WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain IJCNN

An Evolutionary Robotics 3D model for autonomous MAVs navigation, target tracking and group coordination

Fabio Ruini, *Member, IEEE* and Angelo Cangelosi

Abstract— **The work presented herein describes an application of Evolutionary Robotics controller design methodologies to the domain of Micro-unmanned Aerial Vehicles (MAVs). The aim of this paper is to extend and validate preliminary results obtained through a simplified 2D simulator, to a more realistic 3D model. After a technical introduction of the newly developed simulation model, the results generated by three different experimental setups - all of them focused on autonomous navigation toward a specific target area - are described. The first scenario simply involves a single MAV navigating through a plain environment toward a non-movable target. In the second setup the target is able to move away, at different speeds, when approached by the aircraft. Finally, in the third scenario, teams consisting of more than one MAV are employed; the team members have to coordinate among themselves - exploiting implicit communication strategies - in order to reach the target at the same time. The nature of the tasks studied requires a high level of accuracy by the controllers, something which is not common in most of the ER literature.**

I. Introduction

Evolutionary Robotics (ER) [1][2] is a scientific field, having as principal aim the design of autonomous controllers for robots, that has received great attention during the last few years. Despite the number of researches carried out in accord to this paradigm and the amount of theoretical work performed in order to better understand the processes underlying the evolutionary design, the practical results obtained so far can still be considered disappointing from many points of view. Any person who has had the chance to see in action, for example, a Khepera [3] or an E-Puck [4] robot driven by an evolutionary controller, must have immediately noticed how these robots can display quite sophisticated behaviour. The drawback consists in the fact that they mostly tend to behave in a careless and unsystematic way, moving slowly and frequently changing their direction, bumping into obstacles, getting stuck into never ending movement loops, etc..

In practical applications, the difference between an evolutionary controller and a hand-designed one is, generally speaking, so huge that does not need to be highlighted. The sloppy performances shown by ER controllers are typically attributed to the so-called *reality gap effect* [5], taking place when the evolved controllers are transferred from computer simulated robot models to physical platforms facing the complexity of the real world. Computer simulations are a common instrument used in ER, employed in order to reduce the time required by the evolutionary process as well as to do not subject the real robots to potentially dangerous situations. The reality gap effect arises because of the simplifications introduced during the development of the computer models. As models, they tend in fact to be simplified reproductions of the reality, targeted to reproduce the main characteristics of the real world only. A controller evolved in such a low-complexity world might incur in unexpected situations when put inside a real physical environment, resulting in erroneous or non-optimal behaviours.

Fabio Ruini and Angelo Cangelosi are with the Centre for Robotics and Neural Systems, Adaptive Behaviour and Cognition Research Group, School of Computing and Mathematics, University of Plymouth, UK (phone: +44 (0)1752 586288; email: {fabio.ruini, acangelosi}@plymouth.ac.uk.

At the same time it might be argued that ER researchers have not been yet particularly concerned on obtaining fast and high-precision controllers. This work aims to represent a step forward in the direction of developing more accurate evolutionary controllers than those created so far. To demonstrate that no radical modifications to the classic ER approaches are required is another of our goals. In order to pursue this vision, the domain that we have decided to tackle is one that, by definition, requires a great degree of precision. Our focus is in fact on the design of autonomous controllers for fixed-wing Micro-unmanned Aerial Vehicles (MAVs) [6].

A. Literature review

The literature seems not being abundant in terms of research carried out with neural networks - let alone genetically evolved - employed as controllers for robotics aircraft. For the most part, researchers working on autonomous controllers for aircraft outside the traditional engineering fields rely on techniques other than neural networks, such as behavior-based robotics [7], genetic programming [8] and evolutionary-based path planning [9]. There are nonetheless some significant exceptions to this trend, constituted for example by the CSIRO [10] and the SwarMAV [11] projects, where small helicopters use, to different extents, neural networks as controllers. Helicopters, although presenting issues related to flight stabilisation, can be considered much more easily controllable than fixed-wing aircraft. Moving from these considerations, it does not come to surprise that the latter domain is only marginally touched by the literature in autonomous control, with the main contributions coming from the ongoing research at the EPFL [12] [13].

B. Preliminary work

Before describing our latest work, we briefly review the preliminary investigations that have been carried out in order to test our working hypothesis on a simplified setup. In our previous work [14] [15] we have built a 2D computer simulator aimed to identify the possible limitations of an application of the ER approach to the domain of MAVs autonomous navigation within urban-like environments. Inside the simulated environment we created (taking inspiration from the Canary Wharf quarter in London), the MAVs driven by evolutionary controllers have proven to be capable of obstacle-avoidance, target tracking and group coordination based on simple forms of implicit communication. All the tests carried out have generated encouraging results, thus suggesting the possibility of further developments.

II. The 3D model

In the new 3D model we describe in this paper, the general task the MAVs are subject to consists of autonomous navigation toward a certain target area - within an obstacle-free environment - relying on a mixture of local and global information. The assumption underlying the model is that an ”upper” system - aware of the location of the target area - is always available and broadcasts this information in real-time to the aircraft. The MAVs can then match this knowledge with proprioceptive information related to their current position and orientation, finding in this way the path to be followed for reaching the target area.

Moving from a 2D to a 3D simulator implies that the degrees of freedom (DoF) available to the simulated aircraft increase from one to three. In the 2D scenario the MAVs rely in fact on a single DoF, since they can just rotate clockwise or anti-clockwise. An object located inside a three-dimensional environment, instead, can rotate around three different axes. Within the aeronautics field [16] these rotations are commonly named as follows: (a) yaw, the rotation around the top-down axis, (b) pitch, the rotation around the wing-to-wing axis, and (c) roll, the rotation around the nose-to-tail axis (see Figure 1). These are the same rotations the aircraft we simulate can perform within our computer model.

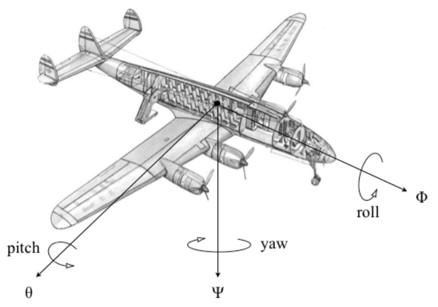

Fig. 1. Graphical representations of the 3 rotations possible for a typical fixed-wing aircraft: yaw, pitch, and roll

From a computational perspective, the introduction of the roll is the most significant addiction to the previous model. According to the current roll angle of the aircraft, in fact, yaw and pitch rotations can produce completely different results. Making difficult, for the controller, to accurately figure out the potential outcome of any given manoeuvre.

A. The computer simulator: technical details

The computer simulator used for this work - written in C++ - has been developed relying on a few open-source libraries, namely Irrlicht[1] as 3D engine, NNFW[2] to manage the neural networks-related aspects, and Qt[3] for multi-threading support. No real physics engines have been used, mainly for two reasons. First, this work explicitly looks at the navigation problem from an higher perspective than what is typically done in control systems literature. In order to focus on intelligent navigation, we assume that the robotic aircraft we simulate are able to respond to high-level commands (e.g., “*yaw* 1° *clockwise*”, or “*pitch* 0.3° *up*”) in the expected way. Once we implement non-physics based but at the same time non-irrealistic aircraft dynamics in our simulator, our purposes are satisfied. Last but not least, ER approach generally requires a significant amount of time for the evolutionary process to reach a steady state. Avoiding a physics engine allows to reduce the impact of this requirement.

The simulator consists of two components running independently: the evolutionary engine and the viewer. The evolutionary engine just performs the computation required for the evolution. The viewer is instead a GUI software capable of loading from the memory an evolved individual and displaying its behaviour to the end user, in order to study the reasons behind certain behavioural patterns emerged.

All the simulations have been carried out on a computer grid managed by Sun Grid Engine[4], consisting of 4 Apple$^{\text{TM}}$Xserve machines (each of those installing two quad-core 2.66GHz Intel$^{\text{TM}}$processors and 4GB of RAM) awarded by our research group through the Apple ARTS program.

B. Neural network controller

The controllers used are constituted for the most part by fully-connected feed-forward neural networks embodied into the MAVs (some variations to this base architectures will be described later on). These controllers are fed with input information coming both from the external environment and from within the robot. Knowledge which is then translated into direct commands for the robot’s motor actuators.

The input the controller receives at any time-step are four: the horizontal angle to the target (Ψ^i), the vertical angle to the target (θ^i), the current roll angle (Φ^i), and the distance from the target (*d*). This information is processed by the network, which in turn affects the activation level of four output neurons. Three of these units (continuous) determine the rotations the MAV will perform in the time unit: yaw (Ψ^o), pitch (θ^o), and roll (Φ^o). The remaining unit (end) is a Boolean one that must be activated by the aircraft only once during its entire life-span, ideally when closer to the target than a certain threshold. All of the non-input and non-Boolean neurons belonging to the network gets activated according to a tan-sigmoid function (slope 1.0), which output values are within $[-1.0; +1.0]$. Summation is the only aggregation function used.

[1]http://www.irrlicht.sourceforge.net
[2]http://www.nnfw.org
[3]http://qt.nokia.com
[4]http://gridengine.sunsource.net/

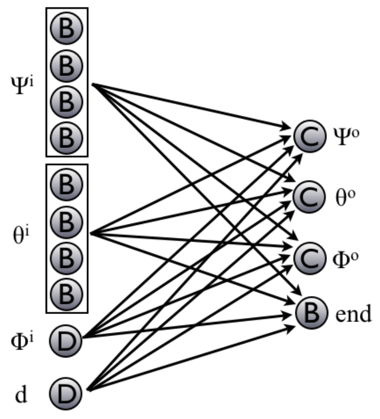

Fig. 2. Example of one of the NN controllers adopted. In this architecture the input is discrete and there are neither hidden layer or memory involved. The aircraft controlled can perform all the possible rotations: yaw, pitch and roll

C. Evolutionary algorithm and general simulation details

Following what the ER paradigm dictates, the proper sets of synaptic weights for the neural network controllers to perform the desired tasks are obtained running an evolutionary algorithm [17], specifically a genetic algorithm (GA).

The starting population used consists in 30 individuals[5], with connection weights and biases randomly assigned at the beginning of the evolution within the $[-10.0; +10.0]$ range. Each controller is tested for four epochs inside a virtual environment - a three-dimensional area with size 1,000 (X) x 1,500 (Z) x 600 (Y) graphical units (GU) - starting each of them from a different position and with the target area randomly dislocated. The aircraft has an approximate length of 3.5GU, while the target is constituted by a sphere with a 15GU radius. A MAV starts each test with 5,000 energy units (EU) available; during each time-step it consumes 1EU, while moving 2GU according to its heading direction. The rotations generated by the controller in the time-unit are included inside the $[-1.0\,°; +1.0\,°]$ range. It is worth noting that, in order to simulate a more realistic flying behaviour, every time the aircraft performs a yaw a corresponding amount of roll is automatically applied. As mentioned above, the Boolean output can be activated only once during the entire individual's life-span. When this neuron turns to 1, as well as when the MAV exits from the environment boundaries or runs out of energy, the current test epoch is immediately considered concluded (successfully or not depending on the MAV-target distance when this operation has took place).

When all the members of the current generation have been evaluated, the five individuals having scored the best performance according to the fitness function in use are selected for reproduction. The best one is copied to the next generation without any modifications (elitism), while the other four are subject to a process of random mutation which affects each of their genes - with probability 0.1 - by a random value picked within $[-0.05; +0.05]$ range. The genome is implemented via parametric encoding, with each gene constituted by a real value, representing either a connection weight or a bias. Five new individuals, with a random set of connection weights and biases, are introduced at any new generation to reduce the risk of premature convergence within the population. The process is iterated for a certain amount of generations and then repeated from the scratch for a few times (we will call each of this run *evolutionary seed*) in order to obtain results the less affected by randomness as possible.

[5]The relatively small number chosen for the population size takes inspiration by the tradition of the Sussex Approach [18].

III. Results

Three different experimental setups have been elaborated. The details of these experiments and the results obtained are described in the following subsections.

A. Plain environment

The first scenario acts as a sort of experimental benchmark. What we are interested in investigating using this setup are the performances generated by different neural networks relying on contrasting input sets and internal organisation.

The fitness formula used for evaluating the behaviour of the aircraft in this setup involves two parameters: α, which represents the average value - across the four epochs of testing - for the differential between the MAV-target distance at the beginning and at the end of the test (thus representing the distance covered on the way to the target); β, indicating instead the overall number of tests succeeded. α is set to 0 in case the MAV has concluded the test because exited from the environment boundaries or ran out of energy. Equation (1) shows the simple way in which this two parameters have been put in correlation.

$$fitness = \alpha + \beta * 100 \tag{1}$$

Various NN architectures have been tested. The variables that have been mixed together in the different architectures are: (a) having the possibility to perform all the rotations described or just a subset of these, (b) short term memory present/absent, (c) hidden layer present/absent, (d) discrete or continuous input stream gathered from the environment. A summary of the 24 architectures analysed is reported in Table I. When the hidden layer is used, the amount of neurons constituting it has been arbitrarily set to 10. According to the presence or the absence of the hidden layer, a basic memory capability has been provided to the networks through either a Elman [19] or a Jordan [20] network.

The desired behaviour has emerged from the evolutionary process in a relatively small amount of generations for most of the experimental setups. An example of this behaviour is visible in Figure 3, while a resume of the main results is included in Table II.

Looking at the results more in details, we can see that the sets of rotations made available to the MAV have a significant impact on its performance. When only yaw is

TABLE I

NEURAL NETWORK ARCHITECTURES TESTED

Arch.	Pitch	Roll	Hid.	Mem.	Input
1	No	No	No	No	D
2	No	No	No	No	C
3	Yes	No	No	No	D
4	Yes	No	No	No	C
5	Yes	Yes	No	No	D
6	Yes	Yes	No	No	C
7	No	No	Yes	No	D
8	No	No	Yes	No	C
9	Yes	No	Yes	No	D
10	Yes	No	Yes	No	C
11	Yes	Yes	Yes	No	D
12	Yes	Yes	Yes	No	C
13	No	No	No	Jordan	D
14	No	No	No	Jordan	C
15	Yes	No	No	Jordan	D
16	Yes	No	No	Jordan	C
17	Yes	Yes	No	Jordan	D
18	Yes	Yes	No	Jordan	C
19	Yes	No	Yes	Elman	D
20	Yes	No	Yes	Elman	C
21	Yes	No	Yes	Elman	D
22	Yes	No	Yes	Elman	C
23	Yes	Yes	Yes	Elman	D
24	Yes	Yes	Yes	Elman	C

TABLE II

RESULTS FROM SIMULATIONS A

Arch.	Av. fitness	Max fitness	Av. succ. rate	Max succ. rate
1	986.2004	1425.6	0.8026	1
2	963.7084	1425	0.7825	0.999
3	856.5177	1393.4	0.6284	0.9915
4	780.2524	1262.4	0.4475	0.7869
5	711.8221	1310.5	0.4641	0.9328
6	383.02	743.4625	0.0071	0.0609
7	988.631	1425.6	0.7952	1
8	976.4259	1428.3	0.7994	1
9	827.5582	1386.3	0.6161	0.9972
10	917.5059	1403.8	0.6809	0.996
11	693.2277	1267.9	0.4095	0.8739
12	599.5794	1045.8	0.2279	0.4684
13	974.5937	1421.3	0.7848	1
14	926.9137	1415.2	0.7465	0.9991
15	715.7472	1266.4	0.3982	0.828
16	452.4319	824.1322	0.0044	0.0805
17	389.929	824.979	0.021	0.1727
18	353.6284	705.1607	0.0009	0.0264
19	790.9081	1345.8	0.598	0.9774
20	779.0397	1340.1	0.5567	0.9692
21	447.198	916.2571	0.0904	0.3263
22	397.223	777.722	0.0138	0.0881
23	319.8735	715.7084	0.0083	0.0854
24	320.8331	639.6556	0.0002	0.007

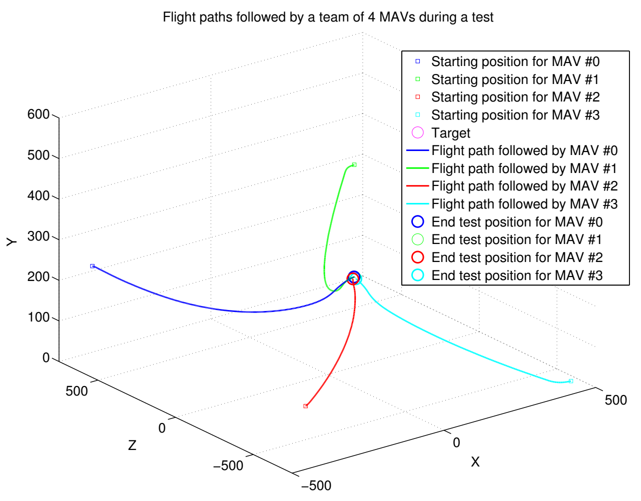

Fig. 3. Example of the flight paths followed by 4 evolved individuals sharing the same controlle, put into the environment at the same time, but moving from different starting positions

permitted (thus recreating, in fact, a 2D scenario) controllers able to perform the desired task with a 100% accuracy level emerge just after a few generations. The same applies when both yaw and pitch are used. The introduction of roll has instead a significant impact and lead to worse performances, that anyway are acceptable for both architectures 5 and 11 (93.28% and 87.39% success rate respectively).

Probably due to simplicity of the elaborated scenario, which does not require any additional ability for the MAV than just pointing to the target area, the architectures providing memory to the controller have not generated any benefits. Even worse, the controllers implementing Elman and Jordan networks have scored significantly lower success rates than those based on purely feed-forward networks. The performances are comparable only for the simplest situation (no pitch and no roll), and partially for the second simplest setup (yaw and pitch, but no roll). This is probably due to the significantly increased search space the evolutionary algorithm has to investigate in order to find an optimum point when the additional connection weights associated to the memory structures come into play.

The use of a hidden layer has proven to be beneficial for the performances of the controllers when no memory structures are employed. With Simulations 5 and 11 being exceptions (in this case, architecture 5, without a hidden layer, has performed better than its counterpart having internal neurons), the controllers with a layer of internal units have in fact outperformed the two-layers networks. The situation is completely different when memory is used. In this case (but again with one exception, constituted by the comparison between Simulations 16 and 22) the lack of a hidden layer seems to be beneficial. As before, this result could be explained through the further increase in dimensionality of the search space generated by the addition of 10 more neurons, with respective synaptic connections and biases.

Finally, using discretised input has always lead to better or sometimes equal results than relying on a continuous encoding. The difference is particularly evident if looking at the results scored by architectures 5 and 6, or 17 and 18.

B. Moving target

The problem of tracking and approaching a movable target is common in the literature. Traditional (engineering)

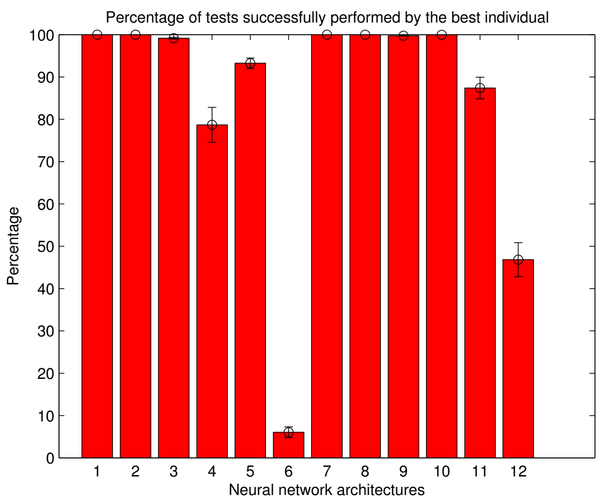

Fig. 4. Barplot displaying the maximum success rate obtained by the best individual evolved with the different controller architectures unprovided of memory structures. The standard error - calculated as standard deviation divided by the number of evolutionary seeds ran - is also shown

approaches to autonomous navigation usually require the robot to elaborate a prediction of the target motion based either on observations or on already available knowledge [21]. Based on this prediction, the robot can then modify its action plan accordingly. The main drawback of this method consists in the fact that extracting the information required and translating it into coherent actions is generally a non-trivial task. It is true that this approach, although generally hard to be implemented, can be effective and lead to good results when carefully implemented. Nonetheless, in presence of a target that does not move according to any specific pattern, the task increases dramatically in terms of complexity, making difficult for a human designer to come out with a working solution. Various authors - such as Bertuccelli [22] and Ablavski [23] - have proposed interesting ways to overcome this issue, respectively focusing on individual or collective behaviours, but the problem remains serious and a definitive solution to it seems far to be achieved. By contrast, we will see in this Section how our ER-based approach to autonomous navigation can be very effective also when dealing with a moving target.

The experimental setup described here is similar to the previous one. The only difference consists in the fact that now the target area can move away from an approaching MAV. At any time step the target can be in either one of two different status: *MAV detected* or *MAV not detected*. When in *MAV not detected* mode, the target scans its surroundings - before the aircraft moves - looking for a MAV within a 35GUs distance from its centre. In case this condition is satisfied, the target switches to *MAV detected* mode with probability 0.5. When the target is in *MAV detected* status at the beginning of any given time step, it has to move to a new place. Table III shows 26 different positions the target can chose between when deciding in which direction to move, based on its current (X, Y, Z) coordinates. These points are calculated in order to be all equidistant from its center (i.e., points on the surface of an imaginary sphere sharing its origin with the centre of the target and having a ray equal to its movement speed) and representative of the entire neighbourhood the target could end up in. The target does not have any kind of preference and its movements are not affected by inertia: at any time step it simply moves to the position which maximise its distance from the MAV.

TABLE III

POSSIBLE MOVING POSITIONS FOR THE TARGET

New X	New Y	New Z
X	Y	$Z + d$
$X + d * \cos(\frac{\pi}{4})$	Y	$Z + d * \sin(\frac{\pi}{4})$
$X + d$	Y	Z
$X + d * \cos(\frac{3}{4}\pi)$	Y	$Z - d * \sin(\frac{3}{4}\pi)$
X	Y	$Z - d$
$X - d * \cos(-\frac{3}{4}\pi)$	Y	$Z - d * \sin(-\frac{3}{4}\pi)$
$X - d$	Y	Z
$X - d * \cos(-\frac{\pi}{4})$	Y	$Z + d * \sin(-\frac{\pi}{4})$
X	$Y + d$	Z
X	$Y + d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(0)$
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{\pi}{4})$	$Y + d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(\frac{\pi}{4})$
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{\pi}{2})$	$Y + d * \cos(\frac{\pi}{4})$	Z
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{3}{4}\pi)$	$Y + d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(\frac{3}{4}\pi)$
X	$Y + d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(\pi)$
$X - d * \sin(\frac{\pi}{4}) * \sin(-\frac{3}{4}\pi)$	$Y + d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(-\frac{3}{4}\pi)$
$X - d * \sin(\frac{\pi}{4}) * \sin(-\frac{\pi}{2})$	$Y + d * \cos(\frac{\pi}{4})$	Z
$X + d * \sin(\frac{\pi}{4}) * \sin(-\frac{\pi}{4})$	$Y + d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(-\frac{3}{4}\pi)$
X	$Y - d$	Z
X	$Y - d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(0)$
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{\pi}{4})$	$Y - d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(\frac{\pi}{4})$
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{\pi}{2})$	$Y - d * \cos(\frac{\pi}{4})$	Z
$X + d * \sin(\frac{\pi}{4}) * \sin(\frac{3}{4}\pi)$	$Y - d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(\frac{3}{4}\pi)$
X	$Y - d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(\pi)$
$X - d * \sin(\frac{\pi}{4}) * \sin(-\frac{3}{4}\pi)$	$Y - d * \cos(\frac{\pi}{4})$	$Z - d * \sin(\frac{\pi}{4}) * \cos(-\frac{3}{4}\pi)$
$X - d * \sin(\frac{\pi}{4}) * \sin(-\frac{\pi}{2})$	$Y - d * \cos(\frac{\pi}{4})$	Z
$X + d * \sin(\frac{\pi}{4}) * \sin(-\frac{\pi}{4})$	$Y - d * \cos(\frac{\pi}{4})$	$Z + d * \sin(\frac{\pi}{4}) * \cos(-\frac{3}{4}\pi)$

The results presented in this section refer to architectures 11 and 5, those that provided the best results in Simulations A - in the most difficult scenario - respectively for the subsets of networks with and without a hidden layer. The aircraft can therefore yaw, pitch and roll. In order to allow the evolution of the more sophisticated behaviour now required, the evolution length has been extended to 20,000 generations and 20 evolutionary seeds have been elaborated. Given M_s the speed of the MAV, five different evolutions have been performed for each architecture, with the target moving respectively at a speeds T_s equal to $\frac{M_s}{5}$, $\frac{M_s}{4}$, $\frac{M_s}{3}$, $\frac{M_s}{2}$ and M_s.

As expected, varying the speed of the target has affected the MAVs' performances. These results are extremely similar to those obtained with our previous work on the 2D simulator [15]. What we can observe is the emergence of a sort of threshold value for T_s that once exceeded makes the MAV unable to succeed in the task anymore. For targets moving at $\frac{M_s}{5}$, $\frac{M_s}{4}$ and $\frac{M_s}{3}$, the success rate for the best individual of the population is about 90%, both when the architecture is lacking a hidden layer (Figure 5), and when the network can rely on this additional computational capability (Figure 6). More specifically, the percentage of tests concluded successfully ranges from 86.57% and 95.71%. A target moving at $\frac{M_s}{2}$ results instead much more difficult to be approached by the MAVs, with the two simulations respectively scoring

54.03% and 59.85%. When the target and the MAV move at the same speed ($T_s = M_s$), the latter practically never succeeds in the task (0.0008% and 0.006%).

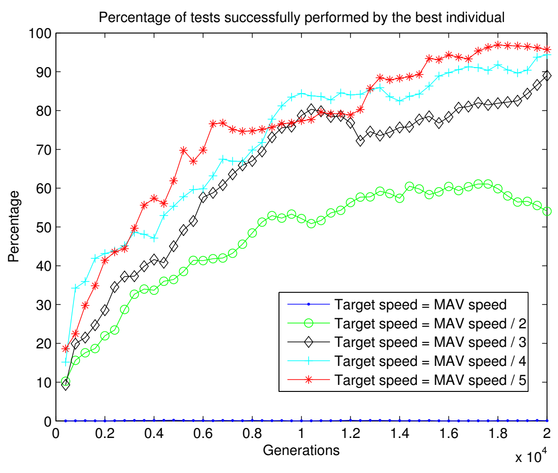

Fig. 5. Percentage of tests concluded successfully for the best individual of the population evolved with neural architecture 5, at different T_s

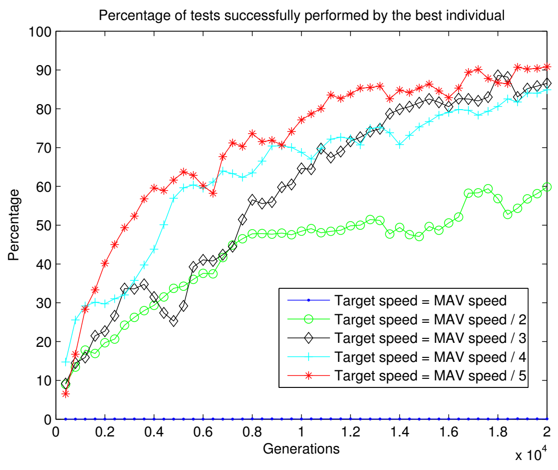

Fig. 6. Percentage of tests concluded successfully for the best individual of the population evolved with neural architecture 11, at different T_s

C. *Implicit cooperation with a non-movable target*

The goal of achieving coordination and cooperation among a team of autonomous robots is a notably interesting problem from both a scientific and technological perspective [24]. This is particularly true when the designer decides to do not rely on a central controller (some criticisms to this approach can be found for example in [25]), but rather opts for pursuing coordination and cooperation in a distributed way [26]. In this section we present the results obtained in a new task, where MAV teams have to achieve coordination among their members - relying on implicit (i.e., non-voluntary) communication strategies - in order to succeed in a task that can not otherwise be achieved individually.

The target area, in this scenario, does not move. The setup is more complicated than the one presented in section B, because of the fact that instead of one single MAV, teams made by 4 MAVs sharing the same controller are now employed. In order to accomplish the test, at least two MAVs need to approach the target area and activate their Boolean neurons in quick succession. From a technical point of view, these modifications have been implemented imposing that the target can be - at any given time - in either one out of two possible alternative states: *sane* or *damaged*. The target starts each test as *sane*, but can be damaged later on during the epoch. The damaging of the target takes place when a MAV activates its Boolean unit within 15GU from the target centre. The test is considered concluded successfully when another MAV manages to correctly approach the target (i.e., activating the *end* unit when close enough to it) when the latter is still *damaged*. In order to guarantee the semi-synchronicity of the process, the target gets restored (i.e., switches back to the *sane* status) after 200 time steps of *damaged* status. The controller has been modified according to the new conditions, with the introduction of two new inputs, specifically two Boolean neurons. These units respectively get activated when the target is in *damaged* status, and when a teammate is perceived within a certain distance range (60GU) from the aircraft embodying the controller. The motion of the MAVs has been modified as well, in order to allow them wider rotation angles. At any time-step they now move just 1GU rather than 2 as in the previous setups. The amount of energy at the beginning of a test has been increased from 5,000EU to 15,000EU. The fitness formula also has required a minor modification in order to cope with the new dynamics and the fact of having MAV teams rather than individual MAVs involved in the evolutionary process.

$$fitness = \langle \alpha \rangle + \gamma * 50 + \beta * 100 \qquad (2)$$

Compared to Equation 1, Equation 2 introduces a new parameter, γ, which represent the amount of tests concluded half-successfully, i.e. with one single MAV managing to properly approaching the target. α has been modified to $\langle \alpha \rangle$, thus indicating that it is now representing the average distance traveled by all the four MAVs during the tests.

The strategy evolved by the best individuals is the same we already observed in [15]. Basically, the first MAV arriving in the proximity of the target does not activate its Boolean unit immediately. It rather starts flying around the target area, waiting for another aircraft to arrive. When a teammate finally arrives, they both activate their Boolean neurons accomplishing the test. An example of this behaviour can be observed in Figure 7.

Unfortunately, this behaviour has demonstrated to be particularly hard to be achieved in the 3D model. Simulations carried out using controllers based on architectures 1, 2, 7, and 8 (no pitch and roll rotations available) have generally generated good performances (success rates for the best individual respectively equal to 89.71%, 39.9%, 93.06%, and

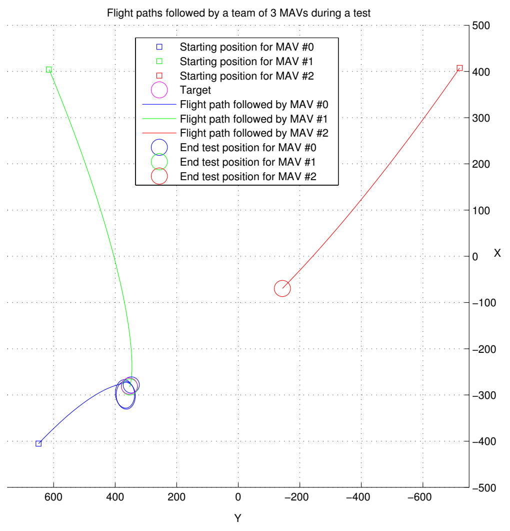

Fig. 7. Example of the flight paths followed by three evolved individuals, evolved with architecture 2, put into the environment at the same time, but moving from different starting positions. In this case MAV #0 is the first to arrive in proximity of the target area in $(-280, 360)$. It then starts flying around it, while waiting for the arrival of a teammate in order to successfully conclude the test

75.38%). Nonetheless the performances have dramatically decreased when the wider sets of rotations allowed to the MAVs have been introduced. Figure 8 shows the results obtained evolving controllers based on architecture 5 (yaw, pitch, and roll possible). In this case, the best team can successfully conclude the task 12.79% of times (just 2.04% for the average team), and simply manages to put the target in *damaged* status the remaining 58.23% of times.

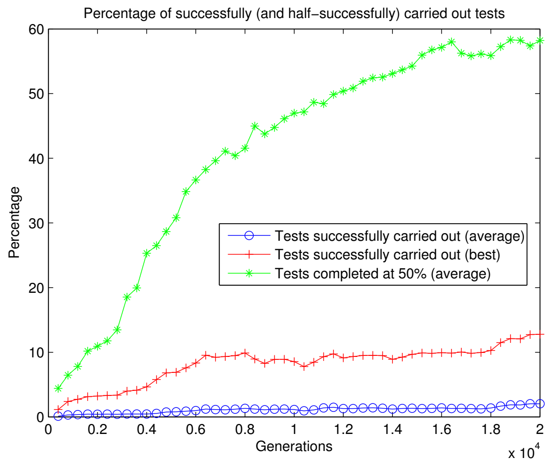

Fig. 8. Percentage of tests concluded successfully for the best and the average individual of the population evolved with neural architecture 5, and percentage of tests concluded half-successfully

IV. Conclusions and Future work

The work described in this paper has confirmed the feasibility of an approach based on Evolutionary Robotics for the development of autonomous controllers dealing with high-precision tasks, such as navigation of fixed-wing flying robots. Comparing the results obtained by the 3D model presented herein with those generated by our preliminary research, we observe that the evolution of the desired behaviours is now significantly more challenging than before. This is not due to a more complex environment the MAVs have to navigate (the 3D environment, free of any obstacle, could be instead considered simpler than the one previously investigated), but rather to the more sophisticated flight dynamics the aircraft are subject to, reflecting in a larger amount of DoF available to their controllers. As we have discussed in Section III, is the roll in particular which make the controlling task so difficult. Nonetheless this is not an insurmountable issue, since it could be overcame on real robotics platforms providing them with devices (i.e. autopilots) that automatically stabilise the aircraft around a 0° roll angle.

Further work is required in order to improve the results obtained so far in the scenario involving implicit communication among the MAVs. New simulations, relying on incremental evolution in order to accelerate the time required by the evolutionary process and to direct the research across the solution space, are being actually developed for this purpose. Once this target will be achieved, future developments will focus on the introduction of more sophisticated tasks, requiring forms of explicit cooperation/communication between MAVs to evolve. At the same time, attempting to fully replicate the results obtained using our previous 2D model, obstacles will be inserted into the environment. The goal will be to replicate a realistic urban-like environment, in order to study possible applications of autonomous MAVs in real-life scenarios.

Modeling field theory [27] [28] has been recently proposed as a learning technique for multi-agent simulation systems that could potentially be applied to our model. One of the advantages of this approach is that of overcoming computational complexity and allowing better scaling up of the model capabilities, e.g. in terms of population size and internal representations. Future studies will explore the combination of modeling field theory with the approach outlined herein.

Finally, we recently started to investigate the possibility of testing the controllers evolved in our computer simulations on physical robotics platforms. Many small-sized R/C aircraft, either modified using off-the-shelf hardware (as demonstrated for example in [29]) or already properly configured (e.g., the senseFly's *swinglet*[6]) can be adapted to the purposes of our research with a relatively small investment, both in terms of time and money. Future experiments will be carried out in collaboration with the EPFL's Laboratory of Intelligent Systems (LIS), following the approach outlined for example in [30]. This methodology relies on the use of a computer board embedded on a swinglet and interfaced with both an autopilot and a Wi-Fi module. In addition to generating commands (mainly turn rates) to be physically

[6]http://www.sensefly.com/products/swinglet

executed, the autopilot takes care of keeping the aircraft stable and flying at a specified speed/altitude. At the same time the Wi-Fi module establishes a communication link with a base station, where a more powerful computer can process the information gathered by the MAV and generate in real-time the flight instructions needed by the aircraft in order to perform the desired task. Adapting such a robotics platform to our purposes is only matter of identifying, installing and configuring the required onboard sensors. At that point, also the investigation of the reality gap effect in the domain of robotic aircraft will become possible.

Acknowledgment

Effort sponsored by the Air Force Office of Scientific Research, Air Force Office Material Command, USAF under grant number FA8655-07-1-3075.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for government purpose notwithstanding any copyright notation thereon.

References

[1] S. Nolfi and D. Floreano, *Evolutionary robotics. The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press, 2000.

[2] D. Floreano and L. Keller, "Evolution of adaptive behaviour in robots by means of darwinian selection," *PLoS Biology*, vol. 8, no. 1, pp. 1–8, 2010.

[3] F. Mondada, E. Franzi, and A. Guignard, "The development of khepera," *Proceedings of the 1st International Khepera workshop*, pp. 7–13, 1999.

[4] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pp. 59–65, 2009.

[5] J. C. Zagal, J. R. del Solar, and P. Vallejos, "Back-to-reality: Crossing the reality gap in evolutionary robotics," *Proceedings of IAV 2004, 5th IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[6] K. Nonami, "Prospect and recent research & development for civil use autonomous unmanned aircraft as uav and mav," *Journal of System Design and Dynamics*, vol. 1, no. 2, pp. 120–128, 2007.

[7] M. Dong and Z. Sun, "A behavior-based architecture for unmanned aerial vehicles," *Proceedings of the 2004 IEEE International Symposium on Intelligent Control*, pp. 149–155, 2004.

[8] G. Barlow and C. Oh, "Evolved navigation control for unmanned aerial vehicles," *Frontiers in Evolutionary Robotics*, pp. 596–621, 2008.

[9] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of uav through uncertain environments," *Proceedings of the AIAA Digital Avionics Systems Conference*, pp. 608–619, 2002.

[10] G. Buskey, J. Roberts, P. Corke, P. Ridley, and G. Wyeth, "Sensing and control for a small-size helicopter," *Experimental Robotics VIII*, pp. 476–486, 2003.

[11] R. D. Nardi, O. Holland, J. Woods, and A. Clark, "Swarmav: A swarm of miniature aerial vehicles," *Proceedings of the 21st Bristol UAV Systems Conference*, p. 9, 2006.

[12] S. Hauert, L. Winkler, J.-C. Zufferey, and D. Floreano, "Ant-based swarming with positionless micro air vehicles for communication relay," *Swarm Intelligence*, p. 22, 2008.

[13] J.-C. Zufferey, A. Beyeler, and D. Floreano, "Near-obstacle flight with small uavs," *Proceedings of UAV 2008, International Symposium on Unmanned Aerial Vehicles*, 2008.

[14] F. Ruini, A. Cangelosi, and F. Zetule, "Individual and cooperative tasks performed by autonomous mav teams driven by embodied neural network controllers," *Proceedings of IJCNN 2009, International Joint Conference on Neural Networks*, pp. 2717–2724, 2009.

[15] F. Ruini and A. Cangelosi, "Extending the evolutionary robotics approach to flying machines: An application to mav teams," *Neural Networks*, no. 22, pp. 812–821, 2009.

[16] M. Cook, *Flight dynamics principles*. London/Philadelphia, PA: Elsevier, 2007.

[17] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, MA: MIT Press, 1998.

[18] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi, "Evolutionary robotics: the sussex approach," *Robotics and Autonomous Systems*, vol. 20, pp. 205–224, 1996.

[19] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[20] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Artificial Neural Networks: Concept Learning*, pp. 112–127, 1990.

[21] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House Publishers, 1999.

[22] L. Bertuccelli and J. How, "Uav search for dynamic targets with uncertain motion models," *Proceedings of the 45th IEEE Conference on Decision & Control*, vol. 5941-5946, pp. 1–6, 2006.

[23] V. Ablavsky, D. Stouch, and M. Snorrason, "Search path optimization for uavs using stochastic sampling with abstract pattern descriptors," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[24] G. Nitschke, "Emergence of cooperation: State of the art," *Artificial Life*, vol. 11, no. 3, 2005.

[25] A. Wu, A. Schultz, and A. Agah, "Evolving control for distributed micro air vehicles," *Proceedings of CIRA '99, IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 174–179, 1999.

[26] G. Baldassarre, D. Parisi, and S. Nolfi, "Distributed coordination of simulated robots based on self-organization," *Artificial Life*, vol. 12, no. 3, pp. 289–311, 2006.

[27] R. Deming, L. Perlovsky, and R. Brockett, "Sensor fusion for swarms of unmanned aerial vehicles using modeling field theory," *Proceedings of Kimas 2005*, pp. 122–127, 2005.

[28] L. Perlovsky, *Neural networks and intellect: using model-based concepts*. Oxford, UK: Oxford United Press, 2000.

[29] P. Thomas and A. Cooke, "Simulation of an automated flight test safety system for autonomous system identification of small uavs," *Proceedings of UAVs 2009, Twenty-Fourth International Conference on Unmanned Air Vehicle Systems*, pp. 35.1–35.16, 2009.

[30] S. Hauert, S. Leven, J.-C. Zufferey, and D. Floreano, "Communication-based leashing of real flying robots," *Proceedings of ICRA 2010, IEEE International Conference on Robotics and Automation*, 2010.