An autonomous flying robot for testing bio-inspired navigation strategies

Verena V. Hafner, Ferry Bachmann, Oswald Berthold, Michael Schulz and Mathias Müller
Humboldt-Universität zu Berlin, Institut für Informatik, LS Kognitive Robotik
Unter den Linden 6, 10099 Berlin, Germany

Abstract

In this paper, we present our approach to studying bio-inspired navigation strategies on an autonomous flying robot. The robot is a quadrocopter that is based on technology from Mikrokopter and extended with several sensor systems, most of them inspired by sensors that animals use for navigating in their natural environment. Using a light-weight embedded computer from gumstix onboard the copter, the system can be fully autonomous. We present prototypical experiments on visual homing, self-stabilisation using optic flow, and altitude control using evolutionary strategies.

1 Introduction

Despite many years of research and development, autonomous navigation is still an open challenge for robotics. Traditional approaches such as SLAM (simultaneous localisation and mapping) require a highly structured environment and accurate and stable sensory information. When bringing in another dimension and expanding the challenge of autonomous navigation to small flying robots, most traditional approaches fail due to additional constraints and conditions such as fast changing sensory data, weight limitations for the sensor system, higher dimensionality of the state space and the system being prone to external manipulations more easily.

There are currently no artificial systems that solve all these challenges, but there is evidence that autonomous flying systems exist that can navigate in new, changing and unstructured environments: flying insects. By looking at the principles of the navigation strategies of these insects and their adaptation to body and environment, it might be possible to construct artificial systems with similar capabilities. The navigation strategies of animals on the ground have already successfully been tested on wheeled mobile robots; see for example the Sahabot project[1].

We present an approach to studying bio-inspired navigation strategies using a lightweight quadrocopter based on Mikrokopter[2] technology and extended with several bio-inspired sensor systems. For self-stabilisation, the Mikrokopter sensors and software are used. These consist of a three-axis acceleration sensor and three gyroscopes. Processing of the navigation strategies and communication with the flight control board is performed onboard on a gumstix[3] embedded computer.

Our bio-inspired sensor approaches so far include a pan-tilt camera system below the robot used for measuring optic flow, a camera system constructed using a convex mirror to realise omnidirectional vision, and a compass based on the polarisation pattern of the sunlight.

Adaptive control of small aerial vehicles has so far mainly been tested in simulation[4]. The most common form of autonomous navigation are waypoint flights using a GPS system[5]. Grzonka et al.[6] use a smaller laser sensor in indoor and outdoor flights. Kendoul et al.[7] use optic flow for an indoor quadrotor helicopter.

Our team will also participate at the motodrone[8] challenges 2010 just before ISR/Robotik 2010 and share the algorithms and results there.

2 Quadrocopter platform

The platform presented in this paper is based on the Mikrokopter[2] construction set. The platform has a weight of about 1000g including batteries and can carry an additional load of about 400g. The choice for this set was made because of its relatively low price, good support from the user community and, essential for our experiments, the open-source software. The quadrocopter (or quadrotor) is a flying robot with four propellers in cross configuration (see Figure 1). Two of the four propellers rotate clockwise, the other two counterclockwise. Thus, with all propellers rotating at the same speed, the resulting torque around the yaw axis of the whole quadrocopter is nearly zero. Even when the goal is to just hover in the air without any movements, the quadrocopter is still a dynamically unstable system. Thus, to make human or high level autonomous control possible, the system is equipped with an internal measurement unit (IMU) and a basic attitude control system. For details about attitude control of a quadrocopter see Bouabdallah et al.[9]. On top of this system the platform is controllable in yaw, pitch and roll by changing the relative rotor speeds and the lift force can be controlled by changing the speed of all rotors simultaneously[10]. This kind of

control is the starting point for our navigation experiments. Figure 1 shows a picture of the quadrocopter platform presented in this paper. The following subsections give an overview of the sensors, actuators and the system architecture of the platform. Quadrotors are becoming a common platform in flying robot navigation[11][12][13].

Figure 1: Quadrocopter platform.

2.1 Sensors and actuators

The Mikrokopter is driven by four brushless DC motors with 10" propellers. The platform is equipped with the standard IMU sensors – three gyroscopes and a 3D-accelerometer. The gyroscopes measure angular velocities up to 300deg/sec, the operating range of the acceleration sensors is $\pm$2g. There is an additional barometric pressure sensors for altitude control. The granularity of the altitude measurement is about 10cm. Because of fluctuations in the barometric pressure this is just a theoretical value. These fluctuations increase strongly when approaching the ground because of the turbulences caused by the propellers. Therefore this sensor is almost unusable for autonomous take-off and landing. This is why we extended the platform with an ultrasonic distance sensor (Devantech SRF02) facing downwards. The operating range goes from 10cm up to 6m making it ideal for take-off and landing control. Another extension is a pan-tilt unit with a small camera at the bottom of the quadrocopter. The pan-tilt unit is actuated by two micro servos. A simple controller uses the acceleration data of the IMU to control the camera attitude. This way the camera can face downwards independently from tilts up to 50° of the quadrocopter. This is an important precondition for the visual approaches presented below. The camera currently used is a simple low cost USB webcam (Logitech QuickCam E 3500).

2.2 System architecture

The basic computation unit of the quadrocopter is an Atmega 644 microcontroller. All sensors and actuators described in the previous subsection are directly connected to this unit. It reads out all sensory data and does the basic attitude control. The control loop frequency is 500Hz. As described earlier, we intend to control the yaw, pitch and roll movements of the quadrocopter on top of the basic attitude control. Therefore we added a gumstix verdex computer which communicates with the Atmega controller with a specified communication protocol. The gumstix can select a set of sensory data and a sample frequency and then the Atmega sends out this sensory data with the given frequency via the UART to the gumstix computer. The gumstix can do whatever kind of high level control needed and answer with yaw, pitch and roll commands to the Atmega. This way high level control frequencies up to 100Hz are possible. We are currently testing Xenomai[14] to provide hard real time support for critical control parts. The gumstix provides the possibility to easily and quickly connect sensors to the platform via standard interfaces. The camera currently used is simply connected to the USB port.

3 Simulation of the Multicopter platform

3.1 Framework

In a robotics project simulations come in handy at various stages of the development process[15], in particular when evolutionary strategies are involved. It is however important to keep in mind the limitations of a simulation concerning real world errors, noise, and complex physics. In addition to employing micro-simulations within standard numeric computing packages for partial aspects of the system, we have been looking for a complete realtime simulation platform that includes physics as well as an internal controller architecture.
The initial options have been developing such a platform from scratch or using existing solutions provided by the Open Source market. The from-scratch approach was dismissed as too expensive. The autopilot project that was used by DeNardi et al.[4] seems to suffer discontinued development since around 2005. Another option, the Flug-Modell Simulator (FMS)[16] does not run natively on Linux and, most importantly, no source code is available for this software. Finally, the crrcsim[17] package was chosen as a basis for our experiments due to its overall matureness, modifiability through published sources and the inclusion of a working multicopter model thanks to comprehensive efforts by Jens W. Wulf[18].
Crrcsim provides fine-grained control over various aspects of the simulation. The system comprises the vehicle model (geometry, motor attachment points, graphics), rigid body equations of motion module, flight data model, motor

model, environmental factors such as wind and thermals and graphical output. The latter can be suppressed to increase overall execution speed if needed.

3.2 Experiments

At the current state of the project, we intend to use the simulator for the evaluation and verification of different approaches to controllers and sensory integration. As is the case with the Mikrokopter[2] hardware and software, the crrcsim multicopter model comes with attitude and rate controllers for pitch, roll and yaw. Altitude and positional PID-controllers had to be added. At first, these were parametrised empirically.

Since its primary intended use is for training RC-model piloting, the input channels include among others the soundcard, mouse or keyboard and joystick. For our purposes a UDP based network input method had to be implemented so that external programs can interact with the running simulation. This does not only apply to copter control but also simulation state control, which is necessary for the realisation of evolutionary and other learning methods.

3.3 Evolution

An initial experiment was to evolve the P, I and D parameters of the altitude controller using evolutionary strategies[19].

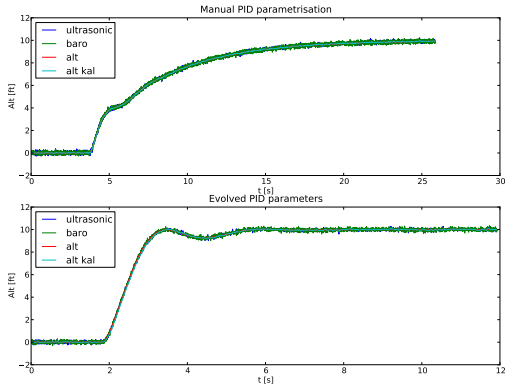

Figure 2: Comparison of manually set and evolved PID parameters. Note the different timescales in the two plots. Top: $(P, I, D)_{man} = (0.05, 0.01, 0.012)$, Bottom: $(P, I, D)_{evo} = (0.0663, 0.0534, 0.0189)$.

After bringing the architectural conditions in place, it was easy to find adequate parameters for the controller in three evolutionary runs after 30 to 100 generations (Figures 2 and 3) and a population size of 20.

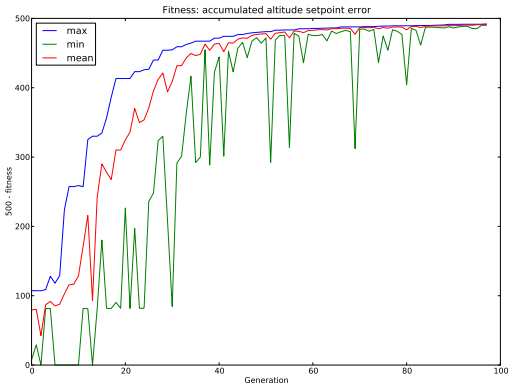

Figure 3: Evolved controller fitness.

The four fittest individuals were carried over to the new population (elitism) and the remaining 16 slots were populated with offspring generated by selecting parents with fitness-weighted probabilities and then applying a mutation operator.

As the experiment was rather a proof of concept, fitness evaluation was based asymmetrically on ascent behaviour. The copter was supposed to climb to a given altitude setpoint with an initial altitude of zero. Fitness was calculated by

$$fitness = \sum_{n} |alt_{set}[n] - alt_{act}[n]|$$

Management of the evolutionary process and control of crrcsim was implemented in Python.

3.4 Sensor fusion

The individual's fitness in the evolutionary process described above was evaluated using the exact altitude value available from the flight data model. As a next step we chose to simulate noisy sensors modelled after those actually available on our hardware. These are a barometric sensor and sonar. The barometric sensor works over a larger range of actual altitudes but is subject to slow pressure drifts, ground effect and various other unwanted influences. The ultrasonic sensor is more precise but only within the range of approximately 0.1 up to 6.0 meters. Also, the sensor update rate depends on the actual distance being measured, since it is TOF based.

After modelling of the individual sensor characteristics, we have looked at and implemented a Kalman filter (KF) for fusing these two sensory channels into a single altitude estimation[20, 21]. Even with minor investment in modelling, this setup works already very well and is awaiting verification on the hardware.

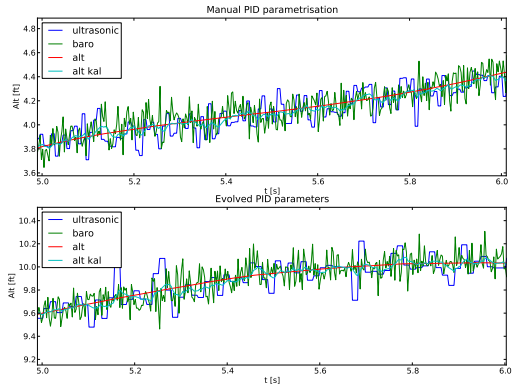

Figure 4: Close-up of Kalman filter output alongside with raw sensor values and simulation state.

3.5 Outlook

We plan to test several more modules within the given simulation framework. These include different types of neural-networks as controllers for the tasks described above as well as simulating onboard camera-sensors to be able to refine the optic-flow based approaches to copter control and navigation.

4 Visual approaches based on Optic Flow on the Quadrocopter

Many insects apply optic flow techniques to estimate egomotion, avoid collisions with obstacles and land smoothly on objects or the ground. This ability is already hard-coded in their nervous system by elementary motion detectors (EMD). In our approach, we use first order differential methods for detecting optic flow since they do not require object recognition and are therefore not too computationally intense. We are currently testing different methods to adaptively correct egomotion sensed by using the optic flow information.
We decided to implement our vision algorithms within the OpenCV[22] framework, as it was designed for computational efficiency and provides a simple-to-use computer vision infrastructure. A further advantage of OpenCV is the availability of precompiled packages for OpenEmbedded which makes it easy to install on gumstix.

4.1 Calculating Optic Flow

The initial hypothesis behind differential methods is that the image intensity I is approximately constant under motion (u, v). Following the standard approach to differential techniques[23], a first-order Taylor series expansion can be applied to obtain

$$I_x u + I_y v + I_t = 0, \quad \text{(optical flow constraint equation)}$$

where I_x, I_y, and I_t denoting partial derivatives with respect to the coordinates (x, y) and time t. The optical flow constraint equation is one linear equation in the two unknowns u and v and therefore ill-posed. Thus, we add the constraint that the image velocity (u, v) has to be aligned with the image gradient. This yields in

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} I_x \\ I_y \end{pmatrix} \frac{-I_t}{\sqrt{I_x^2 + I_y^2}}.$$

It is recommended to use the gradient magnitude as a reliability measure, i.e. to set the velocity to zero if the gradient magnitude is below a given threshold.

Although the algorithm is computational easy, it is difficult to guarantee short time intervals on general purpose hardware like our gumstix with a webcam connected to USB. Another approach is to turn to region-based matching, which defines velocity as the shift that yields the best fit between regions at different times. The main advantage of this approach is the handling of large displacements. A promising method is presented by Fridtjof Stein[24] which uses the Census Transform as the representation of regions and matches them using a table based indexing scheme.

4.2 Egomotion

Egomotion of the quadrocopter can best be measured by fusing information from inertial sensors and visual optic flow data. Since the quadrocopter has 6 degrees of freedom, it is difficult to estimate the egomotion only from a 2D image. To simplify our model, the 3 degrees of rotation can be derived using the information from the gyroscopes. The altitude is given by the barometric sensor and sonar, so that the best camera orientation seems to be parallel to the ground plane. Instead of doing an expensive image transformation, we compensate the observed motion along roll and pitch angles. This is accomplished in hardware by coupling the integral part of the gyroscopes with the pan-tilt camera mount. The remaining estimation of translation in the xy-plane is inversely proportional to the altitude and can be added to the stick values for pitch and roll using a simple PD-controller.

4.3 Outlook

In addition to lateral self-stabilisation, optic flow methods will also be applied for autonomous take-off and landing, obstacle avoidance and path integration. One approach to solve the speed problem of calculating optic flow is the use of an FPGA which will be developed as part of a diploma thesis. Optic flow is an ideal application for parallelisation. Other approaches of using optic flow for UAV control have been presented in [11] and [7].

5 Local Visual Homing

In spite of their tiny brains, insects have some impressive navigation skills. One of them is the ability to return to previously visited places by using visual sensory data. In order to achieve this, they compare a stored representation (referred to as snapshot) of their visually perceived environment at the goal position with the currently available visual sensory data (referred to as current view)[25]. From this comparison, a so-called home vector pointing to the goal position is derived[26]. In recent years, many algorithms have been developed for robot navigation that are based on the snapshot model just described. In general, these local visual homing methods primarily use the current visual sensory data and an internal representation of the goal or home position, but they do not need additional information concerning the environment nor has a self-localisation task to be solved. For this reason the required computational effort is relatively small, yielding an implementation on an embedded computer like the gumstix.

5.1 Warping

One of these local visual homing methods that we are currently testing is the warping method suggested by Franz et al.[27]. Like most methods of its class, warping is based on the processing of omnidirectional panoramic images. That is why we constructed an omnidirectional camera system for our quadrocopter, based on a camera pointing towards a small hollow shiny half sphere. In figure 6, a sample picture taken by the omnidirectional camera system in the air on our quadrocopter is shown. We are well aware that the shape of the mirror is not optimal, however it is sufficient for our purposes and very light-weight.

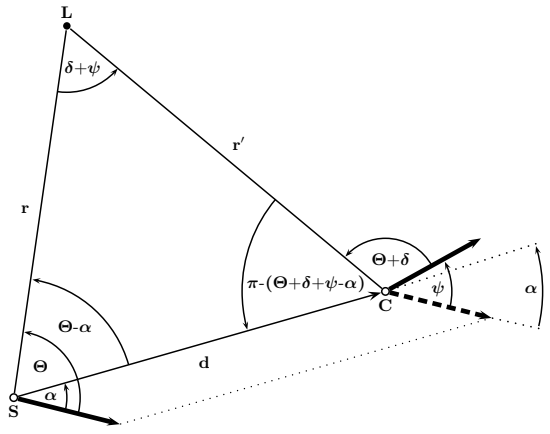

Figure 5: Warping – geometric relations.

The main idea is to distort the current view according to some parameters, which describes the movement of the agent, in order to get a *warped* version of this image. In this way the snapshot can be approximated and finally the direction to the snapshot position can be determined.
Let S be the snapshot (or goal) position, C the current position and L a landmark in the environment (see Figure 5). At position S the agent is oriented as indicated by the thick arrow and perceives L under the horizontal angle Θ. If it moves in direction α relative to its initial orientation and travels a distance d, it arrives at C. The movement is completed by a rotation ψ. The landmark L is still perceivable at position C, but appears under the angle $\Theta + \delta$ now. Under the equal distance assumption (EDA), which means that all landmarks are located at the same distance from S, a warp function can be derived from the geometric relations shown in Figure 5 such that

$$\Theta + \delta = \text{warp}(\Theta|\alpha, \psi, \rho) \quad || \; \rho = \frac{d}{r},$$

where ρ is the relative distance due to the EDA. We have to take this assumption, because r and r' are unknown without additional knowledge about the environment. Although the EDA is usually violated, warping delivers fairly accurate home vectors[27, 28]. Within an exhaustive minimum search through the parameter space with respect to a distance measure between the images, the warp function is used to distort the current view according to a given set of parameters (α, ψ, ρ).

Figure 6: Omnidirectional camera image taken on a quadrocopter flight.

Concerning our project, we have to deal with two problems here. First, as a function of the chosen resolution the search space gets really large and furthermore it is interspersed with local extrema, which inhibits the usage of gradient descent methods. Second, in the original version by Franz et al.[27] the warp function has to be applied in every search step, which accounts a high computational effort. That is why the original version (referred to as 1D-warping) is only practically feasible if the panoramic images are one-dimensional. Even so, the limited computational power in conjunction with the absence of a floating point unit leads to unacceptable runtimes on the gumstix. Over and above,

1D-warping is not competitive nowadays because it is outperformed by modern optic flow methods[28].
Fortunately, there is a reformulated version of Franz et al.'s warping suggested by Möller[28] (referred to as 2D-warping), which solves most of the problems just mentioned. As its name implies, 2D-warping processes two-dimensional panoramic images, which in contrast to 1D-warping makes it unnecessary to drop useful information provided by our omnidirectional camera system. A significant decrease of the overall runtime is achieved by using lookup-tables to store as many precalculations as possible. More in detail two lookup-tables are created: A warp table to store the image of the warp function with respect to the search space as well as the width of the panoramic images and a so-called distance image to store all possible column comparison results between snapshot and current view (given a distance measure).
One problem remains at this juncture: Due to the potentially high number of search steps the warp table can be very large. For instance if one chooses 72 search steps for α, 72 search steps for ψ, 20 search steps for ρ and if the image width is 295, the necessary memory size is roughly 58 MB. This would require nearly half the amount of the gumstix' RAM, which is definitely too much in order to be used in our case. Fortunately, warping is very robust with respect to the decrease of the number of search steps and thus small-sized warp tables can be constructed without losing too much accuracy.

5.2 Outlook

At the current stage we are optimising our implementation of the 2D-warping and we are preparing online tests on our quadrocopter. Besides we are planning to integrate our sensory compass data into the warping process in order to, on the one hand, further reduce the overall runtime and, on the other hand, increase the accuracy.

6 Conclusions

Our system can be used to test different hypotheses on navigation behaviour in animals and extract principles of this behaviour to realise fully autonomous flying robots. We composed a system that fulfils most of the requirements for an autonomous light-weight bio-inspired flying robot. Specifically, we showed experiments on autonomous altitude control using evolutionary strategies, a pan-tilt camera system that measures optic flow for lateral self-stabilisation and a 2D image warping method based on omnidirectional images taken with a special camera system.
There are a variety of interesting real-world applications for autonomous flying robots ranging from disaster management, to surveillance, to applications in precision farming. We have currently started a project for the latter application with several other partners (see `http://agricopter.de/` for reference). There, we will develop an autonomous flying robot for creating aerial maps of fields that can be used for heterogeneous fertilisation. For practical reasons, this system will be extended with GPS in addition to other more biologically inspired sensors and strategies.

Acknowledgements

We thank everybody who contributed to the success of the quadrocopter navigation project, in particular Carsten Huhn, Martin Schumann, Martin Bertheau, Knut Müller, Christophe Gleim, Philipp Stockschlaeder, Benjamin Gehrels, Monika Domanska, Florian Holzhauer, Alexander Schulz and Martin Martius. We also thank Holger Buss and Ingo Busker for their great hardware, and the founders of motodrone e.V. for organising the motodrone competition.
The project is now partially funded by Deutsche Bundesstiftung Umwelt (DBU).

References

[1] Lambrinos, D., Kobayashi, H., Pfeifer, R., Maris, M., Labhart, T., and Wehner, R.: *An Autonomous Agent Navigating with a Polarized Light Compass*, Adaptive Behavior 6(1), pp. 131-161, 1997

[2] Holger Buss and Ingo Busker: Mikrokopter Platform, `http://www.mikrokopter.de/`, 2006-2010

[3] Gumstix, `http://www.gumstix.com/`

[4] Renzo De Nardi, Julian Togelius, Owen E. Holland and Simon M. Lucas: *Evolution of Neural Networks for Helicopter Control: Why Modularity Matters*, Proceedings of the IEEE Congress on Evolutionary Computation (CEC), July 2006

[5] Kim J. and Sukkarieh S.: *Flight Test Results of GPS/INS Navigation Loop for an Autonomous Unmanned Aerial Vehicle (UAV)*, In International Technical Meeting of the Satellite Division of the Institute of Navigation (ION'02), pp. 510-517, 2002

[6] Slawomir Grzonka, Giorgio Grisetti, Wolfram Burgard: *Towards a Navigation System for Autonomous Indoor Flying*, In Proceedings of the International Conference on Robotics and Automation (ICRA), 2009

[7] Farid Kendoul, Isabelle Fantoni, Kenzo Nonami: *Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles*, Robotics and Autonomous Systems, 57(6-7), pp. 591-602, 2009

[8] motodrone, `http://www.motodrone.de/`

[9] Samir Bouabdallah, André Noth and Roland Siegwart: *PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor*, IEEE International Conference on Intelligent Robots and Systems (IROS), pp.2451–2456, 2004

[10] Samir Bouabdallah, Pierpaolo Murrieri and Roland Siegwart: *Design and Control of an Indoor Micro Quadrotor*, In Proceedings of the International Conference on Robotics and Automation (ICRA), Vol. 5, pp. 4393-4398, 2004

[11] Bruno Herisse, Francois-Xavier Russotto, Tarek Hamel, Robert Mahony: *Hovering flight and vertical landing control of a VTOL Unmanned Aerial Vehicle using Optical Flow*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2008

[12] Juergen Eckert, Falko Dressler and Reinhard German: *An Indoor Localization Framework for Fourrotor Flying Robots Using Low-power Sensor Nodes*, Technical Report, University of Erlangen, 2009

[13] Karl E. Wenzel, Paul Rosset and Andreas Zell: *Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller*, Journal of Intelligent and Robotic Systems, 57:1-4, pp. 297-311, 2010

[14] Xenomai, `http://www.xenomai.org/`

[15] J.-C. Zufferey: *Bio-inspired Flying Robots - Experimental Synthesis of Autonomous Indoor Flyers* , CRC Press, 2008

[16] Flug-Modell-Simulator, `http://n.ethz.ch/student/mmoeller/fms/index.html`

[17] Charles River Radio Control Club Flight Simulator Project, `http://crrcsim.berlios.de/wiki/`

[18] Jens Wilhelm Wulf: *How to adjust multicopter parameters, multicopter model version 1*, crrcsim Documentation

[19] Ingo Rechenberg: *Evolutionsstrategien*, Fromman-Holzboog, 1973

[20] Greg Welch and Gary Bishop: *An Introduction to the Kalman Filter*, SIGGRAPH 2001 course 8. In Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques. ACM Press, Addison-Wesley, Los Angeles, CA, USA, 2001

[21] Hugh Durrant-Whyte: *Multi Sensor Data Fusion*, Course Notes University of Sydney, 2006

[22] Open Source Vision Library (OpenCV), `http://sourceforge.net/projects/opencvlibrary`.

[23] S. S. Beauchemin and J. L. Barron: *The Computation of Optical Flow*, ACM Computing Surveys, 27(3):433–467, 1995

[24] Fridtjof Stein: *Efficient Computation of Optical Flow Using the Census Transform*, Lecture Notes in Computer Science, 3175/2004:79–86, 2004

[25] R. Wehner and F. Räber: *Visual spatial memory in desert ants, Cataglyphis bicolor (Hymenoptera: Formicidae)* Experientia, 35:1569-1571, 1979

[26] B. A. Cartwright and T. S. Collett: *Landmark learning in bees* Journal of Comparative Physiology A, 151:521-543, 1983.

[27] Matthias O. Franz, Bernhard Schölkopf, Hanspeter A. Mallot and Heinrich H. Bülthoff: *Where did I take that snapshot? - Scene-based homing by image matching*, Biological Cybernetics, 79:191–202, 1998

[28] Ralf Möller: *Local Visual Homing by Warping of Two-Dimensional Images*, Robotics and Autonomous Systems, 57(1):87–101, 2009