An Evolutionary Approach to Tuning a Multi-Agent System for Autonomous Adaptive Control of a Flapping-Wing Micro Air Vehicle

Michal Podhradsky* and Garrison Greenwood

Dept of Electrical and Computer Engineering, Portland State University, Portland, OR 97207, US

Abstract: Biomimetic flapping wing vehicles have attracted recent interest because of their numerous potential military and civilian applications. In this paper, we describe an evolutionary approach to tuning a Multi-Agent System for autonomous adaptive control of a Flapping-Wing Micro Air Vehicle. The wings of the vehicle are controlled by a split cycle oscillator, which combined with non-linearities and differences between each vehicle, brings significant challenge for selecting the proper parameters for the control system. Adopting a Neo-Darwinistic evolutionary approach, where solutions are evolved in a similar manner as in nature, allows us to precisely learn control parameters for each vehicle. After describing the evolution algorithm and evolving the control parameters, we utilize these values for autonomous waypoint following by the micro air vehicle.

Keywords: Multi-agent control, evolutionary algorithm, flapping-wing micro aerial vehicle.

1. INTRODUCTION

Biomimetic flapping-wing vehicles have been the focus of much recent research due to their potential for both civilian and military application. In either possible application area, adaptive, fault-tolerant, control is paramount. This paper de- scribes a multiagent system for adaptive control of a model biomimetic flapping-wing vehicle.

The flapping-wing vehicles were thoroughly studied – the properties of flapping-wings are both theoretically and experimentally measured [1], various systems were designed, built and tested [2, 3], and different wing configurations were analysed [4, 5]. Even a wing design and optimization tool is available [6] for the researchers. Different control approaches were used for control of the vehicles - ranging from simple open loop control [7], through classical control [8, 9], modern control [10], and adaptive control [11], to vision based control systems [12].

The vehicle employed in this research is a hardware analogue of a minimally-actuated flapping-wing vehicle introduced by Wood [13, 14] with core control laws introduced and subsequently refined by Doman *et al*. [15, 16]. The analogue vehicle [17-19] operates similarly to the minimally actuated vehicles considered by Wood and Doman *et al*. in that all propulsion and control are provided by two minimally actuated wings, each of which possesses a single active and a single passive degree of freedom. It differs in that the analogue vehicles active degrees of freedom are driven by a DC motor through a four-bar linkage instead of a piezoelectric transducer and in that the analogue vehicle is mounted vertically on a circular puck and supported from below by either air or fluid cushion. These changes allow us to experiment with the control of translation and roll without the need to address the practical difficulties of balancing lift and vehicle weight.

2. VEHICLE DESCRIPTION

2.1. Vehicle Configuration

A conceptual vehicle closely related to those described by Wood and Doman et al. is presented in [20]. The physical analogue descendant from it is described in [17-19]. Both vehicles operate in a qualitatively similar manner with two minimally actuated wings providing all propulsion and control forces. They differ only in scale and in that the physical vehicle is supported from below by a fluid or air cushion. The physical vehicle, with its externally provided lift support, approximates a passively upright-stable version of the vehicle in [20] operating near its hover wing flapping frequency. Both vehicle types (hereafter referred to as ”the vehicle”) have two wings mounted in the $X_b - Y_b$ body plane (see Figure **1**). These wings are actively actuated within the range of $\pm \varphi$. As the spars rotate, dynamic air pressure lifts the triangular wing platforms (membranes) up to an angle of α radians under a base vector embedded in the $Y_b - Z_b$ plane. Individual wing flaps produce independent lift and drag forces at each of the two wing roots (points of attachment of the wings to the body). These can be resolved into body frame forces and torques and cause changes in the whole

*Address correspondence to this author at the Dept of Elec and Comp Engr, Portland State University Portland, OR 97207, US; Tel: 503-725-3806;
E-mail: podhrad@pdx.edu, michal.podhradsky@pdx.edu

vehicles position and pose in three space. The wings are controlled by Cycle Averaged/Split Cycle oscillator as described below.

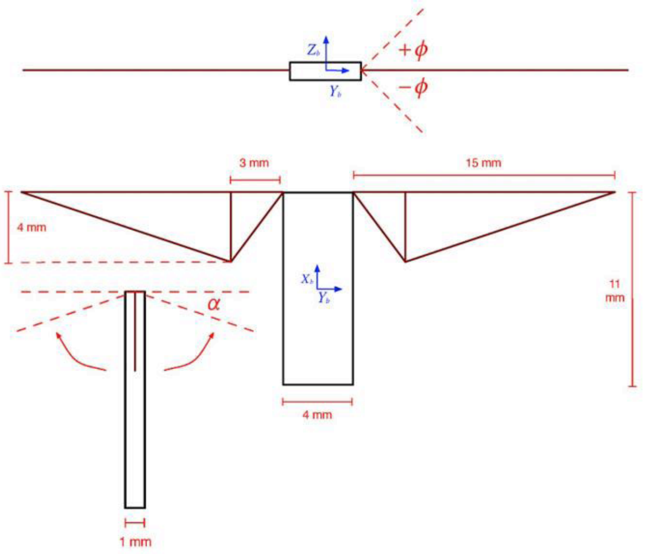

Figure 1: Orthographic view of flapping wing vehicle [20]. Both wing spars are restricted to rotational motion about their joints with the body and in the *Yb –Zb* plane. The range of those rotations is [−1...1] radians, *α* is between *π/*6 and *π/*2 radians. Note that the dimensions are for orientation purposes only, and differ on the actual vehicle.

2.2. Cycle Averaged / Split Cycle Control

The instantaneous wing angle φ can be described by a cosine function $\varphi = \cos(\omega t)$, where ω is the wing beat frequency. During a regular wingbeat, the upstroke and down- stroke are symmetrical, and the only force produced is the lift perpendicular to the wing plane.

In a split cycle control, the wing motion consists of two cosine waves, φ_U for upstroke and φ_D for downstroke. Ad- vancing the upstroke (and consequently impeding the down- stroke) produces a lateral force while keeping the wing beat frequency constant. Formally, $\varphi_U = \cos((\omega - \delta)t)$ where δ is the frequency modifier, and $\varphi_D = \cos((\omega + \sigma)t)$ where σ is dependent on δ.

From [16] we know that $\delta \in [-\infty \ldots \omega/2]$ although certain value ranges are particularly important. If $\delta = 0$ the upstroke is symmetrical to the downstroke and a regular wingbeat occurs. However, if $\delta > 0$ the upstroke is impeded and the downstroke is advanced, as shown in Figure **2**. As a result, a force is generated in the direction of the downstroke. Conversely, if $\delta < 0$ then the downstroke is impeded and the upstroke is advanced, as shown in Figure **3**, resulting in force in the direction of the upstroke. These lateral forces act on the vehicle's body *via* a moment arm producing an angular momentum. Put simply, by applying the split cycle the vehicle can turn. See [16] for a derivation and proof of split-cycle operation.

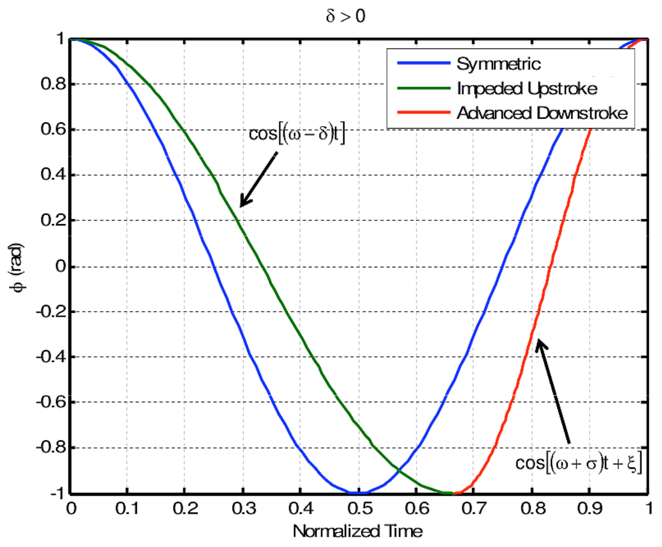

Figure 2: Split-cycle results for $\delta > 0$.

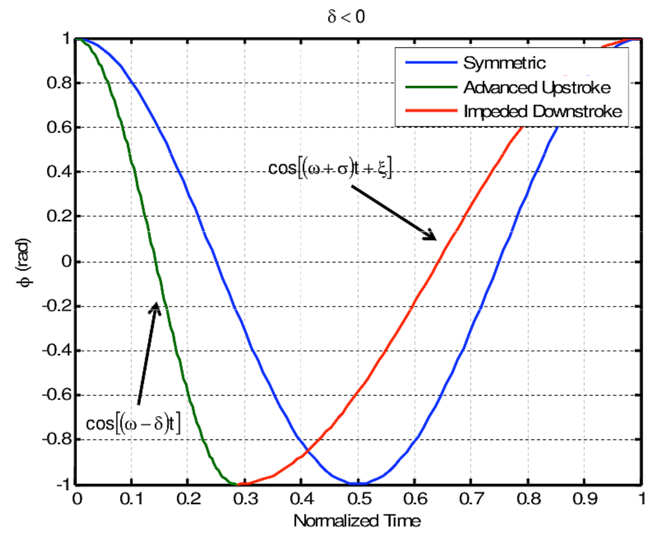

Figure 3: Split-cycle results for $\delta < 0$.

2.3. Hardware Realization

Our vehicle uses stepper motors and mechanical linkages to translate rotational movement of the motor shaft to flapping movement of wings. The stepper motor is able to realise split cycle control – e.g., making an upstroke faster than a down- stroke while keeping the wing beating frequency constant. A model of the wing assembly is shown in Figure **4**.

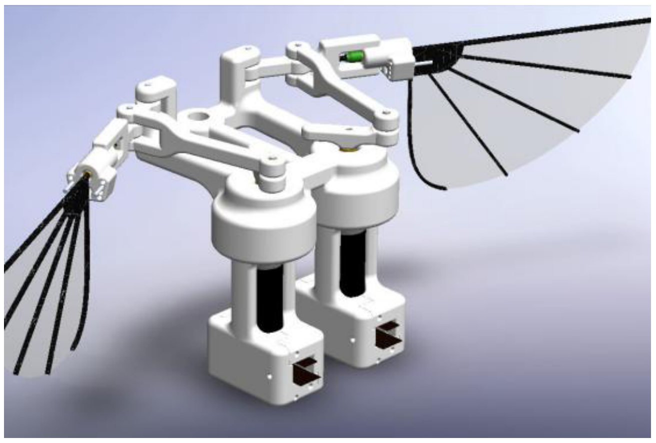

Figure 4: 3D model of wings with linkages and motors (*http://cps.wright.edu/*).

The details about the hardware and software needed for control can be found in [20]. It is important to emphasise wing control (i.e. translation of *ω* and *δ* values into stepper motor commands) is taken care of by an inner control loop and the outer loop control (navigation) cannot directly interfere with it. This outer control loop is implemented by the Multi-Agent System (MAS), described below.

2.4. Multi-Agent Control System

The MAS consists of five agents. A *collection agent* receives pose information *x, y, ψ* from the camera (or simulator), and runs smoothing and averaging algorithms to compute the estimated pose *x, y, ψ*. A *monitor agent* observes vehicle behaviour and requests vehicle diagnostics if the behaviour has deteriorated too much. The *strategy agent* keeps a list of desired waypoints, and provides them upon request to a *controller agent*. The controller agent determines the split- cycle oscillator control inputs (a *δ* and *ω* for each wing) based on the vehicle pose. Finally, a *diagnostic agent* runs vehicle diagnostics and determines if a fault occurred and ultimately Randomly generate the initial population; 2. Ev decides whether the controller agent's rule base has to be adapted. The MAS diagram is shown in Figure **5**, for more details about the MAS refer to [21].

3. BASIC MOVEMENTS EVOLUTION

To follow waypoints using the multi-agent system described in the previous section, we first need to find the appropriate combination of *δ* and *ω* values for the basic vehicle movements. These basic movements (*Move forward, Left turn, Right turn*) are summarized in Table **1**. Every vehicle is slightly different due to inherent non-linearities such as slip between linkages. Thus, the same *δ* and *ω* values cannot be used for every vehicle; they must be learned.

The *δ* and *ω* values will be determined during this initial learning phase using a combination of *extrinsic* and *intrinsic* evolution [22]. The idea is that the new solutions are evolved from existing solutions by emulating Neo-Darwinistic evolution found in nature. In other words, the search for good *δ, ω* values is conducted by evolving them. Highly fit solutions result in good micro air vehicle behavior. The difference between intrinsic and extrinsic evolution is in how the fitness is determined. In *extrinsic* evolution computer models evaluate a given *δ, ω* value set. In *intrinsic* evolution the *δ, ω* values are downloaded to hardware and physical tests are conducted.

Both forms of evolution use a form of an *evolutionary algorithm* (EA) to conduct the search. An EA consists of a population of individuals, where each

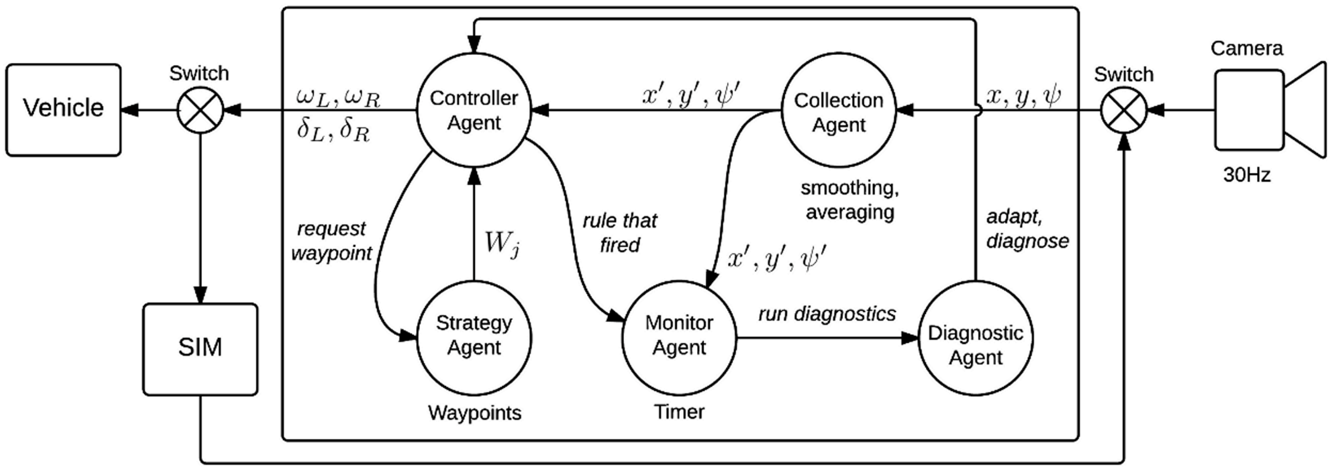

Figure 5: Diagram of Agent-based control architecture. More information about each agent is provided in Section II-D.

individual represents a particular solution to a given problem. New individuals are created from existing individuals *via* random mutation, but only the highly fit ones will survive. (The fitness formula is given later in this section.) An EA runs for a fixed number of generations, at the end of the run the fittest individual is the final solution. The EA steps are shown below:

The needed parameters will be evolved using an EA where one individual is chosen as a parent, and ten off spring are generated from this parent using a mutation operator, which with a certain probability changes the properties of the individual. The individuals are encoded as;

$\{\delta L, \delta R, \omega L, \omega R, \sigma\delta L, \sigma\delta R, \sigma\omega L, \sigma\omega R\}$

where the first 4 parameters are *object parameters* (elements of the solution) and the second 4 parameters are *strategy;*

```
1. Randomly generate the initial population;
2. Evaluate the fitness of the initial population;
while max number of generation not reached: do
    i. Select the best individuals for reproduction;
    ii. Generate new individuals via random mutation;
    iii. Evaluate the fitness of new individuals;
    iv. Discard the least fit individuals;
end
```

Algorithm 1: A basic evolutionary algorithm.

Table 1: Basic Vehicle Movements. ↑, ↓ Indicate Direction of Change, not its Magnitude

Movement	Control Input
Move Forward	$\delta_L \uparrow$ & $\delta_R \uparrow$ (identical)
Left Turn	$\delta_L \downarrow$ & $\delta_R \uparrow$ (opposite)
Right Turn	$\delta_L \uparrow$ & $\delta_R \downarrow$ (opposite)
Idle	$\delta_L = \delta_R = 0$

parameters used to control the mutation step size. The *object parameters* are mutated independently using a normal distribution and the appropriate *strategy parameter*. The equation for production of a new individual *y* from a single parent *x* is given as:

$$\delta_{Ly} = \delta_{Lx} + N(0, \sigma_{\delta L}) \quad (1)$$

where $N(0, \sigma_{\delta L})$ is a normally distributed random variable with zero mean and a standard deviation of $\sigma_{\delta L}$. The other *object parameters* with their respective *strategy parameters* are mutated the same way as described in the Equation 1.

In our algorithm, the parent lives only one generation, because the next generation will be taken from the offspring only. The fittest individual is selected to be the parent in the next generation. The fittest individual encountered is recorded, but it is not a part of the population in future generations. This way the best solution isn't lost. The first parent is randomly initialized, using lower and upper bounds on δ and ω that are needed because of hardware limitations. Hardware limits max value of ω to 30 rad/s (minimum is 1 rad/s), and $\delta \in (-10, 10)$. Split-cycle control requires $|\delta| \leq \omega/2$.

In this phase of the research effort, we used an evolutionary algorithm to search for the optimal values of δ and ω. Other more classical optimization algorithms might be useful but whether or not that is true would require a detailed analysis of the solution space morphology which we did not do1.

Our choice of an evolutionary algorithm to conduct the search was two-fold. First, evolutionary algorithms are usually considered optimization algorithms but basically they are search algorithms. Evolutionary algorithms can search any[1] solution space regardless of morphology. Thus evolutionary algorithms allow us to optimize without conducting a solution space analysis. Secondly, and more importantly, every vehicle is slightly different due to inherent nonlinearities in the linkages and other manufacturing differences (such as a slightly different size of wings, etc.). As a result, *optimal* values for one vehicle will not be optimal for another. The goal here is not to achieve generally optimal movements but rather smooth and repeatable correct movements. Consequently, we needed a search method rather than an optimization method. Evolutionary algorithms allow us to search for good solutions by evaluating actual vehicle behavior, which cannot be accomplished using classical optimization algorithms. This type of search process is called *intrinsic evolution*.

[1]However, our experiments did show small perturbations in δ/ω had no observed behavioral changes which suggests gradient-based optimization algorithms would not be very effective

3.1. Extrinsic Evolution

Because the lifespan of linkages at the vehicle is limited, it is reasonable to first execute extrinsic evolution of control parameters *δ* and *ω* to 1) verify the correctness of the evolution algorithm; 2) get the initial estimate of optimal control parameters. The more precise model of the vehicle is available, the better is the estimate. However, obtaining a precise (first-principle) model of the flapping wing system is very complicated, especially because of the small forces and torques that would have to be measured to correctly identify the model. Modelling non-linearities such as linkage slip also poses a significant challenge. However a simplified model that treats the vehicle as a point mass body and aggregates the generated forces and torques is a sufficient approximation, because it will exhibit similar behavior albeit on a different time scale.

For the purpose of the extrinsic evolution, we started with the following assumption. The faster the wings beat (i.e. higher *ω*), the more force is generated (because the wing acceleration is higher). The higher split cycle shift (higher /*δ*/), the more force is generated (because the difference between upstroke and downstroke is higher). The higher force results into faster movement.

The optimal solution completes the basic movement in shorter time, and is within the imposed constraints. Thus in our simple model we use to evaluate fitness of candidate solution we employ the following equation:

$$fit(x) = K_L.\delta_{Lx}.\omega_{Lx} + K_R.\delta_{Rx}.\omega_{Rx} \qquad (2)$$

where K_1 , K_2 are adjustable weights, in the simplest case:

- $K_L = K_L = 1$ for Move forward
- $K_L = 1$; $K_2 = -1$ for Turn right
- $K_L = -1$; $K_2 = 1$ for Turn left

We ran the EA for 20 generations in each run, for 20 runs total. The expected optimal solution would converge to maximal *ω* for both wings and maximal values for *δ* but with opposite signs in case of turns. The results are shown in Figure **6**. Notice in all cases the runs converged to (or at least very close to) the global optimum.

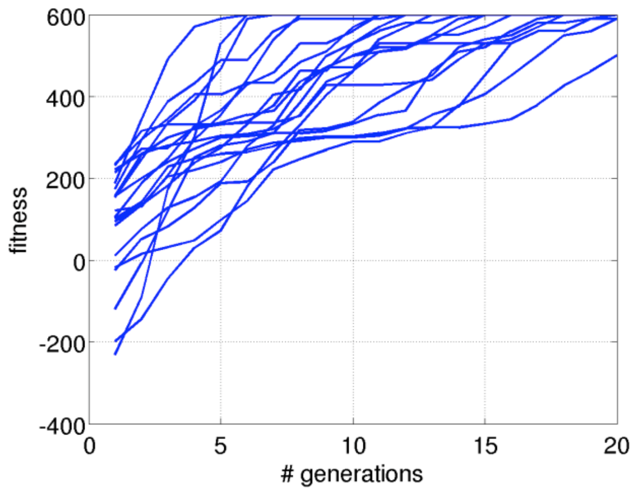

Figure 6: Extrinsic evolution run for *Turn left* move. Notice that the algorithm in almost all cases reaches global optimum *f* = 600.

The best evolved values for our simple model were:

- Turn left: $\delta_L = -10$, $\delta_R = 10$, $\omega_L = \omega_R = 30$
- Turn right: $\delta_L = 10$, $\delta_R = -10$, $\omega_L = \omega_R = 30$
- Move forward: $\delta_L = 10$, $\delta_R = 10$, $\omega_L = \omega_R = 30$

which is consistent with our expectations.

3.2. Intrinsic Evolution

During the intrinsic evolution we used the actual vehicle for evaluating fitness of the candidates. The major difference was that for *Turn left* and *Turn right* fitness is defined as *f* = 1/*T* where *T* is the time needed for the vehicle to turn by 360 degrees from its initial position. For *Move forward* the fitness is defined as $f = 1/(K_1.\ T_{wp}.+\ K_2.\ d_{wp})$, where T_{wp} is time needed to reach x-coordinate of the waypoint *p* (located approximately 30cm in front of the vehicle), d_{wp} is the distance from the y-coordinate of the waypoint *p* when its x-coordinate is reached. $K_1 = 100$ and $K_2 = 1$ are weights to scale the different units (seconds and pixels).

Because the hardware has a limited lifespan, we only ran the EA once and for only 20 generations. This is limiting in the sense that we can reach suboptimal results, but if we were to run more runs as was the case for extrinsic evolution, the linkages could wear out prematurely and would have to be replaced, in which case the learning would have to be done again from the very beginning. In other words, extrinsic evolution provides relatively good results, but to obtain a fine-tuned control system we need to use intrinsic evolution.

The incremental improvements in turn times for evolved control parameters are shown in Figure **7**. For the forward motion, the vehicle actually was not able to reach the desired waypoint (its *y*-coordinate) in vast majority of tries. In such case the experiment was stopped after 2 minutes and the fitness of given individual was marked as zero. Effectively this reduced the EA to a random search, until a viable solution was found. The best solution after 20 generations (and the only one found that had non-zero fitness) is shown in Figure **8**. The best values of control parameters found for our vehicle are summarized in Table II. Notice the best values are very similar to the values found by extrinsic evolution. This validates our model used for extrinsic evolution as usable for initial estimate. The differences in control parameters are caused by imperfections and non-linearities in real hardware, and indeed, those were not included in our simple model. The intrinsically evolved values were used in the next section to perform waypoint following.

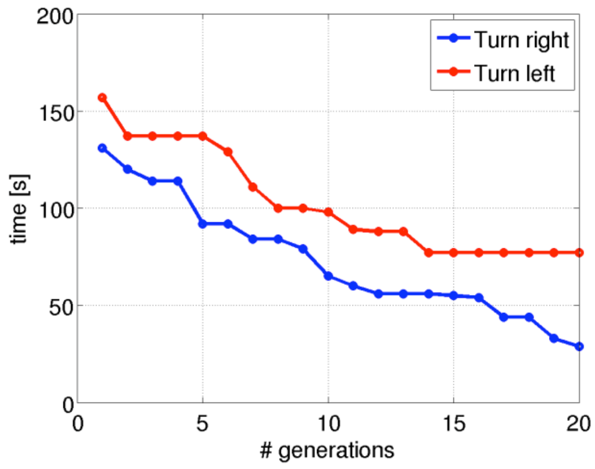

Figure 7: Time needed to complete *Turn left* and *Turn right* moves during intrinsic evolution. Notice that left turn takes longer to finish, which is caused by non-linearities in the hardware.

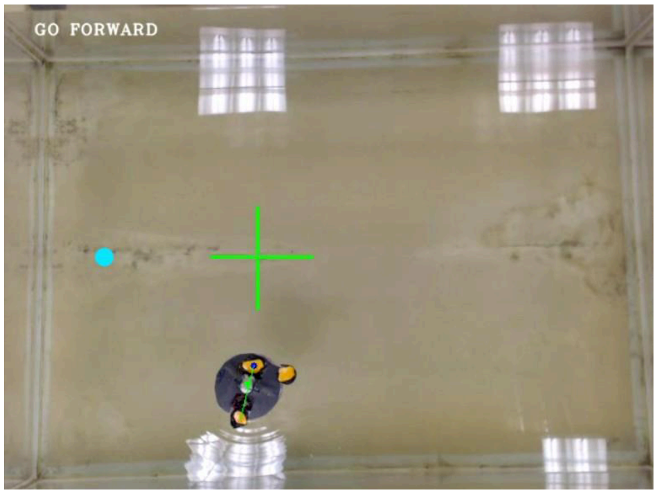

Figure 8: Best found solution for forward movement *Blue:* Initial position of the vehicle, *Green cross:* waypoint the vehicle was commanded to reach. The experiment was stopped once the center of vehicle crossed the *y*-axis of the waypoint.

4. WAYPOINT FOLLOWING

Our experiments are conducted in a large (5 × 7) water tank. The vehicle currently can move on a two-dimensional plane and rotate around its X_b axis. The vehicle is equipped with a pair of Lithium Polymer batteries, a power distribution board, and the main computer, as shown in Figure **9**. All hardware is mounted on a carbon-fiber platform, attached to a floating Styrofoam puck. The water surface acts as a low- pass filter, slowing down the vehicle movement and dampening disturbances.

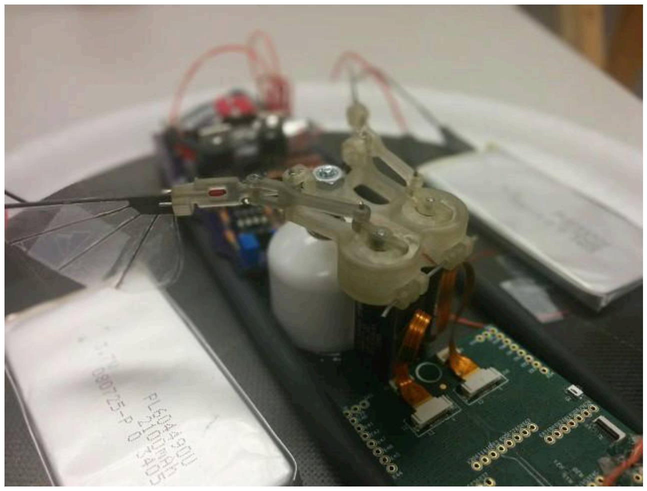

Figure 9: Assembled vehicle before the experiment. Note wings in the middle, LiPo batteries on sides, the power distribution board in the back and the control board in front.

Table 2: Control Parameters for the Basic Movements. Values for *Idle* were Determined Empirically and Based on the Hardware Initialization Procedure (Default Values)

Movement	δ_L	δ_R	ω_L	ω_R
Move Forward	0	0	25	30
Left Turn	0	10	12	30
Right Turn	10	-10	30	30
Idle	0	0	12	12

A camera is placed above the water tank, such as its field- of-view encompasses the entire water tank.

The camera locates and records the vehicle position. The vehicle has color markers for this purpose. Figure **10** shows a close-up view of the vehicle, including the color markers.

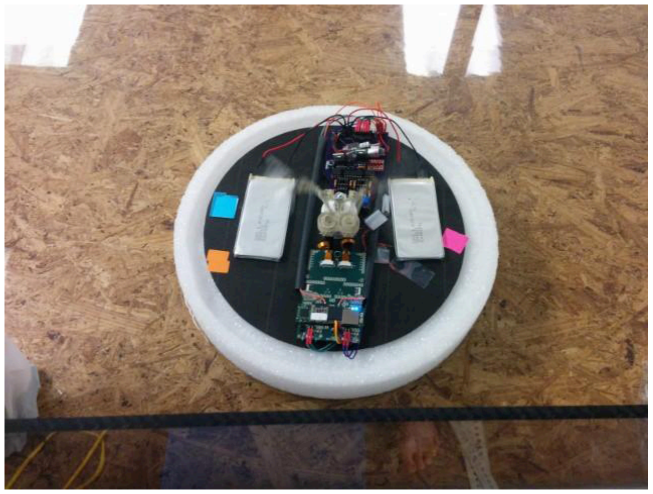

Figure 10: Vehicle during an experiment in the water tank (close-up view). Note the color markers used for machine vision pose estimation.

The video stream from the camera is processed on a regular laptop computer, using the *OpenCV* library for computer vision. The processed video is recorded for reference, and the estimated pose is sent to the onboard MAS *via* a WiFi link.

After learning control parameters for the basic movements the multi-agent system [21] can perform autonomous waypoint following. The experiment is held in a large water tank to lock the vehicle into 2D of freedom and to mechanically low-pass noise in the control system by using water surface as a damper. Machine vision is implemented to track the position of the vehicle using color markers. The waypoints are virtual and dynamically placed in the experiment area. For more details about the experimental setup, the reader should see [21].

Two waypoints were placed to the opposite sides of the water tank, and the vehicle was expected to go back and forth between them. The vehicle was able to follow successfully the waypoints, as can be seen in Figures **11** and **12**. Video of the experiment can be seen at *https://youtu.be/oYEjCaRoLNU.*

5. FUTURE WORK

Both the evolution of control parameters and the consequent waypoint following performed as expected, proving the viability of this concept. The immediate next steps in our work are to implement obstacle avoidance mechanism in the multi-agent system, so the vehicle is able to operate in an environment with obstacles, being more closely related to a real application.

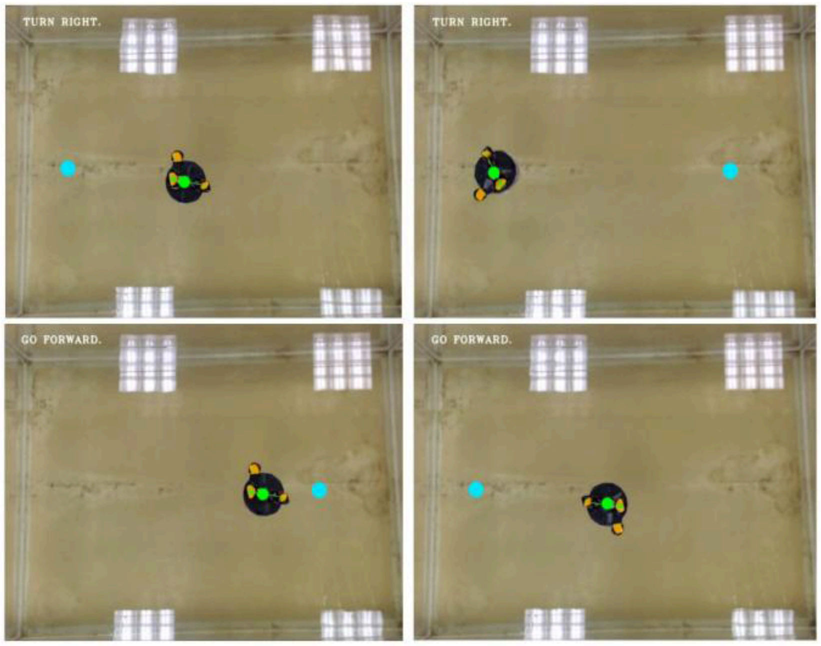

Figure 11: Autonomous waypoint following. The blue dot is the desired waypoint; the vehicle is marked with a bright green dot and a green line pointing towards the front of the vehicle. In the top left corner of the screen is shown the rule that fired. Top left: Initial position of the vehicle; Top right: First waypoint achieved, the vehicle is turning around; Bottom left: Approaching the second waypoint; Bottom right: Second waypoint achieved, moving back to the first waypoint. See video of the experiment at: *https://youtu.be/oYEjCaRoLNU.*

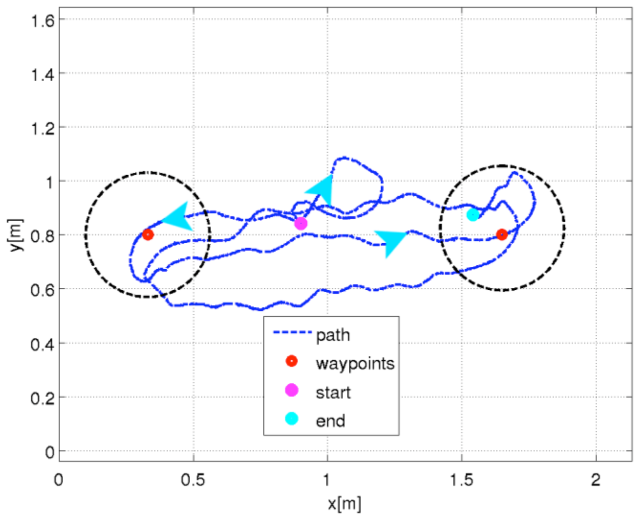

Figure 12: Vehicle trajectory (blue) during waypoint following. Two waypoints being followed are marked with red dots, with distance threshold pictured around them. The vehicle itself is as large as the circle around the waypoints. *Purple:* start position, *Light blue:* end position, *Blue arrows:* indicate orientation of the vehicle.

The second step is to implement in-flight learning and fault- detection function as proposed in [21], so the vehicle can recover from certain faults and finish its mission even if it sustains damage in the control system.

Note that the evolutionary algorithm used in this work is only one of many existing evolutionary algorithms. More sophisticated algorithms, such as *Artificial Bee Colony* [23] or *Particle Swarm Optimization* [24] can be used and compared and evaluated. However, for the scope of this work the basic evolutionary algorithm provided sufficiently good results.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant Numbers CNS-1239196, CNS-1239171, and CNS-1239229.

REFERENCES

[1] Desbiens AL, Chen Y and Wood RJ. A wing characterization method for flapping-wing robotic insects. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems 2013; 1367-1373.

[2] Leys F, Vandepitte D and Reynaerts D. Design of a flapping wing micro air vehicle, based on the rufous hummingbird. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO) 2015; 1266-1271. http://dx.doi.org/10.1109/ROBIO.2015.7418945

[3] Lau GK, Chin YW, Goh JTW and Wood RJ. Dipteran-insect-inspired thoracic mechanism with nonlinear stiffness to save inertial power of flapping-wing flight. IEEE Transactions on Robotics 2014; 30(5): 1187-1197. http://dx.doi.org/10.1109/TRO.2014.2333112

[4] Hines L, Colmenares D and Sitti M. Platform design and tethered flight of a motor-driven flapping-wing system. In 2015 IEEE International Conference on Robotics and Automation (ICRA) 2015; 5838-5845. http://dx.doi.org/10.1109/ICRA.2015.7140016

[5] Colmenares D, Kania R, Zhang W and Sitti M. Compliant wing design for a flapping wing micro air vehicle. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on 2015; 32-39. http://dx.doi.org/10.1109/iros.2015.7353111

[6] Arabagi V, HinesL and Sitti M. A simulation and design tool for a passive rotation flapping wing mechanism. IEEE/ASME Transactions on Mechatronics 2031; 18(2): 787-798. http://dx.doi.org/10.1109/TMECH.2012.2185707

[7] Phan HV and Park HC. Remotely controlled flight of an insect- like tailless flapping-wing micro air vehicle. In Ubiquitous Robots and Ambient Intelligence (URAI), 2015 12th International Conference on 2015; 315-317.

[8] Zhang J, Cheng B, Yao B and Deng X. Adaptive robust wing trajectory control and force generation of flapping wing mav. In 2015 IEEE International Conference on Robotics and Automation (ICRA) 2015; 5852-5857. http://dx.doi.org/10.1109/ICRA.2015.7140018

[9] Mahjoubi H and Byl K. Dynamics of insect-inspired flapping-wing mavs: Multibody modeling and flight control simulations. in 2014 American Control Conference 2014; 3089-3095. http://dx.doi.org/10.1109/ACC.2014.6858637

[10] Zhang J. Cheng B, Roll JA, Deng X and Yao B. Direct drive of flapping wings under resonance with instantaneous wing trajectory control. In Robotics and Automation (ICRA), 2013 IEEE International Conference on 2013; 4029-4034.

[11] Bayandor J. Bledt G, Dadashi S, Kurdila A. Murphy I and Lei Y. Adaptive control for bioinspired flapping wing robots. In 2013 Amer- ican Control Conference 2013; 609-614.

[12] Hsiao FY, Yang LJ, Lin SH, Chen CL and Shen JF. Autopilots for ultra lightweight robotic birds: Automatic altitude control and system integration of a sub-10 g weight flapping-wing micro air vehicle. IEEE Control Systems 2012; 32(5): 35-48. http://dx.doi.org/10.1109/MCS.2012.2205475

[13] Wood RJ. The first takeoff of a biologically inspired at-scale robotic insect. IEEE Transactions on Robotics 2008; 24(2): 341-347. http://dx.doi.org/10.1109/TRO.2008.916997

[14] Teoh Z, Fuller S, Chirarattananon P, Prez-Arancibia N, Greenberg J and Wood R. A hovering flapping-wing microrobot with altitude control and passive upright stability. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on 2012, 3209-3216. http://dx.doi.org/10.1109/iros.2012.6386151

[15] Doman D, Oppenheimer M and Sigthorsson D. Dynamics and control of a minimally actuated biomimetic vehicle: Part i - aerodynamic model. AIAA Guidance Navigation Control Conference 2009. http://dx.doi.org/10.2514/6.2009-6160

[16] ——. Dynamics and control of a minimally actuated biomimetic vehicle: Part ii – control. AIAA Guidance Navigation Control Conference 2009.

[17] Perseghetti BM, Roll JA and Gallagher JC. Robot Intelligence Technology and Applications 2: Results from the 2nd International Conference on Robot Intelligence Technology and Applications. Cham: Springer International Publishing ch. Design Constraints of a Minimally Actuated Four Bar Linkage Flapping-Wing Micro Air Vehicle 2014; 545-555.

[18] Boddhu SK, Botha HV, Perseghetti BM and Gallagher JC. Robot Intelligence Technology and Applications 2: Results from the 2nd International Conference on Robot Intelligence

Technology and Applications. Cham: Springer International Publishing ch. Improved Control System for Analyzing and Validating Motion Controllers for Flapping Wing Vehicles 2014; 557-567.

[19] Botha HV, Boddhu SK, McCurdy HB, Gallagher JC, Matson ET and Kim Y. Robot Intelligence Technology and Applications 3: Results from the 3rd International Conference on Robot Intelligence Technology and Applications. Cham: Springer International Publishing, 2015, ch. A Research Platform for Flapping Wing Micro Air Vehicle Control Study 2015; 135-150.

[20] Gallagher J, Doman D and Oppenheimer M. The technology of the gaps: An evolvable hardware synthesized oscillator for the control of a flapping-wing micro air vehicle. Evolutionary Computation, IEEE Transactions on 2012; 16(6): 753-768. http://dx.doi.org/10.1109/TEVC.2012.2186816

[21] Greenwood G, Podhradsky M. Gallagher J and Matson E. A multi- agent system for autonomous adaptive control of a flapping-wing micro air vehicle. In Computational Intelligence, 2015 IEEE Symposium Series on 2015; 1073-1080. http://dx.doi.org/10.1109/SSCI.2015.154

[22] Greenwood G and Tyrrell AM. Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems. Wiley-IEEE Press 2006. http://dx.doi.org/10.1002/0470049715

[23] Karaboga D and Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization 2007; 39(3): 459-471. [Online]. Available: http://dx.doi.org/10.1007/s10898-007-9149-x

[24] Uosaki K and Hatanaka T. Evolution strategies based particle filters for fault detection. In Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on 2007; 58-65. http://dx.doi.org/10.1109/ciisp.2007.369294

Received on 27-04-2016 Accepted on 02-07-2016 Published on 31-07-2016

DOI: http://dx.doi.org/10.15377/2409-9694.2016.03.01.2