NeuroCopter: Neuromorphic Computation of 6D Ego-Motion of a Quadcopter

Tim Landgraf[1], Benjamin Wild[1], Tobias Ludwig[1], Philipp Nowak[1], Lovisa Helgadottir[2], Benjamin Daumenlang[1], Philipp Breinlinger[1], Martin Nawrot[2], and Raúl Rojas[1]

[1] Freie Universität Berlin, Institut für Informatik, Arnimallee 7, 14195 Berlin, Germany
[2] Freie Universität Berlin, Neuroinformatik, Königin-Luise-Str. 1, 14195 Berlin, Germany

Abstract. The navigation capabilities of honeybees are surprisingly complex. Experimental evidence suggests that honeybees rely on a map-like neuronal representation of the environment. Intriguingly, a honeybee brain exhibits approximately one million neurons only. In an interdisciplinary enterprise, we are investigating models of high-level processing in the nervous system of insects such as spatial mapping and decision making. We use a robotic platform termed NeuroCopter that is controlled by a set of functional modules. Each of these modules initially represents a conventional control method and, in an iterative process, will be replaced by a neural control architecture. This paper describes the neuromorphic extraction of the copter's ego motion from sparse optical flow fields. We will first introduce the reader to the system's architecture and then present a detailed description of the structure of the neural model followed by simulated and real-world results.

Keywords: neural networks, neuromorphic computation, biomimetics, self-localization.

1 Introduction

For decades, honeybees have been used as a model organism to study navigation. They display a rich set of complex navigational capabilities [4,9]. Bees integrate the path they travelled over extended periods of time and keep a robust approximation of the direction to their hive [16]. Recent experimental evidence suggests that - throughout their foraging carreers - bees form a complex map-like memory [10,11,12]. Intriguingly, this map might tell the bees where landmarks are (in an allocentric coordinate system) as opposed to the simplistic alternative that only route memories are formed and triggered by landmarks. Bees display remarkably robust and flexible navigation. They can switch from outbound to inbound routes depending on environmental conditions, choose different targets to visit and even fly novel shortcuts between known locations.

It is desireable to understand the neuronal mechanisms behind the cognitive processes like mapping and decision making in the honeybee for two main

N.F. Lepora et al. (Eds.): Living Machines 2013, LNAI 8064, pp. 143–153, 2013.

reasons: First, on the level of neuronal computation we might discover basic computational motifs and structures responsible for general navigation tasks such as path integration, sensor fusion, state estimation and alike. Second, we want to build robots with similar cognitive capabilities eventually. Recent robotic systems, whether flying or on the ground, already implement methods for simultaneous localization and mapping (SLAM). However, we are interested in the evolution of a neural programming paradigm, which we believe will emerge as an alternative to conventional programming techniques. The paradigm involves three steps: First a complex "brain" is programmed by defining the principal connection scheme of different neural "black boxes" for certain low-level and cognitive functions of a robotic agent. This raw brain is then trained in a simulation that approximates reality sufficiently for the task to be solved and uses techniques as e.g. structural evolution and reinforcement learning to arrive at a brain structure with satisfying performance (e.g. [1]). Then, the matured brain can be copied to the robot that in turn will learn in the real world to fine-tune its brain.

Today, it is still impractical to use complex (spiking) neural networks on autonomous robots since the processing power necessary can not be carried on-board. In the near future, analog neuromorphic hardware (sub-theshold VLSI circuits) will become a powerful alternative to conventional clocked processing units in mobile robots due to their massive parallel processing capabilities and low power consumption [6,13]. Currently, existing analog chips implement only a low number of neurons. The inter-variability of the transistors is still unfeasibly high due to the production process and lacking temperature robustness [14]. However, a great deal of time and financial investments is dedicated to the development of better hardware and industrial production processes. Eventually, neuromorphic hardware will complement conventional processing units in mobile robots.

In an interdisclipinary project with roboticists and neuroscientists ("Project NeuroCopter") we are investigating neural models of honeybees' navigational capabilities and use 3D world simulations and a flying robotic platform to test them. The neural modules are yet simulated in software using Matlab for ANN-models and the neuron simulator IQR [2] for their spiking counterparts. The copter (or the simulated agent in the 3D simulation) sends sensory data to the brain simulation via an interface module that translates numeric data to spikes. The neuro-modules are continuously executed and the activity of pre-defined output neurons is read out and sent back to the real or simulated agents via the interface module. Our long-term goal is to use analog neuromorphic hardware on the copter to reach full autonomy.

This paper presents our approach starting from the mechanics and electronics of our copter, followed by a detailed description of a neuro-module for the computation of the copter's six-dimensional ego motion vector. This module is the first within a computation scheme that represents an initial model of landmark mapping and localization refinement as depicted in Figure 1. We will conclude by discussing future prospects, new neuro-modules and new models of complex insect navigation.

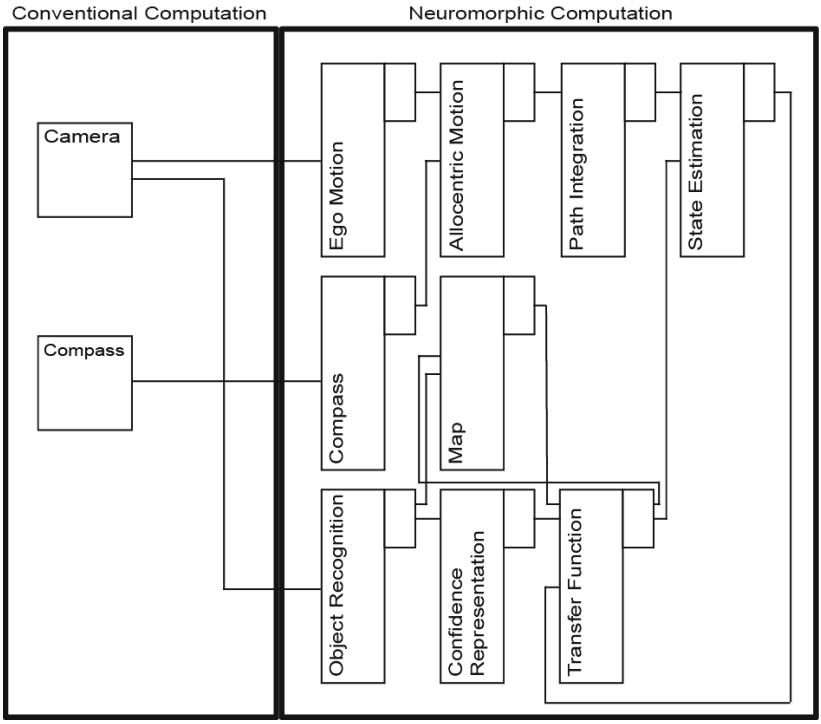

Fig. 1. This figure depicts a connection scheme modeling a central aspect of honeybee navigation: landmark mapping and self-localization based on known landmarks. The connection scheme is composed of neuro-modules, each receiving input at the left side of the respective box and each having a set of output neurons, symbolized by the smaller box to the right. Each module might contain an arbitrary structure and wiring. In the upper left part the neuro-module for ego-motion extraction receives optical flow input from the conventional computer. Its output, a six-dimensional vector coded in an ensemble of neurons, is fed into a module that rotates the ego-motion according to a compass value and translates the motion into an allocentric coordinate system. The output is integrated in the following module which computes the current estimate of the copter's position. This estimate is fed into a neural module responsible for tracking different hypotheses of the own location (e.g. [3]). When landmarks are recognized (either within the neuromodule or outside) their identity is represented by the output of the object recognition module. Its location is fed into the central map, an associative network that receives input from the transfer module which integrates and distributes information among various modules. The confidence module represents the confidence in the position stored in the map. Low confidence drives the transfer network to project the copter's location (as represented by the state estimator) to the map. Is the sensed object known (high confidence) the transfer network updates the state estimation with the output of the map. Medium confidence values result in a mix of both strategies.

2 Mechanics and Electronics of Neurocopter

The copter is a custom built quadrotor. Four carbon fiber tubes are plugged into a central aluminum cross joint milled and turned down to a weight of 30 g. At the end of each tube a brushless DC motor (AC2836-358) is affixed with a custom holder. Each motor is driven by an ESC (electronic speed control, jDrones AC20-1.0) connected to an ArduPilotMega (APM) flight control board which is commercially available and serves as low-level motor controller and flight stabilizer. The board is connected to a GPS unit (MTek GTPA010) and offers common intergrated accelerometer, gyroscope and barometer. The APM connects to a 2.4 GHz 8 Channel radio receiver (Turnigy 9X8CV2) that works with a remote control (Turnigy TGY 9X) for manual / failsafe steering. It also connects through a serial interface with a Linux embedded Cortex A8 based PC (IGEP v2 DM3730). The IGEP runs a message dispatcher that receives UDP packets from the ground station and either collects data from the APM via the serial interface or executes programs on the Linux device itself. The ground station is used for high level control, such as setting semi-automatic behaviors such as starting, landing, position hold, altitude hold or waypoint following. The ground station also collects sensory data for visualization and forwarding to the brain simulation. A common webcam (Logitech Pro 9000) is connected to the IGEP via USB. The total weight of the system is 1340 g and the additional payload possible is approximately 1.5 kg.

Fig. 2. Photograph of the flying NeuroCopter. The quadrotor frame is built from carbon fiber tubes and an aluminum cross joint. The flight control electronics based on the Arduino platform, motors and power supply are standard components. A Linux board is used for wireless communication, camera readout and basic computations. Most processing is done on the ground station's computer.

3 Neuromorphic Extraction of Ego Motion

3.1 General Setup

On the embedded system, we compute a sparse set of image features [15] which are tracked with a pyramidal version of the tracker described in [8]. For each

cell of a 5 x 5 grid optical flow vectors are averaged (Figure 5 depicts a sample image taken in flight). The resulting set of 25 motion vectors are sent down to the ground station via Wifi. The ground station translates the new data set and feeds it to a neural network.

3.2 Simulated Optical Flow

Before developing the neural module we asked whether the optical flow output can be unambiguously assigned to the respective ego-motions. For example, a roll motion produces mostly parallel flow on the image plane as would do a translational sidewards motion (which might occur due to gusts). We therefor simulated a three-dimensional terrain and modelled the camera in use. Artificial optical flow was computed by moving or rotating the camera and projecting a number of ground features onto the image plane before and after the camera motion. Figure 3 depicts the resulting flow in the simulated world.

The optical flow on the image plane was then averaged in a 5 x 5 grid of bins. Each action $\overrightarrow{a}$ (a six-dimensional ego-motion vector) thus is assigned a 50-dimensional data point $\overrightarrow{b}$ (25 bins, each having a two-dimensional vector in the plane). Consequently, we wanted to learn which pairs of actions $(\overrightarrow{a}_i, \overrightarrow{a}_j)$ have a high dissimilarity but produce a pair of similar flow outputs $(\overrightarrow{b}_i, \overrightarrow{b}_j)$. To this end, all data points in the input and output space were normalized such that the absolute maximum actions were vectors of unit length. In the next step, two (eucledian) distance matrices for all pairs in each vector space were computed. Each entry in the distance matrices expresses the similarity of actions or results. Low values denote very similar vectors. To make distances comparable between the two vector spaces, again, the distances were normalized to values between 0 (equal) and 1 (maximum dissimilarity). In the next step all inter-space distances were computed resulting in a set of action-pair-to-result-pair distances. The largest values in this set indicate action pairs having high dissimilarity but corresponding to similar result pairs (or vice versa, although this is another problem not adressed here).

By evaluating the largest distance pairs, we found that some complex motions (high rotational and translational components) might counterbalance the optical flow. If, for example, the copter would rotate into a positive pitch (lean forward), the flow on the image plane would point upwards. If the copter moves forward parallel to the ground surface an opposed optical flow is produced. If the altitude of the copter, the rotational and the forward speed fall into a certain range the flow cancels out and the inverse computation would produce a wrong result. However, this movement (within the erroneous range mentioned) might not happen often under normal operating conditions (a forward pitch normally preceeds the translational motion but strong winds might push the copter in the rotational movement). Secondly, roll and sideward motions or pitch and forward motions do produce similar but not equal optical flow fields: a neural network might well be able to separate both actions with feasible precision.

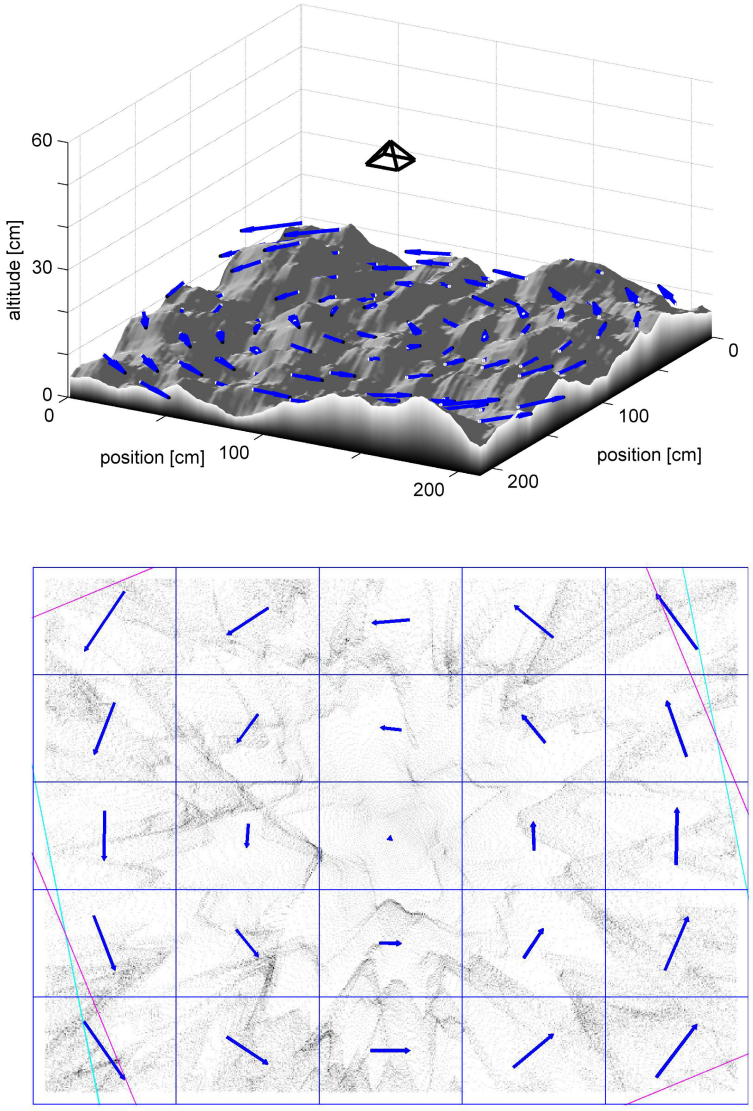

Fig. 3. Rotating a virtual camera (symbolized by the pyramid in the left figure) over a surface introduces image motion on the camera plane. The left figure depicts a three-dimensional surface with a selection of point features (white dots). We simulate the optic flow by projecting those ground features onto the camera plane before and after a certain camera motion. The camera image is divided into 25 bins in which the flow vectors are averaged (blue arrows in the right figure). Those average vectors are the input to the neural network for the ego-motion computation.

We trained a three-layered feed-forward artificial neural network (ANN) using the simulated ego-motion / action pairs. We then tested the reconstruction of a simulated flight by iterating the following commands over a number of time steps:

- move the camera according to a certain (smoothed) six-dimensional ego-motion function
- obtain the flow field by tracking the ground features' projection on the image plane
- average flow vectors in each of the 5 x 5 bins
- feed 50-dimensional input vector into ANN and read result vector

4 Results

The resulting ego-motion could then be integrated over time to reproduce the flight curve. Figure 4 visualizes both the trajectory computed from the input velocities and the network's reproduction. Apparently, the result needs improvement. The reproduction follows the input upwards but substantially deviates from it. The subspace of possible inputs that although being similar are supposed to produce differing outputs could not be learned with optical flow input alone. Feeding the copter's altitude to the network improves the network's performance drastically. This is intuitive: translations in high altitude produce only small flow vectors, whereas rotations are independent of scene depth. It is possible to significantly improve the networks output by updating an initial altitude hypothesis using the ego-motion's upward component and feed this altitude approximation back to the network. This value will accumulate even small errors

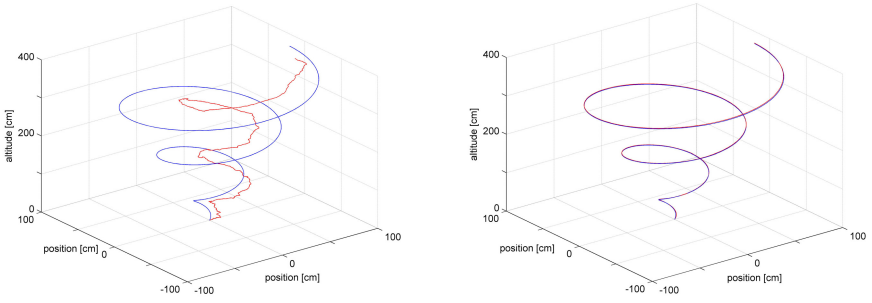

Fig. 4. Integrating the output of the neural network yields a flight trajectory that should preferably equal the trajectory created by integrating the input motions. The left image shows the resulting flight reconstruction of a network that was trained without altitude information. Using the output of the network to update an altitude estimation which is fed back to the network in the following time step yield a precise reproduction of the input.

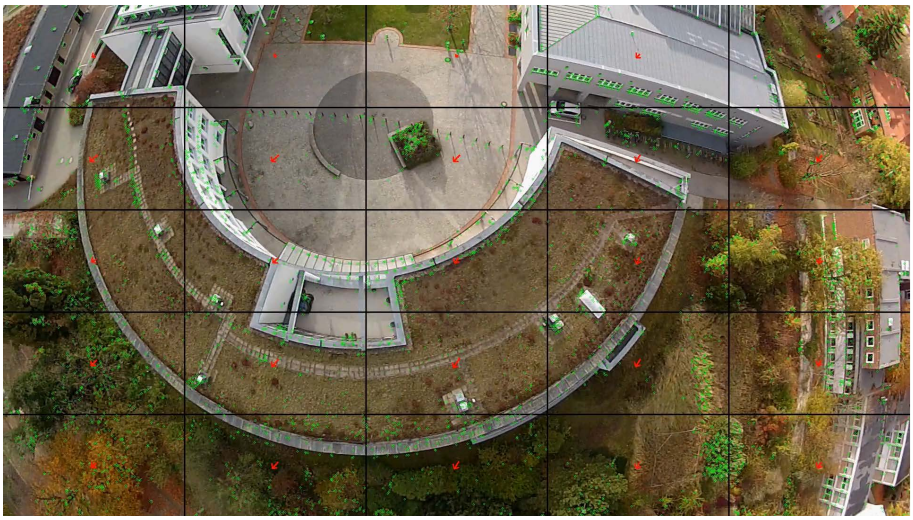

Fig. 5. This figure shows a camera frame captured in-flight over the Institute of Computer Science at Free University Berlin. The frame is overlaid with optical flow features (green points) and average flow vectors (in red) in each of the 25 image bins.

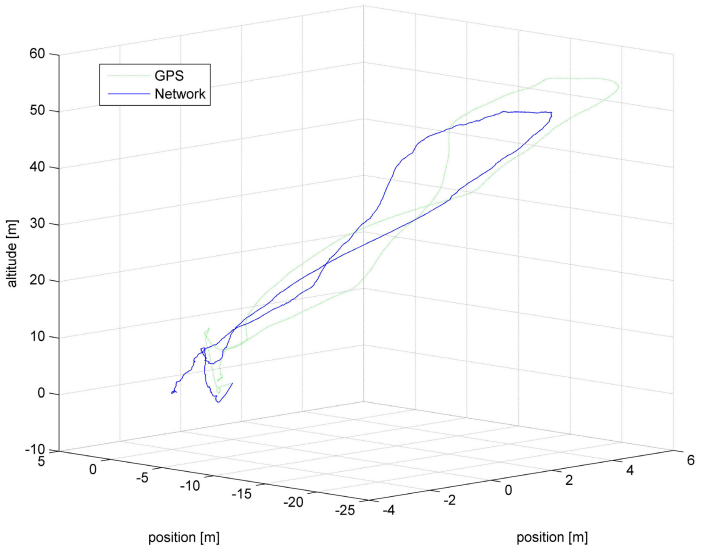

Fig. 6. The figure depicts the flight path reconstruction of a 70 seconds video sequence [7] recorded under strong wind conditions. The path, as computed by the neural module (blue line) resembles closely the trajectory flown (GPS reference trajectory, green dash-dotted line). Fast maneuvers and high flight speeds however lead to significant accumulation of error at the highest point of the path.

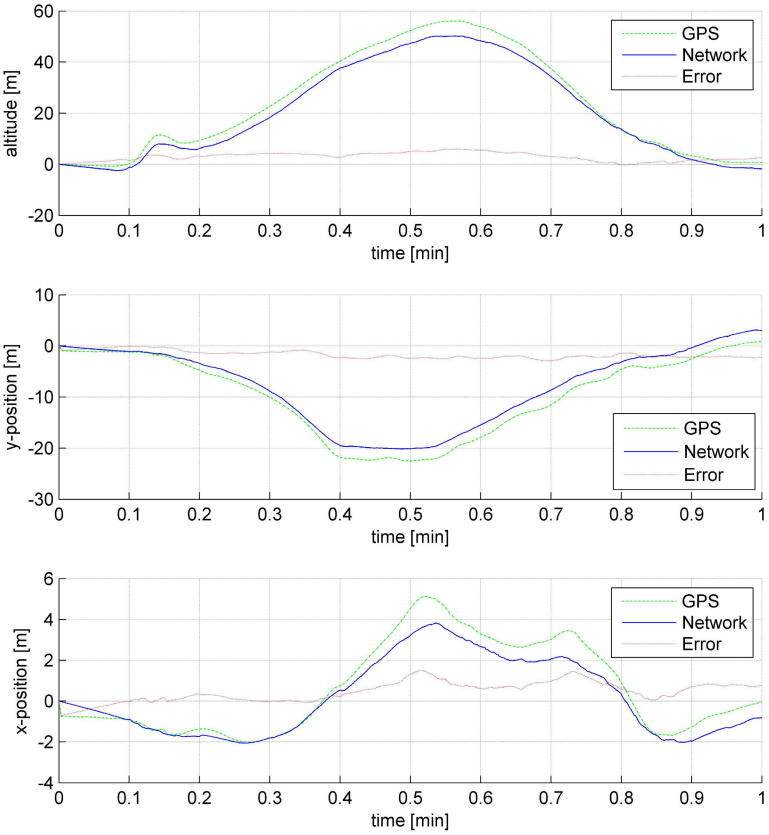

Fig. 7. The reconstruction of the flight path for x, y and z-coordinates. The blue curves depict the network's reconstruction, the green dash-dotted line represents the GPS reference data and the red line describes their difference. Maximum errors for altitude, y-position and x-position are: 5.9 m, 2.9 m, 1.5 m, respectively.

in the computation over time. It is conceivable to either use the copter's barometer or GPS sensor to refine or define the altitude value entering the network. It would also be conceivable to recalibrate the value by static up- or down-movements that produce imploding or exploding flow vectors of a length that is inversely proportional to the altitude.

We tested the ego-motion network module on our copter without XXX feeding back altitude information from the barometer. The test flight was controlled manually by a human pilot. Optical flow was computed on-board and sent down to the ground station. The flow data was fed into the proposed network and the resulting ego-motions were integrated to reproduce the flight trajectory taken. Onboard sensory data was also logged at 50 Hz (gyroscope and accelerometer) and 5 Hz (GPS), respectively. Figure 7 shows the resulting flight path. A video

of the flight sequence with an optical flow overlay can be downloaded at [7]. The reproduction of the flight path resembles closely the actual flight path taken with respect to planar coordinates over ground and altitude.

5 Conclusions and Future Work

This paper shows a neural network structure that computes a six-dimensional ego-motion vector from sparse optical flow fields. The network does not require additional sensors other than the camera used. It yields a sufficiently precise estimation of the motion state of the copter that can be processed further in our neural processing scheme.

Since GPS and barometer on the copter are rather imprecise our confidence in a validation using these references is low. We are currently integrating a sonar for higher altitude precisison and are implementing a conventional, non-neural optical flow odometer. Both sensory systems will be used to increase precision of the state estimation. However, the proposed neural system does not necessarily have to be of high precision. One of our aims is to form a memory of the position of landmarks in the neural network. A landmark that is discovered the first time will be assigned the current allocentric self-position. The more often a landmark is revisited from different directions the better the stored position will get and the higher the confidence in this value. If the copter encounters a known landmark the position will tune the self-localization network and improve the copter's own position estimate.

The focus of this work lies in the formulation of our general approach to investigate neural processing units for navigation in bees. The neural module for the extraction of the robot's ego-motion exemplifies the approach on the module level. We are currently experimenting with prototypes of the neuro-modules for state estimation and the transfer function as shown in 1. The scheme however is not complete. The neural network might be able to map landmarks and use this information to refine the copter's postion estimate. Other cognitive processes, such as decision making or motivation for certain behaviors are not yet integrated into the model. In the near future conventionally implemented functionality will be replaced step by step by neural modules. In the upcoming summer period we are planning to conduct behavioral experiments with the copter. We will investigate the question whether the model can reproduce behaviors observed in flights of real bees (e.g. novel shortcuts). Consequently, we will test different models of insect navigation (e.g. [5]) and compare the resulting behavioral repertoire to the one found in nature.

Acknowledgement. Partial funding was received from the German Ministry of Education and Research within the collaborative project Insect Inspired Robots (grant 01GQ0941 to M.N. and R.R.).

References

1. Teller, M.V.A.: Neural Programming and an Internal Reinforcement Policy
2. Bernardet, U., Blanchard, M., Verschure, P.F.M.J.: IQR: a distributed system for real-time real-world neuronal simulation. Neurocomputing 44-46(null), 1043–1048 (2002)
3. Bobrowski, O., Meir, R., Eldar, Y.C.: Bayesian filtering in spiking neural networks: noise, adaptation, and multisensory integration. Neural Computation 21(5), 1277–1320 (2009)
4. Collett, T.S., Collett, M.: Memory use in insect visual navigation. Nature Reviews. Neuroscience 3(7), 542–552 (2002)
5. Cruse, H., Wehner, R.: No need for a cognitive map: decentralized memory for insect navigation. PLoS Computational Biology 7(3), e1002009 (2011)
6. Indiveri, G., Linares-Barranco, B., Hamilton, T.J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al: Neuromorphic silicon neuron circuits. Frontiers in Neuroscience 5 (2011)
7. Landgraf, T., Wild, B., Ludwig, T., Nowak, P., Helgadottir, B., Daumenlang, L., Breinlinger, P., Nawrot, M., Rojas, R.: Test Flight for Living Machines 2013 (2013)
8. Lucas, B.D., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. Imaging 130(x), 674–679 (1981)
9. Menzel, R.: The honeybee as a model for understanding the basis of cognition. Nature Reviews Neuroscience 13(11), 758–768 (2012)
10. Menzel, R., Greggers, U., Smith, A., Berger, S., Brandt, R., Brunke, S., Bundrock, G., Hülse, S., Plümpe, T., Schaupp, F., Schüttler, E., Stach, S., Stindt, J., Stollhoff, N., Watzl, S.: Honey bees navigate according to a map-like spatial memory. Proceedings of the National Academy of Sciences of the United States of America 102(8), 3040–3045 (2005)
11. Menzel, R., Kirbach, A., Haass, W.-D., Fischer, B., Fuchs, J., Koblofsky, M., Lehmann, K., Reiter, L., Meyer, H., Nguyen, H., et al.: A common frame of reference for learned and communicated vectors in honeybee navigation. Current Biology 21(8), 645–650 (2011)
12. Menzel, R., Lehmann, K., Manz, G., Fuchs, J., Koblofsky, M., Greggers, U.: Vector integration and novel shortcutting in honeybee navigation. Apidologie 43(3), 229–243 (2012)
13. Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M.A., Schmuker, M., Brüderle, D., Schemmel, J., Meier, K.: Six networks on a universal neuromorphic computing substrate. Frontiers in Neuroscience 7 (2013)
14. Poon, C.-S., Zhou, K.: Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities. Frontiers in Neuroscience 5, 108 (2011)
15. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR 1994, pp. 593–600. IEEE Comput. Soc. Press (1994)
16. Wehner, R., Srinivasan, M.V.: Path integration in insects (2003)