BioMAV: bio-inspired intelligence for autonomous flight

Paul K. Gerke[a], Jurriaan Langevoort[a], Sjoerd Lagarde[a], Laurie Bax[a], Tijl Grootswagers[a], Robert-Jan Drenth[a], Vincent Slieker[a], Louis Vuurpijl[a,*], Pim Haselager[a,b], Ida Sprinkhuizen-Kuyper[a,b], Martijn van Otterlo[a] and Guido de Croon[c]
[a] Department of Artificial Intelligence, Radboud University Nijmegen
[b] Radboud University Nijmegen, Donders Institute for Brain, Cognition and Behaviour
[c] European Space Agency, Noordwijk, The Netherlands

ABSTRACT

This paper aims to contribute to research on biologically inspired micro air vehicles in two ways: (i) it explores a novel repertoire of behavioral modules which can be controlled through finite state machines (FSM) and (ii) elementary movement detectors (EMD) are combined with a center/surround edge detection algorithm to yield improved edge information used for object detection. Both methods will be assessed in the context of the IMAV 2011 pylon challenge.

1 INTRODUCTION

Autonomous flight of ever smaller Micro Air Vehicles (MAVs) poses a major challenge to the field of artificial intelligence. Recent successes in autonomous flight of MAVs have been obtained on quad rotor platforms, able to carry and power miniaturized laser scanners [1, 2]. The laser scanner allows the use of well-established *simultaneous localization and mapping* (SLAM) algorithms for navigation.

For reasons of energy efficiency and a greater potential for miniaturization, there is an increasing body of research on autonomous flight of MAVs using small camera equipment. Currently, there are two main approaches to this challenge. The first approach aims to apply visual SLAM to the MAV's environment, tackling both navigation and obstacle avoidance at the same time [3, 4, 5]. A disadvantage of this technique is the computational complexity of the algorithms involved, leading to a requirement of processing power unlikely to be available on light-weight MAVs in the order of a few grams. In addition, there are still problems concerning *drift* [5].

The second approach is more biologically inspired. Typically, camera images are used to calculate the optic flow field [6, 7, 8, 9, 10]. This field is then mapped to actions that allow the MAV to avoid obstacles or navigate corridors. The biological approach is computationally more efficient. However, (artificial) optic flow requires the presence of strong texture in the environment. In addition, the biological approach typically only provides solutions for obstacle avoidance, not for more difficult behavioral tasks such as navigation.

In this article, we present a novel biologically inspired approach to the autonomous flight of MAVs. It extends existing biologically inspired methods in two ways.

*contacting author: l.vuurpijl@ai.ru.nl

The first extension concerns the behavioral capability of biologically inspired MAVs. Most work on biologically inspired autonomous flight assumes the MAV to be flying straight forward, while being corrected by the optic flow in order to avoid obstacles or navigate corridors. We aim to develop a more elaborate behavioral repertoire inspired by the interaction between modules in flying insects [11, 12, 13, 14]. These modules, understood as relatively fixed, simple functional behaviors (reflexes) based on underlying dedicated neural structures and pathways, can, through their interaction, produce quite complex adaptive behavior. Different modules may work in parallel or in sequence, under the influence of regulatory sensory inputs. We were particularly inspired by the interaction between two modules, leading to the visual reflexes [15] of object fixation and expansion avoidance, together producing a relatively straight trajectory towards a salient object without hitting it. The interaction between the modules will be set by tuning the parameters governing their interaction through artificial evolution (see, e.g. [16]). The result of our research is a control structure that will represent a small step beyond reactive control structures, in virtue of its biological grounding.

The second extension involves vision, and in particular, going beyond the sole use of optic flow. Recent studies on the visual behavior and cognition of fruit flies [17] suggest that flies actively navigate in their environment on the basis of visual appearance cues as well: they prefer to move towards long vertical structures and avoid small (predator-like) structures. In [18], a visual appearance cue was used successfully to complement optic flow, allowing the flapping wing MAV DelFly II to avoid obstacles in a wider array of environments. While tests on real flies are typically done in visually simplified environments (led arrays / black-and-white flight arenas), normally flies and real MAVs have to fly in the visually complex real world. Recognizing a long vertical structure in real-world camera images is a challenging problem. Yet, evidence suggests that flying insects like honeybees use edge detection to guide their landing behavior [19] as well as for the recognition of shapes [20]. In this paper, we will introduce a computationally efficient edge detection algorithm that combines detected edges with motion information provided by elementary movement detectors (EMDs) [21], in order to identify and track behaviorally relevant objects over time.

Both extensions to existing biologically inspired methods will be tested in context of the indoor pylon challenge, one

http://www.imavs.org/

http://www.imavs.org/

of the indoor competitions at the International Micro Air Vehicle conference and competitions 2011 (IMAV 2011). Our flight platform is the Parrot AR.Drone quadricopter [22], to which, because of our biological perspective, we will refer to as BioMAV. The pylon challenge requires the MAV to fly as many 8-shapes as possible around two poles in the environment. Although not in itself a challenge faced by biological systems, it is suitable to investigate the two extensions. The behavioral complexity of flying 8-shapes is higher than that of following a corridor or avoiding obstacles, making it suitable for the first extension. Moreover, vertical structures are important affordances to flying animals, making the detection of large poles suitable for investigating the second extension.

Since our hardware is fixed, we have to adapt and fine-tune our biologically inspired algorithms for a robust autonomous flight of our Drone. To avoid too many crashes and to be able to use evolutionary algorithms we have built a simulator for the Drone. An iterative strategy of testing in simulation and on the real platform will be used to obtain the best result in the real world.

The remainder of the article is organized as follows. The simulation platform is described in Section 2. In Section 3, promising results of our first explorations of simulated behaviour of our BioMAV in the pylon challenge are presented. In Section 4, we will present our new vision module, which combines the results of center/surround edge detection with dynamic information provided by elementary motion detectors. As discussed in Section 4, our first results with this combined approach are very promising. Finally, in Section 5 we conclude and outline our next research steps.

2 PLATFORM AND SIMULATOR

In this section we will first present our physical and simulated environments.

2.1 Tests on a real platform

The Parrot AR.Drone, a model-sized helicopter with four rotors [22], utilizes an embedded linux-based operating system to automatically stabilize itself when flying. The operating system allows to control the flight of the drone using abstract control commands, like "takeoff", set the drone-pitch to let it accelerate in a certain direction, or "hover" which lets the drone automatically cancel its movement with respect to the ground. Table 1 shows a complete overview of the available commands. These commands provide an abstraction layer from physical aspects and implementation details which are required for maintaining a stable drone flight. These commands allow us to focus on the development of biologically inspired vision processing modules and high-level behavioral control layers.

2.2 Simulation and ground control

The final program that enables the drone to fly 8-shaped figures around two poles will consist of three modules (see Figure 1 for the architecture): The ground control software is

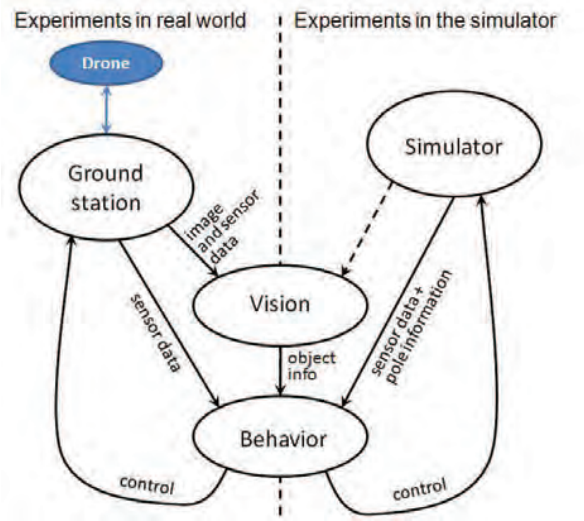

Figure 1: Architecture of our BioMAV platform

written in C++ and it directly communicates with the drone. It communicates sensory data and control commands to a Java environment. The vision module is a Java program which uses a biologically inspired algorithm to detect poles in front camera images. The positions of the detected poles are translated into relative heading and an approximated distance (see Section 4). These data are given to the third module which manages the behavior of the drone. The biologically inspired behavior module uses sensory data and the extracted data about visible poles to navigate the drone in 8-shaped figures around the visible poles (see Section 3).

The ground control software is an application programming interface that allows to send commands to the drone using Java code. Since we will not need fine-grained height control, the ground station allows to set the drone height rather than the height-speed. The other commands available for controlling the drone are modeled after the original control commands mentioned in Table 1.

We are developing the simulator in such a way that the behavior and vision modules can be developed and tested independently. In the final implementation, the vision module will provide information about poles that are visible to the drone. In the controlled world of the simulator we can calculate this information without using the vision module, thereby allowing us to test the behavior module on its own. Likewise, image data can be generated using the simulator to test the vision module. The simulator allows us to test algorithms for the drone without using the actual drone itself. The advantages of using a simulator are that the test cycles are faster and the tests inside the simulator do not endanger the available hardware. These advantages allow us, in a later step, to use the simulator as part of a program which iteratively tunes parameters of the behavior module to optimize the drone's flight behavior. The disadvantage of using a simulator is the

Command	Effect
Hover	Lets the helicopter hover over a certain point at a constant height. For this the helicopter makes use of the bottom camera to calculate the optic flow and nullify its ground speed. Furthermore, it uses its height sensor (ultrasound sonar) to maintain a stable height above the ground.
Pitch	Makes the helicopter pitch until it reaches a given angle to control forward or backward movement. The helicopter thereby maintains a stable height. If desired this command can be combined with the *yaw*, *roll*, or *lift* commands.
Yaw	Makes the helicopter yaw around its vertical axis at a given speed. The helicopter thereby maintains a stable height. If desired this command can be combined with the *pitch*, *roll*, or *lift* commands.
Roll	Makes the helicopter roll until it reaches a given angle to control sideward movement. The helicopter thereby maintains a stable height. If desired this command can be combined with the *pitch*, *yaw*, or *lift* commands.
Lift	Makes the helicopter ascent upwards or descent downwards at a given speed. If desired this command can be combined with the *pitch*, *yaw*, or *roll* commands to let the helicopter change its height during these commands.
Takeoff	Lets the helicopter start and ascent to a standard height (about 1 meter).
Land	Lets the helicopter land on the ground.
Emergency mode	Cuts the power to all helicopter motors. Only used for safety purposes.

Table 1: Available commands for controlling the Parrot AR.Drone [22]

http://www.imavs.org/

reality gap. We will revisit this problem in Section 3.

The simulator generates visual input corresponding to the images that the camera of the drone would generate. The simulator can provide in addition all other information the drone itself would produce. The drone's control commands are modeled after the commands that the ground station provides. We use OpenGL[1] to render the three-dimensional environment for the drone. An integrated rigid body simulation based on the OpenDynamicsEngine library[2] provides us with collision detection algorithms and Newtonian mechanics for simulated bodies. The reactions of the drone to the given set of control commands are specified using a script written in the scripting language Lua[3]. The simulated reactions are a simplification of the real behavior. The drone-behavior Lua script allows for easy adaption of the drone's behavior to make the drone react more realistically in future simulations. One of the avenues we are exploring is to train the parameters of a neural network or a Kalman filter to map the current state and commands to the resulting behavior of the drone. The reasons for this approach are that (a) the system identification of the individual components of the Parrot platform is difficult with the standard firmware, and that (b) some physical parameters may be hard to identify in the first place (like the effects of the bending of the rotor blades during flight maneuvres).

Using a second simulator Lua script, the simulated world can be modified. In this script, the layout of the simulated world can be determined by creating walls, placing poles, or setting the initial position of the drone. The future goal is to expose a Lua interface to Java so that arbitrary Lua scripts can be executed from Java. Thereby the Lua scripts can control the simulated world or the drone inside it. We plan to release the simulator as an Open Source project after the competition.

[1] `http://www.opengl.org`
[2] `http://www.ode.org/`
[3] `http://www.lua.org`

3 BEHAVIOR

Many behaviors can be accomplished by means of feedforward, reactive controllers [23, 24]. Indeed, in the field of evolutionary robotics, most studies focus on the evolution of feedforward neural networks for successful control of the robot [25, 26]. When moving towards more complex behaviors, recurrent neural networks of various kinds are used [24, 27, 28, 29, 30, 31]. These systems are hard to analyze and to correct in cases where the displayed behavior differs from the behavior in simulations or when the performance in operational settings decreases compared to the training conditions. Our ongoing research explores Finite State Machines (FSM) [32] as an alternative framework for the design of complex behaviors. FSM are mathematically well-understood and have the additional advantage that they are easier to understand by a human designer. FSMs can combine different behavioral modules to achieve the required behavior. In the current article, the IMAV2011 pylon challenge is used as a vehicle to introduce our approach of exploring different simulated behaviors implemented through FSMs.

An abstract FSM schema as used in the BioMAV drone, containing *states* and *transitions*, is depicted in Figure 2. Rounded boxes represent states and arrows represent state transitions. A state describes, at a general level, the type of behavior displayed by the drone. Transitions describe the conditions that need to be met in order to move from one state to another. Only one state can be active at a time.

http://www.imavs.org/

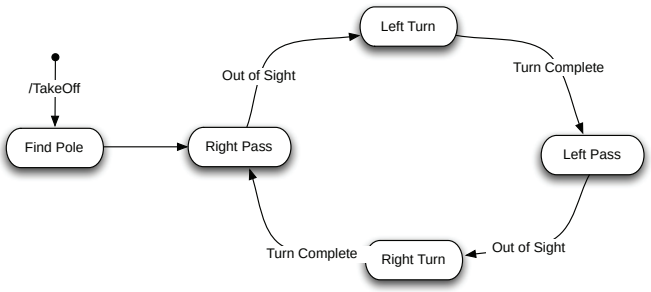

Figure 2: FSM schema for 8-shaped flight.

After take off, the drone enters the 'Find Pole' state. Upon finding a pole, the transition to the 'Right Pass' state is made. In this state the drone flies towards the right of a pole. When the pole is out of sight, the 'Left Turn' state will become active. Here the drone will make a left turn around the pole, completing the first half of the 8-shaped figure. There is no final state; the drone will continue to fly 8-shaped figures.

Each state of the FSM executes three 'sub-behaviors': *entry behavior*, *state behavior*, and *exit behavior*. Entering and exiting a state results in executing, respectively, the corresponding entry or exit behaviors. In the current implementation no entry behaviors are specified. All states have an exit behavior that makes the drone hover at its current location as to stabilize the vision input. The state behavior makes the drone execute a specific sequence of commands, e.g. processing of sensory data. A state can have multiple transitions associated with it, each described by its own set of criteria (triggers, guards) that need to be met for the transition to become active. Each transition can also be equipped with a behavior that will be executed when the transition is active. In the case of the 'Out of sight' transition, for example, the transition behavior is that the drone will fly a bit further forward in order to make a safe turn.

3.1 First FSM flight simulations: timers and gyroscope

The behaviors of the first version of the FSM were based on timers. Timing parameters determined the amount of turn time to fly around a pole and the time to fly in a straight line. Our simulation results without noise factors such as drift and vision failures showed the feasibility of our approach. To test the performance of the drone in a more realistic setting, Gaussian noise was added to the horizontal and vertical speeds ($\mathcal{N}(0, 0.10)$ in m/s) and to the initial position ($\mathcal{N}(0, 0.7)$) of the drone.

Figure 3 shows a simulation where the drone flies in an unstable, jagged trajectory. The first part of the 8-shaped figure is completed without problems. On the second turn, however, the drone loses its path and misses the next pole.

To explore our approach in more realistic noisy conditions, the use of simulated gyroscope readings was evaluated as an alternative method. These readings provide feedback on the turning angle when performing a "Left Turn" or "Right Turn". The results as displayed in Figure 4 show that using gyroscopes is a more robust method than using timers.

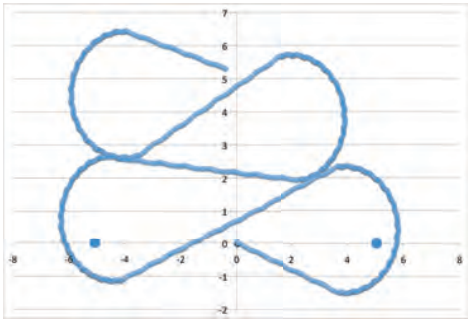

Figure 3: Simulated trajectory of the drone, using additive Gaussian noise on the horizontal/vertical speeds and the starting position of the drone. Behavioral modules are based on timers. The drone starts at position (0,0) and the poles are located at (-5,0) and (5,0).

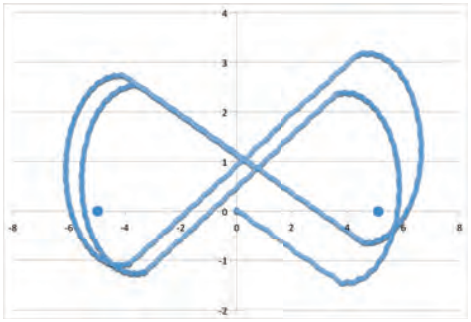

Figure 4: Simulated trajectory of the drone, using additive Gaussian noise on the horizontal/vertical speeds and the starting position of the drone. Control of behavior is based on gyroscope readings.

3.2 FSM flight: closed-loop control with simulated vision

A second, more complex, FSM was used to incorporate simulated vision (see Figure 5).

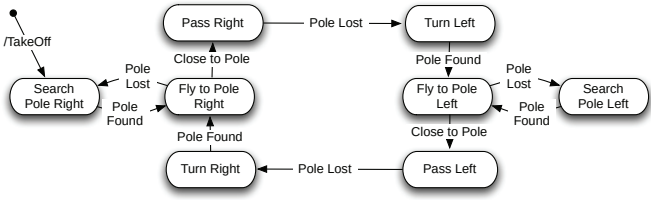

Figure 5: The structure of the improved FSM.

The simulator acts as a stand-in vision module, by providing (i) the location of and (ii) the distance to the pole with random noise and frame-loss. The position of the pylon was used to determine the angle of approach relative to the drone. Based on this information, the new FSM controls three transitions, which each use a different threshold

http://www.imavs.org/

to determine whether one of these conditions hold: "Pole Found", "Close To Pole" and "Pole Lost". Each setting of these three thresholds results in a certain flight behavior (trajectory) which can be assessed quantitatively by comparing the generated trajectory to the pre-determined 8-shaped figures. The required thresholds are set through an evolutionary algorithm (EA) which uses these quantitative measures as fitness value. Again, several noise factors are added to explore the robustness of the FSM when transferred to the real world.

In Figure 6 the new flight paths of the drone are depicted. Because the behavior is now more sensor driven, the drone is able to correct its path and repeatedly fly in an 8-figure around the poles. Even when the drone lost the pole, it is able to correct its path by including an extra loop in its flight.

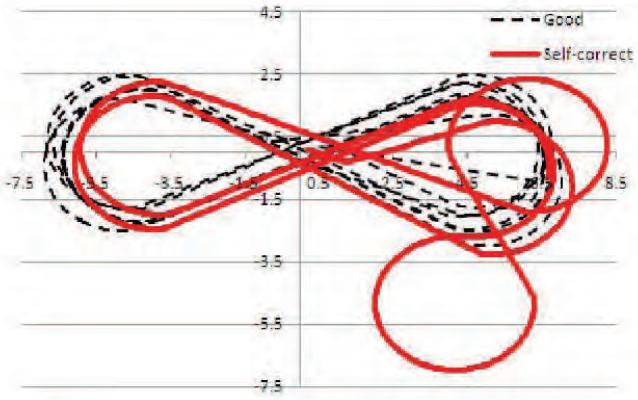

Figure 6: Simulated trajectory of the drone, using the sensory driven FSM. The trajectories are from two separate simulations: one (black, dashed) in which everything went smoothly and one (red) in which the drone corrected its path by flying an extra loop.

It can be concluded that using FSMs for controlling the drone's behavior provides us more insights in which parameters are important for executing a certain behavior. Moreover, the use of a genetic algorithm for searching for suitable parameter settings shows promising results.

3.3 From simulated experiments to real flight

The fitness function used for the EA mentioned above employs a quantitative measure which is based on the difference between the required trajectory and observed *simulated* behavior. Many algorithms for trajectory matching can be employed for this purpose, e.g., borrowed from the field of gesture recognition [33]. Using the high-resolution tracking equipment from our virtual reality lab [4], the real flight behaviors of our BioMav can be recorded. We have created a setup that allows for an evaluation whether behaviors of the drone in simulations can be compared to trajectories recorded during real flight. Please consider Figure 7 for an example.

[4] See http://www.rivierlab.nl

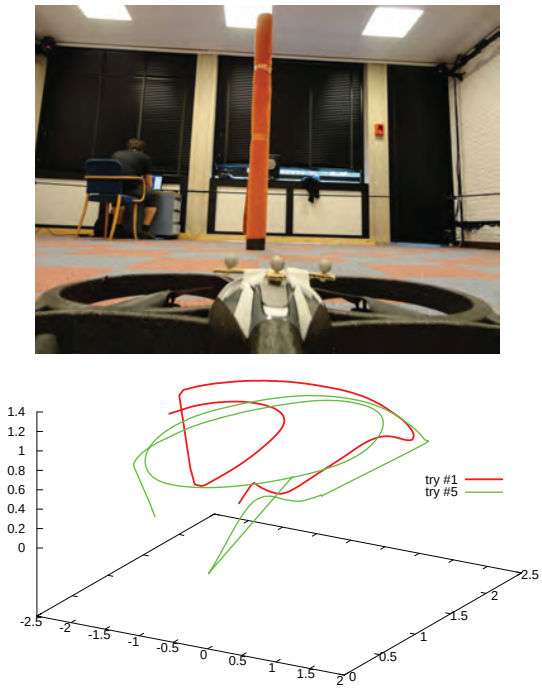

Figure 7: Picture of the drone with rigid body attached (top picture). The bottom picture shows the 3-dimensional flight patterns of two short tries in which the drone attempted to fly in circles.

To record the three-dimensional orientation and position of our BioMAV, a light-weight rigid body is attached on top of the drone. This has no significant impact on the drone's flying behavior and can be tracked by the tracking equipment of the VR lab. The bottom picture from Figure 7 shows the recorded flight trajectories of the drone while trying to complete circles. The drone is controlled via the ground station software which was described in Section 2.2. We are currently using this set up to improve our FSM models for the IMAV2011 pylon challenge.

4 VISION

It is well known that natural vision systems fuse different information sources to obtain more robust and accurate estimates concerning objects in the environment. In our ongoing research, appearance and motion cues are explored for extracting information about the color, location, movement and shape of objects in the environment [34, 35, 36]. In the IMAV2011 pylon challenge, the main task for vision is to detect poles and return the distance and angle between pylon and drone. Traditionally, in computer vision, this task is achieved by image segmentation, which transforms a raw image representation in sets of detected objects. Many approaches for image segmentation have been proposed, which all use representations of texture, color, and shape to some extent. Since the quality of the contours of detected objects relies heavily on the detection of relevant boundaries, or edges, in the image, edge detection is a prominent step in image segmentation. We propose to combine edge detection with

http://www.imavs.org/

motion information provided by elementary movement detectors (EMDs) [21]. EMDs use spatially separated inputs with a certain delay in time to produce a measure for the motion in a specific direction. The use of EMDs is especially useful in UAVs, since the flying task induces temporal and motion effects (which are known to cause the detection of spurious edges), ensuring that there is always activation from the EMDs. During the competition, we will apply an object recognition method on the resulting edge images to detect the size and location of the pole in each image. We will consider available techniques such as the generalized Hough transform for this purpose. The resulting information will subsequently be used by the FSM described in Section 3 to generate appropriate control commands based on the angle and distance to detected poles.

4.1 EMD-based filtering of spurious edges

To generate motion information, we have used the EMD implementation of Zhang et al. [37]. This implementation is less dependent on differences in contrast and color. The idea behind the use of EMDs to improve the image segmentation, is that borders of relevant objects will produce higher EMD responses. Rotational movements of the drone will lead to edge enhancements by the EMDs. Translational movements of the drone will lead to larger apparent motion of the objects closer to the drone, which are typically more relevant. This holds in particular for a flying platform that is constantly moving. The constant self motion will provide a steady and reliable source of information about the objects in the scene.

Although very useful, the EMDs rarely give a complete picture, because they rely on differences in contrast to activate. Therefore the EMDs alone are not enough to successfully segment the image and to provide the drone with sufficient information about the world. However, EMDs do provide an abundance of additional information to the more traditional approach.

Next to the resulting EMD information sketched above, our algorithm uses the edges (as provided by a simple center/surround edge detection algorithm) as a basis on which to work. Edge detection assumes a difference in contrast and color, where the boundaries of objects can in general be distinguished from the background. However, edges may also be detected in image samples containing rough textures or areas with uneven terrain. These edges are known as spurious edges. The EMDs are less dependent on contrast differences for their activity and depend more on movement. As can be observed in the bottom-left picture from Figure 8, this produces clean edges of objects in the direction of the movement. By combining the edge and EMD information, noise can be removed from the edge information, producing a clean image of the relevant edges.

4.2 Results of combined edge detection

Figure 8 shows a natural scene (data recorded in the virtual reality lab), the EMD information of this scene and both the edge information and the combined EMD and edge information. As can be observed, the use of EMD information can be used to filter out much of the noisy edges. Large planes such as floors, walls and ceilings are prone to produce noisy data. Yet if they are far away, they have very little observed movement, which allows us to confidently remove many superfluous edges. If they are close by, such as the floor can be, fine texture is typically smoothed out by motion blur, again leading to low EMD activity.

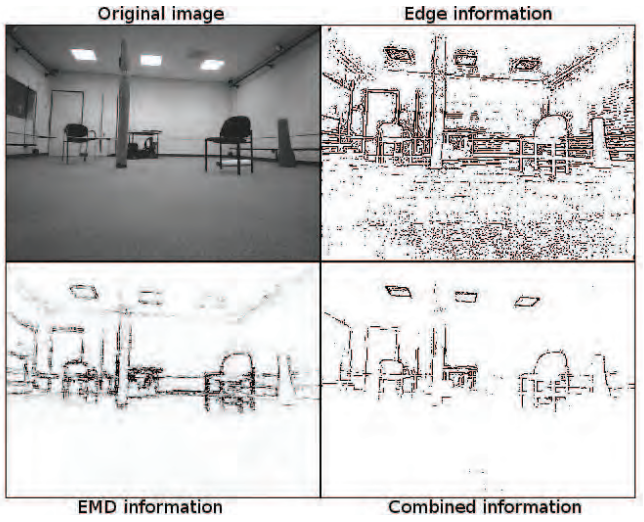

Figure 8: Example of using EMD for removing spurious edges.

To quantify the improvement of this combined edge detection method, the difference between edge detection with and without the use of the EMD information is computed for each pair of pixels in both edge images. The two methods are compared to manually segmented images of realistic input captured by the frontal camera of the drone in our VR lab. Contours in the segmented groundtruth images are manually produced and subsequently blurred by a small Gaussian filter to allow for small variations in the location of edges. In this way 10 images have been processed. Due to the Gaussian brush width, a relatively large number of pixels was marked as "edge": on average 24% of the pixels are labeled as positive. Table 2 shows the confusion matrices of both algorithms averaged over the images (expressed in as ratios).

	Plain edge detection		Combined with EMD	
	true	false	true	false
positive	0.96	0.86	0.67	0.30
negative	0.14	0.04	0.70	0.33

Table 2: Confusion matrices of plain edge detection and of its combination with EMD.

When considering Table 2, the effect of filtering spurious edges becomes apparent. The combined technique improves

http://www.imavs.org/

both the false positives (where the edge detection erroneously detects edges) and the true negatives (where the edge detection correctly detects no edge). However, also the number of true positive edges decreases because of the filtering. In Figure 8 this effect is in particular visible at the top of the pole. While plain edge detection correctly classifies 34% of all pixels, the combination with the EMD leads to correct classification of 69% of the pixels. Our current efforts are targeted at the implementation of suitable object detection techniques which operate on the resulting improved edge images. Within the context of the pylon challenge, we will further explore the trade-off between removing spurious edges and losing parts of the target object.

5 CONCLUSIONS

In this paper, we have introduced our ongoing research on biologically inspired MAVs (BioMAVs). Two new approaches have been discussed. For the FSM-based control of behavioral modules, the feasibility of our approach was assessed and promising results have been obtained. Furthermore, the novel edge detection algorithm which exploits motion information provided by EMDs, has shown to yield much cleaner edge images than when using traditional edge detection techniques. Our main conclusions and next steps are described below.

The FSM controller structure was inspired by the presence of behavioral modules in natural systems. As outlined in Section 3, finite state machines offer a suitable framework for implementing such modules for controlling different behaviors. Results from various simulations show that the explored FSM architectures provide a mechanism for the execution of the pylon challenge. Furthermore, using this approach, the designer is able to evaluate different settings and locate and modify defects. Three experiments were presented in this paper. All experiments were run in the simulation environment which is presented in Section 2. Timer information was used as a first naive approach, which mainly assessed the feasibility of our approach. Simulations were run on gyroscope information in several noise settings, resulting in successful 8-shaped flight patterns. Furthermore, we developed an evolutionary algorithm to optimize the drone control on the basis of simulated vision. Based on closed-loop control, using the perceived distance and angle of approach to the drone, the resulting FSM was able to exhibit the required 8-flight behavior. It was demonstrated that the drone is able to correct its flight when missing a pole.

For the required object detection and tracking, we are still in the process of finalizing our vision module. We have argued that the detection of relevant edges is important in biologically-inspired vision. The results of the vision module indicate that edge detection is helped by combining motion cues with information provided by edge detection. This improved edge detection will most probably lead to fewer spurious segments in the subsequent image segmentation and object detection processing steps.

Within the context of our BioMAV project, we will gradually shift from performing simulations to real flight control. Through controlled experimental studies in our virtual reality (VR) lab, the gap between simulation and reality can be explored. As we have shown in Section 3.3, the VR lab enables us to employ high-precision tracking equipment to record actual flying behavior of our BioMAV. The results of these experiments are expected to yield two important contributions to our work. On the short term, we will be able to systematically calibrate the drone behavior according to the environmental and task conditions of the IMAV2011 indoor pylon challenge. On the longer term, we hope to improve our understanding of issues that cause the reality gap, by establishing a proper calibration between simulation algorithms and parameters, corresponding simulated behavior, and the real flight capabilities of our BioMAV. For an ongoing report of our progress and achievements, the reader is invited to visit our BioMAV website at `http://www.biomav.nl`.

REFERENCES

[1] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *European Micro Air Vehicle conf. and competitions, EMAV 2009*, The Netherlands, 2009.

[2] S. Shen, N. Michael, and V. Kumar. 3D estimation and control for autonomous flight with constrained computation. In *ICRA*, 2011. In press.

[3] A. Davison and D. Murray. Simultaneous localisation and map-building using active vision. *IEEE PAMI*, 2002.

[4] K. Celik, S.J. Chung, and A. Somani. Mono-vision corner slam for indoor navigation. In *(EIT 2008)*, pages 343–348, 2008.

[5] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *IEEE Int. conf. on Robotics and Automation*, 2010.

[6] N. Franceschini, J.M. Pichon, C. Blanes, and J.M. Brady. From insect vision to robot vision. *Philosophical Trans.: Biological Sciences*, 337(1281):283–294, 1992.

[7] F. Iida. Goal-directed navigation of an autonomous flying robot using biologically inspired cheap vision. In *(ISR 2001)*, 2001.

[8] A. Beyeler, J-C. Zufferey, and D. Floreano. 3d vision-based navigation for indoor microflyers. In *ICRA 2007, Italy*, pages 1336–1341, 2007.

[9] F. Ruffier and N.H. Franceschini. Aerial robot piloted in steep relief by optic flow sensors. In *(IROS 2008)*, pages 1266–1273, 2008.

[10] A.M. Hyslop and J.S. Humbert. Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow. *AIAA Guidance, Control, and Dynamics*, pages 147–159, 2010.

[11] E.M. Leise. Modular construction of nervous systems: a basic principle of design for invertebrates and vertebrates. *Brain Research Reviews*, 15:1–23, 1990.

[12] M.A. Frye and M.H. Dickinson. Fly flight : A model for the neural control of complex behavior. *Neuron*, 32:385–388, 2001.

[13] D. Floreano, J.-C. Zufferey, M.V. Srinivasan, and C. Ellington. *Flying insects and robots*. Springer, Berlin, 2011.

[14] D. Floreano and J.-C. Zufferey. Insect vision: A few tricks to regulate flight altitude. *Current Biology*, 20(19):R847–R849, 2010.

[15] M.B. Reiser and M.H. Dickinson. Drosophila fly straight by fixating objects in the face of expanding optic flow. *The J. of Exp. Biology*, 213:1771–1781, 2010.

[16] P. Petrovic. Evolving behavior coordination for mobile robots using distributed finite-state automata. In H. Iba, editor, *Frontiers in evolutionary robotics.* Viena: I-Tech Education and Publishing, 2008.

[17] G. Maimon, A.D. Straw, and M.H. Dickinson. A simple vision-based algorithm for decision making in flying drosophila. *Current Biology*, 18(6):464–470, 2008.

[18] G.C.H.E. de Croon, E. de Weerdt, C. de Wagter, B.D.W. Remes, and R. Ruijsink. The appearance variation cue for obstacle avoidance. In *ROBIO*, 2010.

[19] M. Lehrer, . M.V. Srinivasan, and S.W. Zhang. Visual edge detection in the honeybee and its chromatic properties. *Proc. of the Royal Society B: Biological Sciences*, 238(1293):321–330, 1990.

[20] Natalie Hempel de Ibarra and Martin Giurfa. Discrimination of closed coloured shapes by honeybees requires only contrast to the long wavelength receptor type. *Animal Behaviour*, 66(5):903 – 910, 2003.

[21] W. Reichardt. Evaluation of optical motion information by movement detectors. *J. Comp. Phys. A*, 161:533–547, 1987.

[22] S. Piskorski. *AR.Drone Developer Guide SDK1.5*, 2010.

[23] S. Nolfi. Power and limits of reactive agents. *Neurocomputing*, 42:119–145, 2002.

[24] M. van Dartel, I. Sprinkhuizen-Kuyper, E. Postma, and J. van den Herik. Reactive agents and perceptual ambiguity. *Adaptive Behavior*, 13(3):227–242, 2005.

[25] F. Ruini and A. Cangelosi. Extending the evolutionary robotics approach to flying machines: An application to mav teams. *Neural Networks*, 22:812–821, 2009.

[26] N. Bredeche, E. Haasdijk, and A.E. Eiben. On-line, on-board evolution of robot controllers. In *Proc. of the 9th int. conf. on Artificial evolution*, pages 110–121, 2010.

[27] K. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proc. of the Genetic and Evolutionary Computation Conf.*, 2002.

[28] C.F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. on Systems, Man, and Cybernetics*, 34(2):997–1006, 2004.

[29] Stefano Nolfi and Davide Marocco. Evolving robots able to integrate sensory-motor information over time. *Theory in Biosciences*, 120:287–310, 2001.

[30] R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[31] F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Trans. on Neural Networks*, 12(6):1333–1340, 2001.

[32] V. Sklyarov. Hierarchical finite state machines and their use for digital control. *IEEE Trans. on Very Large Scale Integration Systems*, 7:222–228, 1999.

[33] D. Willems, R. Niels, M. van Gerven, and L. Vuurpijl. Iconic and multi-stroke gesture recognition. *Pattern Recognition*, 42(12):3303–3312, 2009.

[34] Reid R. Harrison. *An Analog VLSI Motion Sensor Based on the Fly Visual System*. PhD thesis, California Institute of Technology, Pasadena, California, 2000.

[35] Reid R. Harrison and Christof Koch. A robust analog vlsi motion sensor based on the visual system of the fly. *Autonomous Robotics*, 7:211–224, 1999.

[36] L.F. Tammero. The influence of visual landscape on the free flight behavior of the fruit fly drosophila melanogaster. *J. of Exp. Biology*, 205:327–343, 2002.

[37] Tianguang Zhang, Haiyan Wu, Alexanderr Borst, Kolja Kühnlenz, and Martin Buss. An FPGA implementation of insect-inspired motion detector for high-speed vision systems. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 335–340, 2008.

http://www.imavs.org/