Bioinspired Vision-only UAV Attitude Rate Estimation using Machine Learning*

M. Mérida-Floriano[1], F. Caballero[2], D. García-Morales [3], F. Casares [3] and L. Merino[1]

***Abstract*— This paper presents a bioinspired system for attitude rate estimation using visual sensors for aerial vehicles. The sensorial system consists of three small low-resolution cameras (10x8 pixels), and is based on insect ocelli, a set of three simple eyes related to flight stabilization. Most previous approaches inspired by the ocellar system use model-based techniques and consider different assumptions, like known light source direction. Here, a learning approach is employed, using Artificial Neural Networks, in which the system is trained to recover the angular rates in different illumination scenarios with unknown light source direction. We present a study using real data in an indoor setting, in which we evaluate different network architectures and inputs.**

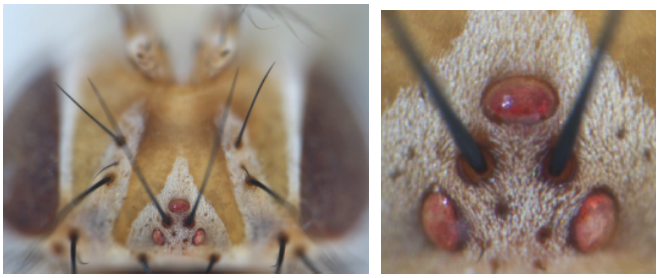

Fig. 1. Left: Head of the *Cortunotum helvum*. The two compound eyes are at the sides of the picture, while the three ocellar sensors can be seen at the bottom, center. Right: close-up view of the three ocelli.

I. Introduction

In the last decade impressive demonstrations [1], [2] have shown the potential of Micro-Aerial Vehicles (MAVs, aerial vehicles between 0.1 and 0.5 meters and 0.1-0.5 kg. in mass). They may open up a new plethora of applications for aerial robotics, both in indoor and outdoors scenarios.

The main basic functionalities that are required for the application of aerial autonomous platforms are low-level stabilization and waypoint navigation, including obstacle avoidance (sense and avoid). Most of the current outdoor solutions rely on GPS receivers and inertial measurement units (IMUs) for navigation and control. Sense-and-avoid can be implemented using vision sensors, radar or LIDAR-based rangefinders, although there are still not mature solutions for obstacle avoidance. However, many of the current solutions can only be applied to large unmanned aerial systems. Most current systems for MAVs rely on external sensing and computing, using a VICON or similar motion capture system and carrying out most of the processing externally. The very limited payload of the small/micro vehicles imposes constraints on the kind of sensors and processing power that can be carried on board, and thus, limits the autonomous capabilities of these vehicles [1].

Regarding sensing, vision systems offer a promising alternative, as cameras are low-power passive sensors and can be made small. Vision-based procedures have been proposed for odometry [3], [4], [5], localization [6], [7], [8], mapping [9] and navigation [10]. All these vision systems are computationally demanding and require large onboard processing power, though.

On the other hand, it is very impressive the maneuverability that flying insects like flies can achieve with their very small payloads. Gaze fixation and flight stabilization are functions contributed by two types of visual organs (see Fig.1): the compound eyes and the ocelli (and in the case of dipterans flies- also by small gyroscopes, the halteres). Compound eyes are large, formed by clustered arrays of unit eyes and inform the insect mostly about motion, size and color and light polarization vector. Their processing neuropiles are large and, in terms of neuronal diversity, comparable to those in the mammalian retina. Conversely, the three ocelli found on the forehead of most insects are small, structurally simple camera type eyes. Their large lens makes them extremely sensitive light sensors while, by focusing beyond the retina, blur the image. Compared to the compound eyes, ocelli, with their large convergence and direct connection with motor centers, are capable of very quick visual processing to trigger swift stabilization reflexes [11], [12].

The remarkable maneuverability of flying insects is the reason why several authors have looked for bio-inspired solutions for the development of new sensors and/or actuators for micro aerial vehicles and even pico-aerial vehicles, like [13], [14], [15]. However, most of these works have been devoted to the development of artificial compound-like eyes.

The insect ocelli system offers an interesting alternative. And several approaches inspired on the ocellar system have been also proposed in the literature. In [16], the authors present a device based on the ocellar system using 8 photodiode pairs tuned for the ultraviolet and green parts of the light spectrum. The device is used for flight stabilization

*This work was supported by MINECO (Spain) grant OCELLIMAV (TEC-61708-EXP)

[1]M. Mérida-Floriano and L. Merino are with School of Engineering, Universidad Pablo de Olavide, Seville, Spain {mmerflo,lmercab} at upo.es

[2]F. Caballero is with the Department of Systems Engineering and Automation, University of Seville, Spain fcaballero at us.es

[3]D. García-Morales and F. Casares are with Department of Gene Regulation and Morphogenesis, CABD, CSIC and Universidad Pablo de Olavide, Seville, Spain {dgarmor,fcasfer} at upo.es

with respect to the horizon. The outputs of the photodiodes are used to obtain a reference signal for the stabilization. In [17], the ocellar sensors are modeled as sensors providing an estimation of time derivatives of scalar luminance values. Then, system identification techniques are used to derive a linear relation between the ocelli inputs and robot states. They conclude that there is a relation between the ocellar input and roll and pitch angular rates, as well as heave rate. This is then used to develop an analog angular rate sensor based on photodiodes. A similar sensing approach is followed in [18], where an ocelli-inspired flight stabilization system has been implemented on a bee-sized flying robot. The addition of a torque controller based on a proportional feedback to the estimated angular velocity has sufficed to stabilize the upright orientation of the system.

The previous ocelli-inspired approaches disregard the spatial information of the ocelli simple eyes, and in some cases [18] some assumptions are needed with respect to the light source direction. On the contrary, in [19], linear receptive fields are optimized from data to obtain the relation between the sensorial input of simulated insect-like eyes and attitude angles. Here we follow a similar approach, but estimating angular rates instead, and learning neural network maps from the emulated ocellar inputs using data. In particular, we leverage the capabilities of Deep Neural Networks (DNNs), which have been shown to be very successful for visual information processing and has been even used for applications of robot estimation control [20], [21].

The paper is organized as follows. Next section briefly summarizes the ocelli morphology in Drosophila and describes a hardware setup based on cameras inspired by this. Next, we describe the artificial network architectures considered in Section III. Section IV evaluates the different architectures using datasets obtained in indoor environments. The paper ends with a discussion and lines for future work.

II. Hardware Setup

A hardware setup has been conceived to emulate the Drosphila's Ocelli as a computer vision sensor. It is shown in Fig.2. As commented above, the three ocelli found on the forehead of most insects (see Fig.1) are small, structurally simple camera type eyes. Their large lens makes them extremely sensitive light sensors while, by focusing beyond the retina, blur the image. There are two lateral ocelli and one median ocellus, oriented around 30-40 over the horizontal. In Drosophila, our insect model, each ocellus comprises 40 photoreceptors (PRs).

The setup consists of three small cameras spatially distributed according to the geometry of Drosphila's Ocelli. To emulate the biological system, the images are downsampled so that 80-pixel images are gathered per camera. The optics were also chosen to have a wide field of view in each camera as in the biological sensor.

An inertial measurement unit (IMU) has been attached to the camera setup in order to have a ground-truth of angle rate. The sensor integrates a three-axis gyroscope, a three-axis accelerometer and a tree-axis magnetometer to obtain a smooth and bias-free angle rate. These rates will be used in Section IV to train the different neural network approaches proposed and also as ground-truth for validation.

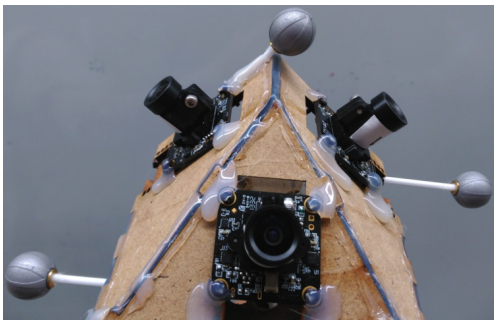

Fig. 2. Hardware setup. Three cameras are disposed at angles. An IMU is used to gather attitude ground truth data. Also, visual markers for a motion capture system can be seen. These are used for positioning ground truth data, not used in this paper.

The camera setup has been also equipped with a set of passive infrared markers. These markers are used by a motion tracking system to estimate the 6DoF position of the camera system to double check the IMU orientation. The camera position information will be also used in future work as ground-truth for linear velocity estimation of the cameras, but it is not used in this research work.

All the previous information have been captured and recorded using Robot Operating System (ROS) [22]. Thus, for every image triplet (left, right and front cameras in Fig.2) we have its associated position, orientation and rotation rate.

Finally, it is worth to mention that the camera frame rate is fixed at 30Hz. While this is not a major problem for smooth rotations, this frame rate limits the maximum rotation rate the cameras can sense because the proposed models are based on the temporal analysis of the images. However, this limitation does not invalidate the approach itself because faster low resolution cameras can be easily found in the market.

III. Learning to estimate angular rates using Artificial Neural Networks

In this section we describe the method to estimate the angular rates from the images captured by the system described above. We focus on angular rates as they can be used for flight stabilization. Also, previous works like [17], [18] point to the relation between ocellar inputs and angular rates.

Differently from those works, we follow here a model-free approach. We apply a learning approach, in which we estimate a mapping between the image inputs and the angular rate outputs. In particular, we employ artificial neural networks (ANNs). We examine two different kinds of artificial neural networks: a multi-layer perceptron network (henceforth MLP) and a deep neural network (DNN), including convolutional neural network layers (CNNs) with some fully-connected layers.

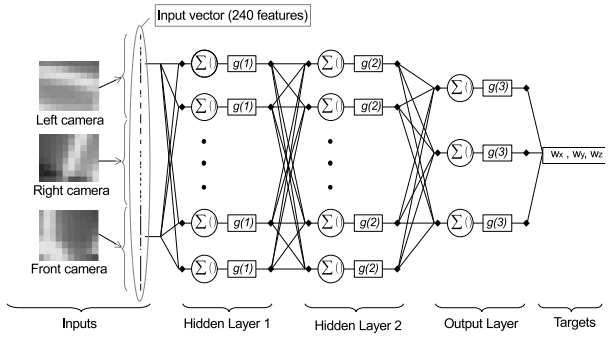

Fig. 3. Proposed multi-layer perceptron. The input is a vector with the derivatives of the pixels values for the three images (240 values). The first and second hidden layers have *ReLU* as activation function, g(1) and g(2). The output layer has 3 units that computes the three angular velocities with a g(3) = linear activation function.

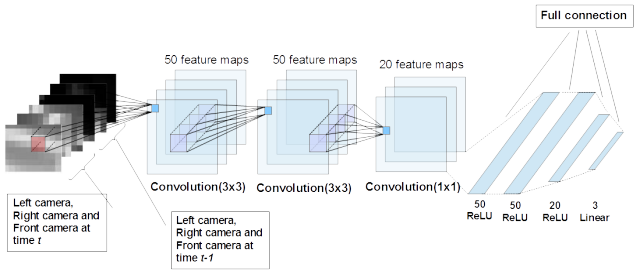

Fig. 4. Convolutional Neural Network with one input and raw pixel values. The first three channels correspond to the left camera, right camera and frontal camera images at time *t* respectively; the last three channels are the three images at time *t-1*.

A. Inputs and targets

The system of Section II provides gray-scale images with a resolution of 320x240 pixels, one image per camera; that is, we have three images at time *t*. As commented, the images are downscaled and blurred through bilinear downsampling to approximate the vision of *Drosophila* through the Ocelli. We downsample the images through bilinear filtering 5 scales, obtaining thus 3 images of 10x8 pixels dimensions (80 PRs).

These images are the inputs to the network. We will analyze different input structures, depending on how these inputs are organized and processed. These inputs structures are classified as:

- Networks in which the inputs are directly the pixel values of the downsampled images (we denote this structure as `pixels`), or the derivative of the values of the pixels (we denote this structure as `derivatives`)
- Networks in which the three images are processed by different layers and then combined at a later stage (denoted `3 inputs` networks) and networks in which the three images are processed jointly by the network (`1 input` networks)

The time derivative of the pixels is approximated by the difference between the value of the pixel at time *t* and time *t-1*.

The objective of our neural networks is to recover the angular velocities on the three axes, $\omega_x, \omega_y, \omega_z$, only with the information of the three cameras. Thus, at time t we have three inputs images which correspond to a vector with three angular velocities, our targets. These velocities have been obtained with the inertial measurement unit (IMU) located on the base of our device (see Fig. 2).

B. Neurals Networks

1) Multi-layer perceptron:
The Multi-layer perceptron is a feedforward neural network with several layers (one input layer, at least one hidden layer and one output layer), each one fully-connected to the next one. Each direct connection between units is represented by a weight (W_{ji}) that propagates the output or *activation* of unit *j* to unit *i* in the next layer. The units of a layer share the same activation function which, applied to a weighted sum of the inputs of the layer, calculates the corresponding output or activation of that layer.

The proposed network follows the structure presented in Fig.3. The information of the three cameras is assembled into a single input vector; thus, in each sample the network takes one structure of 240 values (80 elements x 3 images).

The inputs of the network are the derivatives of the pixel values of the three cameras (240 values). We discarded the use of raw pixels because in this case we would need 480 inputs (240 x 2 images), and, hence, a significant number of parameters.

2) Convolutional neural network:
The CNNs architectures developed in this paper are similar to the architecture of AlexNet network [23]. In these networks we combine convolution layers with pooling layers and a fully-connected network. One of the advantages of CNNs over MLP networks working with images as inputs is the fact that convolution layers are able to extract constant features from inputs. A convolutional layer is compound of kernels or filters with a certain dimension (mostly depending on the input resolution). These filters convolve with different receptive fields on the input, going all over the image. Thus, each convolution layer unit receives the information of a small region of the previous layer with the kernel size. The result of a convolution operation is a feature map that can detect a particular spatial aspect (edges, corners, etc.). Combining several convolution layers the network is able to extract higher-order features.

The ability to extract patterns from images is the reason for analyzing the response of the network depending on the nature of the inputs we insert. By introducing the `pixels` inputs at time *t* and time *t-1* the network learns temporal and spatial features of interest.

We will train two main architectures of CNN: `1 input` architecture (see Fig.4) and `3 inputs` architecture (see

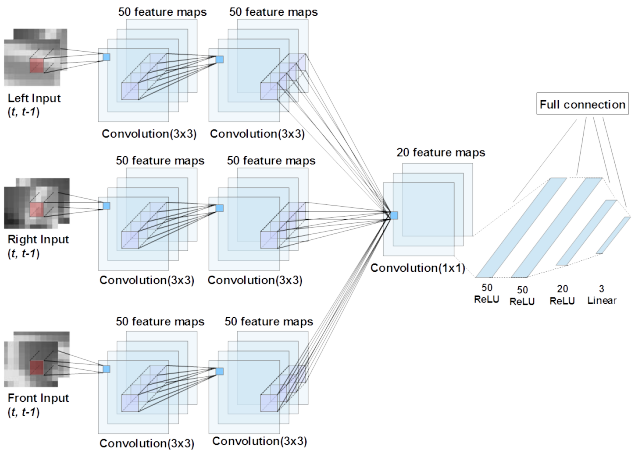

Fig. 5. Convolutional Neural Network with three inputs and raw pixels. The first channel of each input corresponds to the respective image at time *t*; the second channel is the image at time *t-1*.

Fig. 6. Some examples of images gathered by the front camera during the experiment. It can be seen how images were taken at different illumination conditions. These images are downsampled to 10x8 before input the neural networks

Fig.5). The difference between both architectures mostly resides in the way the inputs are processed. In the case of `1-input` networks taking `pixels` as inputs, the input depth is set to 6; the first three channels are the three camera images (left, right and front) at time *t* and the three last channels are the cameras at time *t-1*. Taking `derivatives` as inputs, the depth input is set to 3, where each channel is a image time derivative (the first one for the left camera, the second one for the right camera and the last one for the frontal camera).

`3-input` networks has three inputs, one for each camera (Fig.5). The input depth is 2 when inputs are `pixels`, where the first channel image corresponds to time *t* and the second channel to time *t-1*. Taking `derivatives` inputs, each input receives an image with only one channel, the image time derivative of the corresponding camera.

To study the networks' internal parameters, three CNN are trained in each case. These three variants are chosen to examine two characteristics of the network: its depth (number of layers) and its width (number of units in each layer):

a) First structure (`a1`): This net contains several layers (convolutional and fully-connected) to study the network depth (Fig.5, case with `3 inputs` architecture). For each input, two convolutional layers with 50 filters and 3x3 kernel are applied consecutively, separated by a dropout layer. The output of each processed input are merged using an additive layer and passed to a single 20 filters convolutional layer with a 1x1 kernel. The output of this layer are the inputs of a fully-connected network with 50units-50units-20units-3units.

b) Second and third structure (`a2`, `a3`): With these two structures we study the network width. Both of them share the convolutional layers' parameters: a convolution layer with 40 filters and 2x2 kernel, a 2x2 pooling and a 20 filters convolution layer with 2x2 kernel. Structure `a2` presents a fully-connected part with 100units-50units-3units, whereas structure `a3` has a less complex MLP network: 30units-15units-3units. In `3 inputs` architecture, a merge layer is applied after pooling layer.

Hence, in this paper we analyze 12 CNNs: these three structures with `1 input` and `3 inputs` architectures taking `pixels` and `derivatives` as inputs.

IV. Experiments

In this section, we evaluate the capabilities of the previous network architectures to recover the angular rates from the input images.

A. Experimental setup and datasets

A set of experiments have been conceived to both train and validate the different networks proposed in this paper. To this end, a total of 14 datasets have been recorded, all of them indoor. During the experiments, all sensor information detailed in Section II was gathered.

The three cameras were setup to have exactly the same internal parameters (gain, shutter, etc). These parameters remained the same in all the dataset experiments to avoid affecting the learning process.

Each dataset was captured at different positions, including two different rooms and one corridor. In addition, the lighting conditions were modified from one test to the other by closing/opening windows or turning on/off lights in the rooms (see Fig.6). Special care was taken to record information with the light sources at different positions and orientations, so that the system is able to generalize to every position.

The experiments are composed by pure rotations, pure translations, combined motions (rotation+translation) and steady periods. All the motions are performed with the camera system on hand. Although the objective of this research work focuses on the rotation rate estimation, linear velocities were also considered into the datasets to improve the learning process.

As a whole, the dataset is composed by 33229 samples, captured at approximately 30 Hz. Each sample integrates the images from the three cameras (inputs) and the rotation rate from the IMU (targets). In addition, one experiment composed by 2127 samples was reserved for validation purposes.

B. Training

We implement the networks using Python and the Keras library [24]. The learning algorithm used to train the networks is the Adaptive Moment Estimation (Adam), an algorithm for first-order gradient-based optimization. It has been demonstrated the effectiveness of Adam over other stochastic first-order methods on MLP and CNN [25]. We use the mean squared error (MSE) between outputs and targets as the loss function for the training phase.

Each learning process takes 200 epochs with a batch size of 100 samples. Unless otherwise specified, fully-connected layers have *ReLU* as activation function, except the output layer, which computes the outputs of the network with *linear* activation.

Finally, a 20% dropout layer [26] was included after every convolutional or fully connected layer (except for the output) to prevent the computed neural networks from overfitting.

C. Evaluation

Here we evaluate the results of each tested architecture. After the training process, we validate the model with a set of 2127 samples not used in the training (test set). This set is performed with the device on a hand doing different movements, that is the reason we expect some slight vibrations on the data set targets. Approximately, the first 400 samples the device stays in repose. Then, firstly approximate pure rotations are performed around the Z axis from 400 samples to 900 samples. After that, from 900 samples to 1200 samples, the rotation is performed around the Y axis. Rotations around the X axis come from 1200 samples to 1500 samples. The last movement, around 1500 samples to 2050 samples, is a rotation around the three axes.

We use the mean squared error (MSE) between the estimated angular rate and the known targets as metric to compare the fitness of the different trained networks. In order to quantifying the kindness of the MSE, the standard error of the mean (SEM) is calculated. Thus, we are able to compare the learning results contrasting the MSE and SEM of each network.

D. Learning results

1) MLP network:

The MLP network with derivatives presented in Section III was implemented and trained using the datasets and configurations previously introduced. Different architectures have been trained, as wider hidden layers (up to 500 neurons) and deeper structures (up to 3 hidden layers).

Table I summarizes the MSE and SEM for the different MLP architectures. The architectures are represented by the number of neurons for each layer. It can be seen that the errors for all MLP tested are significant. Adding more neurons allows reducing the errors, the best MSE obtained over the validation dataset was $0.37\frac{rad^2}{s^2}$.

TABLE I
MLP WITH DERIVATIVES TEST ERRORS

MLP Architecture	MSE (rad^2/s^2)	SEM (rad^2/s^2)
35-20-3	0.4682	0.0227
75-50-3	0.4163	0.0202
150-75-3	0.4052	0.0212
300-150-3	0.4047	0.0196
300-300-3	0.3937	0.0197
300-500-3	0.3863	0.0186
300-500-200-3	0.3733	0.0204

The MLP architecture with pixels provides even worst results, this is way it was not considered in this paper as an option.

In general, the obtained errors led the authors to think that the proposed MLP architectures are not able to properly generalize. This drawback invalidates the use of the proposed MLPs for angular rate estimation.

2) Convolutional Network:

Learning results of the twelve convolutional networks trained are presented in Table II. In general, networks with `pixels` inputs provide lower loss over the test set than networks trained with `derivatives` inputs. That means that better results are obtained letting the convolution layers extract also temporal patterns. As for the internal structure, it is shown that `a3` networks behave worse in all cases. This points that certain complexity in the network is required to obtain a good result, so that it is needed a wider fully connected network (more number of units per layer). Using `derivatives` inputs, structure `a1` stands out, whereas introducing `pixels` inputs structure `a2` gives better results. Depending on the network structure and its inputs, `3 inputs` networks and `1 input` networks obtain different results, so we can obtain no conclusion with respect to the structure of the inputs.

The worst trained network presents a `3-input` architecture with `a3` internal structure and `derivatives` as inputs, providing a MSE of $(0.291 \pm 0.013)\frac{rad^2}{s^2}$ over the validation data set.

In contrast, the best fitting convolutional network trained is the one taking `pixels` inputs, with an architecture of `3 inputs` and an internal structure `a3` (*i.e* 100units-50units-3units in its fully-connected part) which provides a train loss of $(0.0403 \pm 0.0005)\frac{rad^2}{s^2}$ and a loss over the test set of $(0.15201 \pm 0.006)\frac{rad^2}{s^2}$. Henceforth we refer to this best fitting CNN as *the network*.

A subset of the estimated and real angular rate obtained with the network over the training set in the three axes is shown in Fig.7, where the network has learnt a wide range of movements.

E. Validation

In order to examine the generalization of the network the validation data set processed by the best trained network and

TABLE II

CNN LEARNING RESULTS

Architecture	*1 Input*						*3 Inputs*					
Inputs	*Pixels*			*Derivatives*			*Pixels*			*Derivatives*		
Structure	a1	a2	a3	a1	a2	a3	a1	**a2**	a3	a1	a2	a3
Train loss: MSE (SEM) (rad^2/s^2)	0.0547 (0.0006)	0.0743 (0.0010)	0.0968 (0.0011)	0.0678 (0.0010)	0.1042 (0.0015)	0.1514 (0.0020)	0.0449 (0.0006)	**0.0403 (0.0005)**	0.0868 (0.0011)	0.0690 (0.0010)	0.1038 (0.0015)	0.1424 (0.0020)
Test loss: MSE (SEM) (rad^2/s^2)	0.199 (0.008)	0.178 (0.008)	0.208 (0.009)	0.260 (0.014)	0.280 (0.013)	0.282 (0.013)	0.176 (0.008)	**0.152 (0.006)**	0.177 (0.007)	0.190 (0.009)	0.290 (0.014)	0.291 (0.013)

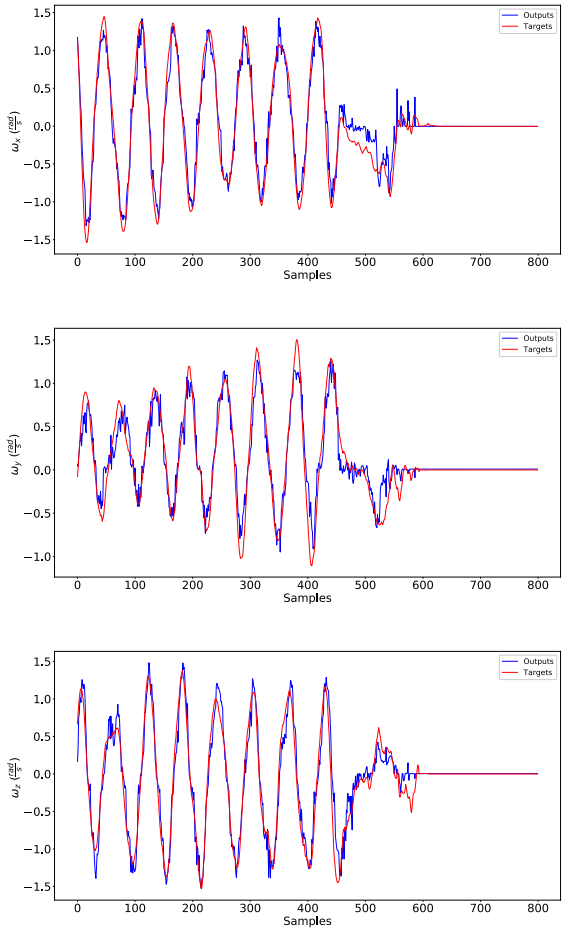

Fig. 7. Targets and outputs temporal evolution in X-Y-Z axes for a subset of the training data. 3 inputs CNN with pixels values

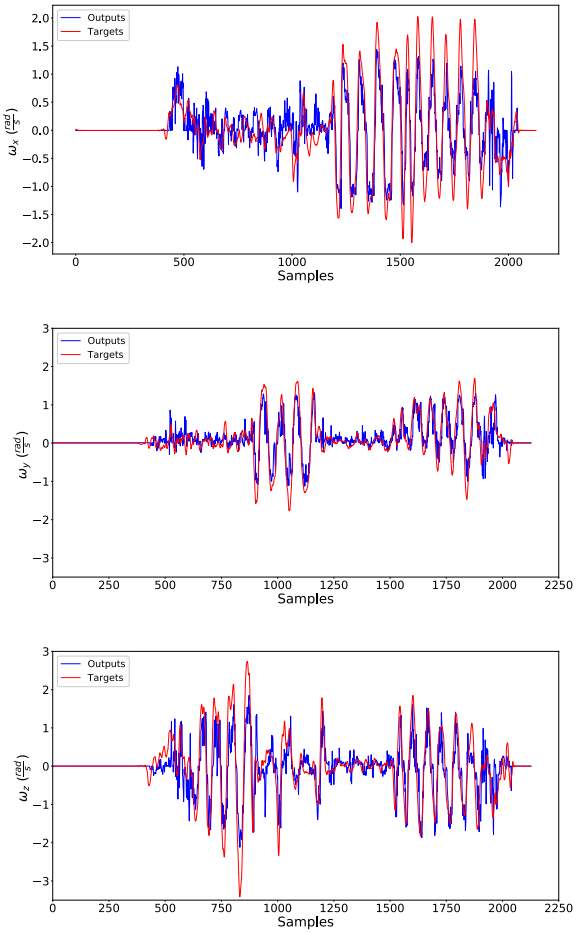

Fig. 8. Targets and outputs temporal evolution in X-Y-Z axes for the validation set. 3 inputs CNN with pixels values.

the estimated response is studied in each axis. The qualitative comparison between the outputs of the network and the known targets for the validation set is shown in Fig.8.

The mean squared error between estimated and real angular rate for each axis and their respective standard errors are presented in Table III. With a MSE of $(0.082 \pm 0.004)\frac{rad^2}{s^2}$ the Y axis (corresponding to roll in our setup) is the best estimated axis, followed by X axis (corresponding to pitch) with a MSE of $(0.144 \pm 0.006)\frac{rad^2}{s^2}$, and $(0.230 \pm 0.009)\frac{rad^2}{s^2}$ on Z axis (yaw), the worst estimated angular rate. The poorer estimation in the yaw rate is in accordance to other results from the literature [17], [18].

V. CONCLUSIONS

This paper has presented a vision-based system for attitude rate estimation for its application in aerial vehicles. The system, inspired by insect ocelli, is based in three small low-resolution cameras. Using a learning approach, the system is able to recover the angular rates from vision-only inputs.

Different ANN architectures have been evaluated. As

TABLE III

3 INPUTS CNN WITH PIXELS ERRORS

Axis	MSE (SEM) (rad^2/s^2)
X	0.144 (0.006)
Y	0.082 (0.004)
Z	0.230 (0.009)

expected, the behavior of CNNs clearly surpasses that of MLPs. Furthermore, using directly the pixels as inputs leads to better results, as the system also learns relevant temporal features. The system is more accurate recovering roll and pitch rates than yaw rates, which is in accordance to the literature considering similar systems [17], [18].

The presented sensor setup and learning-based architecture is a first step. As future work, we plan to evaluate the capabilities of the system in estimating translational velocities, and the ability to close control loops. While small cameras have been used, the same architecture can be employed for faster cameras or systems based on matrices of photodetectors. Also, the ocellar system is related in insects to fast avoidance responses. The system opens the possibility of end-to-end learning of navigation behaviors.

Future work will also consider extending the current database for learning with outdoor experiments. This extended database will be released and open to the public to help the research community to develop new machine learning approaches for UAV attitude estimation.

REFERENCES

[1] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012. [Online]. Available: http://dx.doi.org/10.1177/0278364912455954

[2] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[3] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and slam for medium and high altitude flying uavs," *Journal of Intelligent and robotics systems*, vol. 54, pp. 137–161, 2009.

[4] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 957–964.

[5] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013. [Online]. Available: http://dx.doi.org/10.1177/0278364913481251

[6] L. Merino, J. Wiklund, F. Caballero, A. Moe, J. R. M. De Dios, P.-E. Forssen, K. Nordberg, and A. Ollero, "Vision-based multi-uav position estimation," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 53–62, 2006.

[7] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[8] A. Amor-Martinez, A. Ruiz, F. Moreno-Noguer, and A. Sanfeliu, "On-board real-time pose estimation for uavs using deformable visual contour registration," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2595–2601.

[9] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2609–2616.

[10] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a uav," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 3309–3316.

[11] M. Mizunami, "Functional diversity of neural organization in insect ocellar systems," *Vision Research*, vol. 35, no. 4, pp. 443 – 452, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/004269899400192O

[12] H. G. Krapp, "Ocelli," *Current Biology*, vol. 19, pp. 435–437, 2009.

[13] F. L. Roubieu, J. R. Serres, F. Colonnier, N. Franceschini, S. Viollet, and F. Ruffier, "A biomimetic vision-based hovercraft accounts for bees complex behaviour in various corridors," *Bioinspiration & Biomimetics*, vol. 9, no. 3, p. 036003, 2014. [Online]. Available: http://stacks.iop.org/1748-3190/9/i=3/a=036003

[14] D. Floreano, R. Pericet-Camara, S. Viollet, F. Ruffier, A. Brckner, R. Leitel, W. Buss, M. Menouni, F. Expert, R. Juston, M. K. Dobrzynski, G. LEplattenier, F. Recktenwald, H. A. Mallot, and N. Franceschini, "Miniature curved artificial compound eyes," *Proceedings of the National Academy of Sciences*, vol. 110, no. 23, pp. 9267–9272, 2013. [Online]. Available: http://www.pnas.org/content/110/23/9267.abstract

[15] J. C. Zufferey and D. Floreano, "Fly-inspired visual steering of an ultralight indoor aircraft," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 137–146, Feb 2006.

[16] J. Chahl and A. Mizutani, "Biomimetic attitude and orientation sensors," *IEEE Sensors Journal*, vol. 12, no. 2, pp. 289–297, Feb 2012.

[17] G. Gremillion, J. S. Humbert, and H. G. Krapp, "Bio-inspired modeling and implementation of the ocelli visual system of flying insects," *Biological Cybernetics*, vol. 108, no. 6, pp. 735–746, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00422-014-0610-x

[18] S. B. Fuller, M. Karpelson, A. Censi, K. Y. Ma, and R. J. Wood, "Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli," *Journal of The Royal Society Interface*, vol. 11, no. 97, 2014. [Online]. Available: http://rsif.royalsocietypublishing.org/content/11/97/20140281

[19] T. R. Neumann and H. H. Bülthoff, "Behavior-oriented vision for biomimetic flight control," in *Proceedings of the EPSRC/BBSRC international workshop on biologically inspired robotics*, 2002, pp. 196–203.

[20] P. Sermanet, R. H. M. Scoffier, M. Grimes, J. Ben, A. Erkan, C. Crudele, U. Muller, and Y. Lecun, "A multi-range architecture for collision-free off-road robot navigation," *Journal of Field Robotics*, 2009.

[21] A. Giusti, J. Guzzi, D. C. Cirean, F. L. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, July 2016.

[22] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, p. 2012.

[24] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.