Neural sensorimotor primitives for vision-controlled flying robots

Oswald Berthold and Verena V. Hafner

I. Abstract

***Abstract*— Vision-based control of aerial robots adds the challenge of visual sensing on top of the basic control problems. Here we pursue the approach of using sensor-coupled motion primitives in order to tie together both the sensor and motor components of flying robots to leverage sensorimotor interaction in a fundamental manner. We also consider the temporal scale over which autonomy is to be achieved. Longer term autonomy requires significant adaptive capacity of the respective control structures. We investigate linear combinations of some nonlinear network activity for the generation of motor output and ultimately, behaviour. In particular we are interested in how such networks can be set up to autonomously learn a desired behaviour. For that, we are looking at bootstrapping of motor primitives. The contribution of this work is a neurally inspired architecture for the representation of sensorimotor primitives capable of online learning on a flying robot.**

II. Introduction

Our aim is the design of control circuits for multirotor type helicopters to safely and robustly move and navigate in unmodified everyday environments. This requires a highly adaptive architecture which can be realized by using Reinforcement Learning (RL) techniques in combination with autonomously acquired internal models of the robot to be controlled. The main sensory mode is vision, aided by several auxiliary channels, the overall dynamics of the underlying system, however, are largely unaffected by the choice of sensors. This course of action is motivated both by technical requirements and biological models. The general setting is to start the robot learning episode with a default parameterization of the primitive, or policy. The policy is stochastic and the system follows the gradient of a cost function with respect to the policy parameters. The major challenge is to do this without the robot damaging or destroying itself.

The control problems that we consider here are basic stabilization of positions in space in the presence of continuous perturbation and noise. We consider the question: Can we bootstrap a controller based purely on robotic self-exploration with the marginal condition of not destroying the robot during learning?

III. Related work

A. Biology

The biological idea of primitives has a long history. Motor primitives date back at least to Helmholtz and are still receiving current interest. In biological context motion primitives are mostly referred to as Central Pattern Generators (CPG), emphasizing the fact of their autonomous nature. The decomposition of sensing into the activity of specialist primitives starts at the latest with the ecological approach of Gibson. In investigations of motor control, several authors describe a coherent picture of the features of biological control systems (Arbib 1981; Mussa-Ivaldi and Solla 2004; Grillner 2006; Ijspeert 2008).

In summary, these features are their distributed character which results in intelligent behaviour from the ground up through component autonomy and goal-awareness at lowest levels due to motor-referenced sensory primitives; their subsumption of concepts such as regulators, feedback controllers, homeostasis and their mediating function in output transformation; presence of internal models coupled with identification mechanisms to realize adaptive control; the ability to realize minute yet significant adjustment which is in many cases necessary for successful actions. Overall, adaptability appears to be favored over precision.

B. Proposed architectures for representing primitives

Recently proposed computational models for primitive representation can be classified according to their location on the model-based to model-free spectrum. On the model-based end, Lupashin et al. (2010); Schoellig, Mueller, and D'Andrea (2012); Mellinger, Michael, and Kumar (2012) use a detailed quadrotor flight dynamics model derived from first principles which is then refined iteratively. Behaviourial targets are trajectory following with varying DoF and aerobatics. Faust et al. (2013) learn swing-free trajectories on load-carrying quadrotors

using Approximate Value Iteration (AVI) on top of similar models. The autonomous plane of Bry, Bachrach, and Roy (2012) also falls into this class.

Using attractors in dynamical systems has been proposed repeatedly for achieving computational function, an example are Attractor Neural Networks (ANNs) (Mussa-Ivaldi and Solla 2004). Ijspeert, Nakanishi, and Schaal (2002) introduced the concept of Dynamic Movement Primitives (DMP) which are explicitly constructed attractor systems which drive a second system via a nonlinear coupling that is generated with a set of weighted basis functions. The weight parameters can be optimized with Policy Search (PS) methods (Schaal et al. 2004; Kober, Wilhelm, et al. 2012). Tedrake et al. (2009) have used a similar approach for controlling a model plane around stall conditions. Sensory coupling of DMPs is not as straightforward however (Kober, Mohler, and Peters 2008).

Kolter and Ng (2009) and Lau 2011 integrate coarse models of the systems under consideration into their policy representations. The coarse model reflects the intuitive understanding of the system, in many cases the sign of the control input dependent derivatives suffices. Michels, Saxena, and Ng (2005) used a Markov Decision Process (MDP) model and the PEGASUS RL algorithm for learning vision-based obstacle avoidance on an RC car.

Rottmann et al. (2007) are bootstrapping an altitude controller using RL and Gaussian Process Regression (GPR) for approximating the state-action value function but special provision must be made to partition learning into chunks suitable for GPR. In their comprehensive work on learning autonomous helicopter aerobatic maneuvers, Abbeel and Ng (2004); Abbeel, Coates, and Ng (2010) use Inverse Reinforcement Learning (IRL) to optimize a coarse model through expert demonstrations and use that for learning aerobatic maneuvers on an autonomous helicopter. This is an incomplete sample of the literature, for a good survey of RL in robotics please refer to Kober and Peters (2012).

Finally, Waegeman, Wyffels, and Schrauwen 2012 propose a neural realization of motion primitives which are acquired through imitation using regression, which is a standard reservoir computing method (Jaeger 2001; Maass, Natschlaeger, and Markram 2002).

IV. Methods and model

A. Multirotor motion primitives

We choose to work on top of a standard attitude-stabilized controller which accepts $\boldsymbol{u}_{robot} = (u_1, u_2, u_3, u_4)^T$ as the time-dependent control input, where the u_i for i = {1,.., 4} correspond to roll, pitch, yaw and thrust. The time index t is omitted for clarity. Simply speaking, thrust effects a vertical acceleration, setting roll and pitch angles effects lateral accelerations in the plane. By integrating these actions, we attain velocity and position in space. We are searching for an alphabet of excitation patterns of $\boldsymbol{u}$ with which to synthesize acceleration controlled behaviour in the 6 DoF of a rigid body in $\mathbb{R}^3$.

B. Network model

We use neural reservoirs for representing the internal model from which controller output is derived. The reservoir is a recurrent neural network with random connectivity per instantiation. It can be used as a universal dynamical system approximator. The first important parameter of such a reservoir is its size, that is, the number of neurons N in the network. Increasing the network size N increases the general modelling capacity of the network.

We represent the Sensorimotor Primitive (SMP) with a leaky integrator reservoir and an output weight vector $\boldsymbol{W}_i^{\text{out}}$ associated with output channel i. These components model the actual physical system and can generate control and prediction outputs. The network state evolves according to

$$\triangle \boldsymbol{x}_t = \lambda \boldsymbol{W}^{\text{res}} \boldsymbol{r}_t + \boldsymbol{W}^{\text{in}} \boldsymbol{u}_t + \boldsymbol{W}^{\text{f}} \boldsymbol{z}_t + \boldsymbol{W}^{\text{bias}} 1$$

$$\boldsymbol{x}_{t+1} = (1-\tau)\boldsymbol{x}_t + \tau \triangle \boldsymbol{x}_t$$

$$\boldsymbol{r}_{t+1} = \tanh(\boldsymbol{x}_{t+1}) + \nu_{state}$$

The matrices $\boldsymbol{W}^{\text{res}}, \boldsymbol{W}^{\text{in}}, \boldsymbol{W}^{\text{f}}, \boldsymbol{W}^{\text{bias}}$ are the $N \times N$ reservoir, $N \times U$ input, $N \times Y$ readout feedback and $N \times 1$ bias matrices respectively. U and Y designate the number of inputs and readout units. The scalar λ is a scaling factor to effect a desired spectral radius for the reservoir matrix and in our experiments is chosen as $\lambda = g\frac{1}{\sqrt{pN}}$, where gain $g = 1.1$ and the connection probability for the reservoir matrix $p = 0.1$. The state noise ν_{state} is uniformly distributed with amplitude -0.1 to 0.1 and has a regularizing effect. Network output is computed as

$$y_{i,t} = \boldsymbol{W}_{i,t}^{\text{out}} \boldsymbol{r}_t$$

for each channel i at time t. The final policy can be formed according to

$$\boldsymbol{u}_{robot} = (y_1, y_2, y_3, y_4)^T$$

In the experiments we do not synthesize the full $\boldsymbol{u}_{robot}$ but only look at one or two readout units. The other controls are either assigned constant values or are driven by e.g. a PID controller.

C. Learning

There are several methods available for training reservoirs. We interrelate three different methods, learning rules, PS, and regression in Figure 1. In the bootstrapping case, we want use close to no prior information on how the control signal depends on the sensory input. The priors we do introduce in the process are limited to an offset for the thrust command to bridge the "dead band" in the lower half of the allowed range (the system is unresponsive), a time constant for the network dynamics, the size of the reservoir and an amplitude for the exploration noise ν. We use an unsupervised learning method implemented via a Hebbian learning rule (Hoerzer, Legenstein, and Maass 2012). The learning rule exploits correlations in the exploration noise ν and the immediate reward. During learning, Gaussian noise is added to the readout signal by letting

$$\hat{y}_{i,t} = \boldsymbol{W}^{\text{out}}\boldsymbol{r}_t + \nu_t = y_{i,t} + \nu_t$$

with $\nu \sim \mathcal{N}(\mu, \sigma)$. The reward is computed by a performance measure P_t at each time step. A low-pass filtered version $\bar{P}_t = \alpha P_{t-1} + (1-\alpha)P_t$ is also generated with $\alpha = 0.8$. The modulator signal is derived from P_t via

$$M_t = \begin{cases} 1 & \text{if } P_t > \bar{P}_t \\ 0 & \text{otherwise} \end{cases}$$

and the weight update then is given as

$$\Delta \boldsymbol{W}^{\text{out}}_{i,t} = \eta_{i,t}\boldsymbol{r}_t(y_{i,t} - \bar{y}_{i,t})M_t$$

,

with $\bar{y}_{i,t}$ being a low-pass filtered version of $y_{i,t}$ analogously to $\bar{P}$. The signals P_t and M_t indexed with motor channel i, but this is not strictly necessary. The entire situation is schematically depicted in Figure 1.

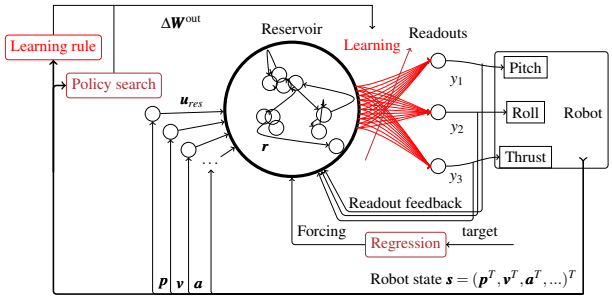

Fig. 1. Schematic of the learning system. Those components directly involved with learning in the proposed setup are colored in bright red, related methods in a darker shade. Application of the Hebbian learning rule can be regarded as policy search. PS methods emit $\Delta \boldsymbol{W}$ derived from online exploration, regression computes $\boldsymbol{W}$ directly from a batch.

D. Simulator

Simulations were done with the flight model simulator crrcsim [1], which was modified to be used as a software-in-the-loop drop in for the real helicopter. This allows using the same software infrastructure as on our real robots. The control loop is executed at 36 Hz.

V. Experiments

In these experiments we consider basic stabilizing behaviour for a multirotor helicopter, in particular hovering at fixed altitudes and keeping a fixed lateral position. What we aim to bootstrap is a policy for general cases, even when we might not have a clear a priori idea of the general direction of motor induced sensory consequences. These experiments were performed to investigate the suitability of the approach to application on a real robot.

A. Bootstrapping asymmetrical stabilization: hover

The most basic capability of a flying robot is, arguably, that of getting off the ground. In the simplest case, the goal state could be required to be inside a given band, that is, between a lower and an upper limit disregarding the specific altitude achieved by the controller. Nonetheless we set a singular setpoint as the target for learning.

For bootstrapping we apply exploratory Hebbian learning to the reservoir readout weights. This implements a policy gradient method via immediate

[1] `http://sourceforge.net/apps/mediawiki/crrcsim/index.php?title=Main_Page`

reward correlation with action. A crucial point before learning can take place is determining the dead-time or temporal delay T of the sensory consequences with regard to motor action. Here we bumped the system and determined the temporal delay with cross-correlation of the recorded reponses. The reward used in this experiment is $P_t = a_t \operatorname{sgn} e_t$ with acceleration a_t, positional error $e_t = p_d - p_t$, the desired position p_d and the sign of the error $\operatorname{sgn} e \in \{1, -1\}$. The performance measure rewards acceleration pointing in a direction reducing the error with a sharp inversion when crossing the setpoint. The delta rule needs to respect the temporal delay T so we used $\Delta \boldsymbol{W}^{\text{out}} = \eta(y_{t-T} - \bar{y}_{t-T}) M_t \boldsymbol{r}_{t-T}$.

Structurally, the reservoir receives $\boldsymbol{u}^{\text{res}} = (p, v, a)$ as input, where p, v, a are position, velocity and acceleration respectively with regard to the vertical axis. The readout is connected to the thrust motor channel and is also fed back into the reservoir as an efference copy. Before learning, the readout weights are initialized to zero as $\boldsymbol{W} = \boldsymbol{0}$ and after a short washout period of 300 time-steps, learning is started. Figure 2 shows an exemplary learning run in that configuration. After about three (real-time) minutes the system achieves ground clearance and after twice the amount of time achieves satisfying performance. Learning is performed with a constant learning rate η for about 550 seconds when learning is clamped and the system is put into testing mode. The testing episode is magnified in Figure 3. The spikes are strong perturbations achieved by halting controller execution for several seconds, propagating the last motor command emitted, and then resuming execution. The system returns to the operating point without overshoot. Figure 4 shows the first two principal components of the reservoir activation during another testing episode of a different controller instance with positive and negative perturbations around the setpoint.

What is special in this setup, apart from the obvious dynamical assymmetry is that in order to take off, the robot needs to move through Ground Effect (GE). GE alters the thrust to lift ratio, so that a thrust value which initially lifts the robot off the ground is insufficient for climbing once GE is left.

B. Results

We generated statistics over 100 learning runs to assess the robustness of the learning process itself. The learning duration has been set to a fixed amount, during which a major proportion of runs reach a satisfying level of performance. Those that do not, fall into two categories. In some cases the learning process picks up the wrong direction right at the beginning, resulting in negative thrust values

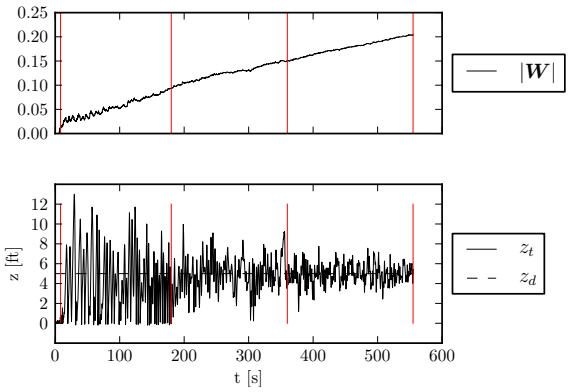

Fig. 2. Exemplary learning run for bootstrapping the hover controller. Learning starts at $t = 8.3$s and the robot makes its first jump in the air shortly after. The weight vector norm oscillates slightly in the beginning due to the invserion of the reward value when crossing the target setpoint but this behaviour subsequently ceases. At about 3 minutes the robot finally clears the ground and remains airborne for the rest of the episode, except for touching the ground once at $t = 300$. Until termination of learing at $t = 555$s, variance steadily decreases.

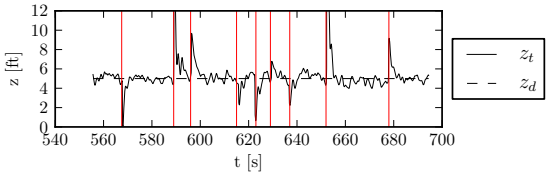

Fig. 3. The testing episode of the controller whose acquisition is shown in Figure 2. The system is perturbed at various points marked in red during the episode visible as large spikes in the curve. It returns robustly to baseline activity around the designated setpoint. The seemingly chaotic resting activity is induced by internal system noise.

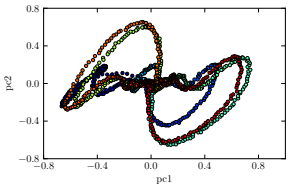

Fig. 4. First two principal components of the reservoir activation during hover testing. The system went through positive and negative perturbations of varying magnitude robustly returns to a stable fixpoint.

and making the system unresponsive. In the other cases, the reservoir locks into autonomous oscillations which the learning process fails to suppress and which dominate the output. See Figure 5 for examplary altitude profiles and RMSE for all 100 runs.

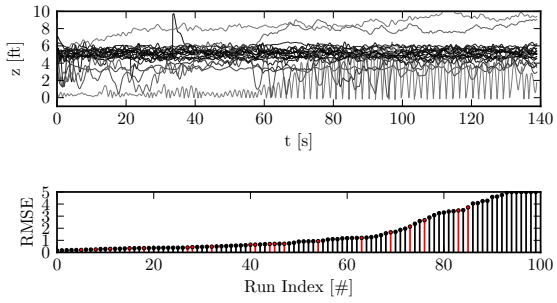

Fig. 5. Statistics over 100 learning runs. The upper graph shows a sample of superimposed temporal altitude profiles during testing of the converged controller. The lower graph shows the RMSE for the final testing interval for all runs. Those runs displayed in the upper graph are colored red in the lower one.

We ran the experiments with reservoirs of sizes $N = \{20, 100\}$ although we did not, at this point, generate comparative statistics. Not entirely surprising, the larger reservoir overall tends towards better performance in terms of convergence speed, final error and robustness. Apart from disturbances, we also tested injecting additional measurement noise in the altitude reading, clamping of inputs and setpoint modulation by adding an offset to the measurement. We applied the same method to learning lateral stabilization of the robot and also investigated episodic policy refinement using PS methods. Due to limited space these results cannot be further illustrated here. Other possible follow-up experiments from this general scenario are discussed in section VI.

VI. Discussion and future work

All the learning methods discussed here are only able to find local optima in the performance landscape. This is still sufficient for a variety of tasks. Of particular importance to us was the applicability of learning to real robots without the immediate danger of destroying the robot during exploration.

We think the learning process of the hover controller is remarkable in that bootstrapping occurs in a double sense. First, the controller itself has to be learnt but at the same time, the learning task is not stationary but the system needs to learn implicitly to first take off before it can stabilize itself at a given hover altitude requiring different magnitudes of changes to output signal.

The acquired primitives are innately grounded in that they are only referenced to locally measurable quantities. Using only the sign of the positional error besides acceleration in the reward puts weak demands on the visual sensing of position on the real robot. Compared with other approaches our method is very general in that almost no prior knowledge is needed for a successful learning run. It can readily be applied to a variety of vehicles in an online setting. At the same time, if prior knowledge can be given in the form of a demonstration, this could also be exploited.

Future work will involve verifying the approach on real robots (Berthold, Müller, and Hafner 2011) and autonomous determination of the inherent temporal delay. It has to be noted however, that using episodic policy search, the temporal delay issue is of no concern.

VII. Summary

We presented a neural architecture capable of representing sensorimotor primitives suitable for model-free reinforcement learning. We demonstrated the application of that architecture to problems in MAV motion control although we are not limited in scope to MAVs. We were able to bootstrap SMPs for stabilization tasks using online learning and exploiting sensorimotor interaction. The bootstraping process is gentle enough to not destroy the system during exploration and thus can be applied to learning on real hardware. In addition, the same process can be applied to post-bootstrapping adaptation of the acquired primitives to changes in the environment. This enables implementation of continuously learning system. The resulting controllers display anticipatory behaviour and are extremely robust to noise which make them well suited for use in combination with vision-based state estimates. Supplementary material includes python scripts for learning of stabilizing an acceleration controlled point mass in two dimensions, thus illustrating the principles.

VIII. Acknowledgements

This work is supported by the DFG Research Training Group METRIK. We are grateful for inspiring discussions to Christian Blum and Damien Drix.

References

Abbeel, Pieter, Adam Coates, and Andrew Y. Ng (2010). "Autonomous Helicopter Aerobatics through Apprenticeship Learning". In: *I. J. Rob. Res.* 29.13, pp. 1608–1639.

Abbeel, Pieter and Andrew Y. Ng (2004). "Apprenticeship learning via inverse reinforcement learning". In: *Proc. ICML*. ACM, pp. 1–.

Arbib, Michael A. (1981). "Supplement 2: Handbook of Physiology, The Nervous System, Motor Control". In: APS/Wiley. Chap. Perceptual Structures and Distributed Motor Control, pp. 1449–1480.

Berthold, Oswald, Mathias Müller, and Verena V. Hafner (2011). "A quadrotor platform for bio-inspired navigation experiments". In: *International workshop on bio-inspired robots*.

Bry, Adam, Abraham Bachrach, and Nicholas Roy (2012). "State estimation for aggressive flight in GPS-denied environments using onboard sensing". In: *ICRA*, pp. 1–8.

Faust, Aleksandra et al. (2013). *Learning Swing-free Trajectories for UAVs with a Suspended Load in Obstacle-free Environments*. ICRA WS AL.

Grillner, Sten (2006). "Biological Pattern Generation: The Cellular and Computational Logic of Networks in Motion". In: *Neuron* 52.5, pp. 751 –766.

Hoerzer, Gregor M., Robert Legenstein, and Wolfgang Maass (2012). "Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning". In: *Cerebral Cortex*.

Ijspeert, Auke Jan (2008). "Central pattern generators for locomotion control in animals and robots: A review". In: *Neural Networks* 21.4, pp. 642 –653.

Ijspeert, Auke Jan, Jun Nakanishi, and Stefan Schaal (2002). "Learning Attractor Landscapes for Learning Motor Primitives". In: *Proc. NIPS*. MIT Press, pp. 1547–1554.

Jaeger, Herbert (2001). *The ''echo state'' approach to analysing and training recurrent neural networks - with an Erratum note*. Tech. rep. German National Research Center for Information Technology.

Kober, Jens, B. Mohler, and Jan Peters (2008). "Learning perceptual coupling for motor primitives". In: *Proc. IROS*, pp. 834–839.

Kober, Jens and Jan Peters (2012). "Reinforcement Learning in Robotics: A Survey". In: *Reinforcement Learning*. Springer Berlin Heidelberg, pp. 579–610.

Kober, Jens, Andreas Wilhelm, et al. (2012). "Reinforcement learning to adjust parametrized motor primitives to new situations". In: *Autonomous Robots* 33.4, pp. 361–379. ISSN: 0929-5593.

Kolter, J. Zico and Andrew Y. Ng (2009). "Policy search via the signed derivative." In: *Proc. RSS*.

Lau, Tak-Kit (2011). "PRISCA: A policy search method for extreme trajectory following". In: *Proc. CDC-ECC*, pp. 2856–2862.

Lupashin, S. et al. (2010). "A simple learning strategy for high-speed quadrocopter multi-flips". In: *Proc. ICRA*, pp. 1642–1648.

Maass, Wolfgang, Thomas Natschlaeger, and Henry Markram (2002). "Real-time Computing without stable states: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14.11, 2531–2560.

Mellinger, Daniel, Nathan Michael, and Vijay Kumar (2012). "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *I. J. Rob. Res.* 31.5, pp. 664–674.

Michels, Jeff, Ashutosh Saxena, and Andrew Y. Ng (2005). "High speed obstacle avoidance using monocular vision and reinforcement learning". In: *Proc. ICML*, pp. 593–600.

Mussa-Ivaldi, F. A. and S. A. Solla (2004). "Neural Primitives for Motion Control". In: *IEEE Journal of Oceanic Engineering* 29, pp. 640+.

Rottmann, Axel et al. (2007). "Autonomous blimp control using model-free reinforcement learning in a continuous state and action space". In: *Proc. IROS*.

Schaal, Stefan et al. (2004). "Learning movement primitives". In: *Proc. ISRR*. Springer.

Schoellig, Angela P., Fabian L. Mueller, and Raffaello D'Andrea (2012). "Optimization-based iterative learning for precise quadrocopter trajectory tracking". In: *Autonomous Robots* 33.1-2, pp. 103–127.

Tedrake, Russ et al. (2009). *Learning to Fly like a Bird*.

Waegeman, Tim, Francis Wyffels, and Benjamin Schrauwen (2012). "A discrete/rhythmic pattern generating RNN". eng. In: *Proc. ESANN*, pp. 567–572.