



## 2009 Special Issue

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

Fabio Ruini\*, Angelo Cangelosi<sup>1</sup>

*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.

© 2009 Elsevier Ltd. All rights reserved.

## 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 Gaudiانو, 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](mailto:fabio.ruini@plymouth.ac.uk) (F. Ruini), [a.cangelosi@plymouth.ac.uk](mailto:a.cangelosi@plymouth.ac.uk) (A. Cangelosi).

<sup>1</sup> Tel.: +44 (0) 1752 586217.

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  $710 \times 760$  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  $-20^\circ$ ,  $0^\circ$ , and  $+20^\circ$  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  $[-1; 1]$ . 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 ( $\pm 20^\circ$  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  $[-20; 20]$ ); 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.

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

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













