counter-clockwise, while constrained to move along their heading direction at every time step. They were substantially relying on a single DOF. The aircraft we are now modeling in the 3D environment are instead able to rotate along three different axes. If we consider an orthogonal axis system fixed on the aircraft and constrained to move with it (what is sometimes defined as "body axis system"), the rotations that are possible to the MAVs can be respectively defined as "yaw", "pitch", and "roll". In detail: yaw corresponds the rotation of the aircraft around its vertical axis; pitch is the rotation around the side-to-side axis; roll refers to the rotation around the front-to-back axis (for a more accurate description of the systems of axes and notations used in flight dynamics, see for example Cook, 2007).

The use of a new 3D simulator involves an additional issue in the domain of autonomous controllers for aircraft. This regards terrain avoidance (see for example Netter & Franceschini, 2002), and it represents a key capability that should be provided to MAVs along with obstacle avoidance. For the sake of simplicity, the scenario used for this simulation is assumed as a flat ground plane located at sea level. Though an explicit control strategy for terrain avoidance is not planned in the model; as we will see later the MAVs will acquire this ability as a side product of the evolutionary process. MAVs have to be able to avoid crashing into the ground, but they also must not fly at an excessive height. The simulator, in fact, will consider "lost", without any distinctions, all the aircraft exiting from the bottom or top boundaries of the environment.

The environment simulated in the 3D model measures 1,000 (X) $\times$ 750 (Y) $\times$ 600 (Z) graphical units (GU^2). The target is represented by a sphere with 15 GU radius. All the MAVs have an identical shape and size with a length of 5 GU and a wingspan of 3.5 GU.

The task in this new simulator is the same as before. A certain number of MAVs – all members of the same team, but moving from different positions – have to navigate through the environment, reach a certain target deployed inside a central area (without exiting the environment boundaries) and then perform a detonation in order to neutralize it. The test is considered successful when the detonation happens within a 15 GU distance from the target. If all MAVs "die" without having neutralized the target, the test is considered failed. In addition to losing an MAV because of self-detonation or exiting from the environment boundaries, an aircraft

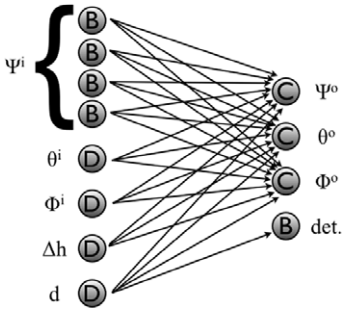

Fig. 7. Graphical representation of the neural network controller used for the 3D model. The input layer, on the left, contains four neurons encoding the horizontal angle separating the MAV and the target (Ψ^i), the MAV's pitch and roll angles (θ^i and Φ^i respectively), the delta height (Δh) and the distance (d) between the MAV and the target. The output layer, on the right, contains the neurons generating yaw, pitch and roll (Ψ^o, θ^o, and Φ^o), plus the one dedicated to the detonation of the aircraft (det.). [B: Boolean, D: discrete, C: continuous].

can also be lost during a test if it runs out of energy or if it collides against a teammate.

The controller we have designed for this model (Fig. 7) is based on a feed-forward neural network, constituted by one input and one output layer. The input layer receives five chunks of information: (1) horizontal angle between the MAV and the target, (2) distance to the target, (3) delta height compared to the target, (4) current pitch angle, and (5) current roll angle. The output layer provides the effectors of the MAV's behavior: (1) yaw, (2) pitch, (3) roll, and (4) detonation. The two layers are not fully connected. While the neural modules dedicated to modifying the MAV flight trajectory (i.e., yaw, pitch and roll) receive their incoming connections from all the input neurons, the module dedicated to the detonation only receives information about the distance between the MAV and the target. Each controller is characterized by 25 connection weights and 4 biases (each of them applied to one of the output layer's units). All the neurons belonging to the input layer do not have any bias and use an identity transfer function, which does not modify the values set in input. The information fed to the network is processed and then passed to the output layer. The output neurons that manage yaw/pitch/roll produce a continuous output value, based on the weighted sum of all the contributions received in input. They are activated according to a tan-sigmoid function (slope 1.0), which outputs in the range $[-1.0; +1.0]$. The output unit dedicated to the detonation is instead a Boolean one, activated by a step function with a 0 threshold. When it turns to 1 the MAV detonates, while nothing happens when the value of this neuron is 0.

For measuring pitch and roll angles (also referred to as "attitude") we have chosen a right handed axis system. This means that positive pitch is nose up and positive roll is right wing down. The correlation between the output values generated by the controller and the effects made on the aircraft is 1-to-1. During each time step an MAV can therefore perform a pitch between -1.0 (nose down) and $+1.0$ (nose up) degrees, and a roll ranging from -1.0 (left wing down) and $+1.0$ (right wing down) degrees. Positive yaw is considered as a clockwise rotation around the aircraft's vertical axis. Again the amount of this rotation ranges between -1.0 and $+1.0$ at each time step. The 0 output does not generate any change to the current aircraft's orientation (for more details about the input/output encodings used, see Ruini & Cangelosi, in press). Consider that in this model both roll and yaw are available as options to the controller. Real aircraft, and this is particularly true for MAVs, can perform different steering manoeuvres according to how they have been designed. Typically the decisions made during the design phase allow the aircraft either to roll (when ailerons have been installed on the trailing edge of the central wings in case of a fixed-wing airplane) or alternatively to yaw (if the steering is implemented through the rear wing). In the latter scenario, a yaw manoeuvre also generates a correlated roll of the aircraft (see Watson, John, & Crowther, 2003). Providing the controller with the capability of exploiting both yaw and roll, and notwithstanding how this decision makes the evolutionary process harder, we can obtain a greater degree of flexibility from the model.

Each MAV's movement is 3 GU long and cost 1 energy unit. The flying speed is assumed as constant, it cannot be modified during flight and cannot therefore affect the energy consumption.

Also in this case it is an evolutionary process that leads to a suitable configuration of weights and biases for the network. From a GA perspective, an initial population of 100 teams is created. When the simulation begins, each value contained in the genotype (i.e., each connection weight and bias characterizing the neural network) assumes a random value in the range $[-5.0; +5.0]$. All agents within a team are genetically clones, thus sharing the same genotype. Each team is formed by 4 MAVs, and it is tested for 8 epochs. At the beginning of each epoch the energy stored in each MAV amounts to 1,000 fuel units (FU). MAVs can therefore fly for 3,000 GU. When an epoch starts the target is randomly placed in a certain position, 15 GU above ground level. All the MAVs belonging to the same team are deployed in different starting positions close to the four environment corners and tendentially (due to some random noise added to alter their initial position) facing the center of the cuboidal arena.

The fitness formula used for driving the evolutionary process has been kept as simple as possible:

$$f = -\alpha + \beta, \tag{4}$$

where: α is the average distance (measured in GU) between the target and the team member detonated closest to it, calculated based on the various tests; and β is the average amount of energy retained by the MAV detonated closest to the target during each test. If no MAVs detonate during the entire set of test epochs (which is a condition that could frequently happen during the first stages of evolution), the values of α and β are automatically set to 1,386.54 and 0 respectively.

At the end of each generation, the 10 teams that have scored the best performances according to the fitness formula are selected for reproduction. Every parent team generates 9 offspring teams, which inherit the same genome. For each new team, a mutation process is applied. All the connection weights and biases are affected, with probability 0.2, by the addition of a random value included within the range $[-1.0; +1.0]$. Elitism is also applied, so the first offspring of the best team within a given generation is not affected by any kind of modification of its genome and is kept unchanged in the next population.

3.1. Preliminary results

The evolutionary process described in the previous section is iterated for 350 generations. Five different replications (i.e., the same simulation with different initial random genotypes at generation 0) are carried out. The results reported here (to which we refer as experiment C) are averaged across the five replications in order to identify a general evolutionary trend. The charts described below provide a summary of the results generated by the evolutionary process.

[2] We are now following the conventions of the 3D engine used, namely Irrlicht (http://irrlicht.sourceforge.net), which uses "graphical units" instead than pixels.

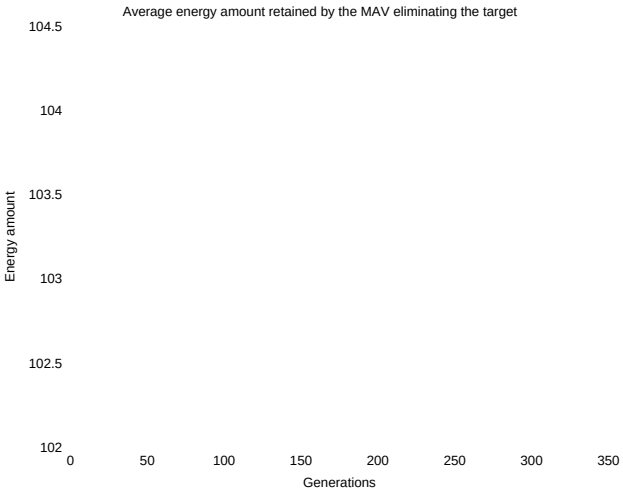

Fig. 8. Simulation C — average and maximum fitness.

Fig. 10. Simulation C — average amount of energy retained by the MAV eliminating the target.

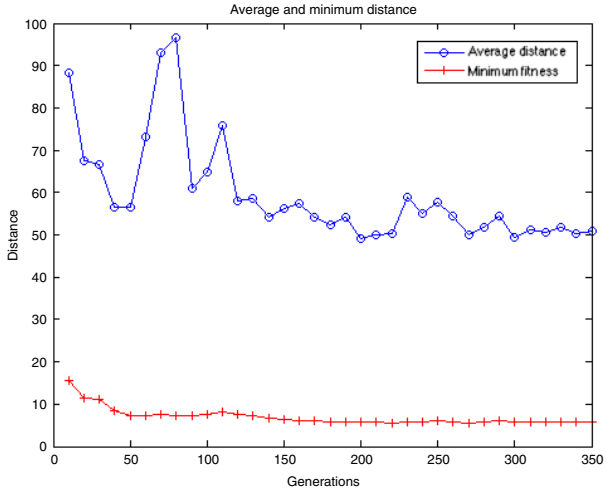

Fig. 9. Simulation C — average and minimum distance between the target and the detonation point of the MAVs exploded closest to it.

Fig. 11. Simulation C — percentage of tests concluded successfully.

In Fig. 8 we can see how average and maximum fitness increase generation by generation, until reaching what looks like a steady state after 300 generations. As expected, both lines start from low fitness levels (for the first 50 generations the average fitness is negative, indicating that the "average team" is not yet able to perform the desired task) and then quickly increase.

As mentioned above, the fitness formula used for evaluating the performance of an MAV team has two main components. The first is the average distance between the target and the MAV detonated closest to it, during the 8 tests carried out for each team. Fig. 9 shows how the minimum average distance (i.e., the one scored by the best swarm averaging the 8 tests carried out) reaches a steady state after a few generations. The line representing the average for the entire population is instead more jagged and decreases more slowly.

The second component of the fitness formula is the amount of energy retained by the MAVs that have neutralized the target, averaged for all the successful tests performed. This parameter has been included in the fitness formula with the purpose of creating a discriminatory effect whenever most of the MAV teams within the same generation would have been able to correctly reach and neutralize the target. The idea is to favor those teams able to perform the task faster than other members of the population. We were therefore expecting a curve characterized by values increasing generation by generation (or at least starting to increase after the average success rate for the entire population has exceeded a certain threshold). The results presented in Fig. 10 show how this expected phenomena in fact does not take place. This is presumably due to the number of generations evolved, which is too limited for this trend to appear.

It is interesting to note that even if the fitness graph suggests that the evolution has reached a steady state, in reality this is not the case. Looking at Fig. 11 it becomes obvious that the evolutionary process is still going on. What is happening is that the entire population is still converging to the optimum point individuated by the evolutionary algorithm. This phenomenon might make results difficult to see in Figs. 9 and 10, but it is quite clear in Fig. 11.

4. Conclusions

The work described in this paper has demonstrated the feasibility of an approach based on a mixture of evolutionary robotics and multi-agent systems methodologies for the development of autonomous controllers for flying robots. Controllers evolved in