Autonomous Blimp Control using Model-free Reinforcement Learning in a Continuous State and Action Space

Axel Rottmann* Christian Plagemann* Peter Hilgers† Wolfram Burgard*

Abstract— **In this paper, we present an approach that applies the reinforcement learning principle to the problem of learning height control policies for aerial blimps. In contrast to previous approaches, our method does not require sophisticated hand-tuned models, but rather learns the policy online, which makes the system easily adaptable to changing conditions. The blimp we apply our approach to is a small-scale vehicle equipped with an ultrasound sensor that measures its elevation relative to the ground. The major problem in the context of learning control policies lies in the high-dimensional state-action space that needs to be explored in order to identify the values of all state-action pairs. In this paper, we propose a solution to learning continuous control policies based on the Gaussian process model. In practical experiments carried out on a real robot we demonstrate that the system is able to learn a policy online within a few minutes only.**

I. Introduction

Compared to other flying vehicles, aerial blimps have the advantage that they operate at relatively low speed, that they do not need to move in order to keep their altitude, and additionally are not overly sensitive to control errors like, e.g., helicopters. In this paper, we investigate the problem of learning to control the height of an autonomous blimp online and without pre-defined physical models. The goal of this work is to allow the blimp to autonomously learn the actions necessary to maintain its height from scratch after it has been switched on. We have several demands to such a learning approach:

1) It should be able to learn the control policy directly on the blimp, i.e., without the need for simulation or human intervention,
2) it should be able to learn the policy within a few minutes on the real blimp, and
3) it should be able to deal with the continuous state-action space given by the current estimate for the height and the vertical velocity.

The first requirement assures that the blimp adapts its behavior to the current properties of the environment and its own dynamics such as the amount of helium contained in the envelope. The second property is necessary to ensure that the blimp can actually learn new policies from scratch whenever needed and still being able to fulfill its mission. The third requirement is important to deal with the continuity of the underlying state-action spaces.

Our approach is based on the reinforcement learning principle, i.e., the goal to be achieved by the controller is specified by virtual rewards given to the system when certain system states are reached. Our approach employs a Kalman filter to estimate the ground clearance based on noisy distance measurements obtained from an ultrasound range sensor. Based on these height estimates, we apply Monte Carlo reinforcement learning in combination with Gaussian processes to represent the Q-function over the continuous state-action space. To evaluate our approach, we implemented it on the blimp depicted in Figure 1. Experimental results demonstrate that our approach can quickly learn a policy that shows the same performance as a manually tuned PD^2-T_2 controller.

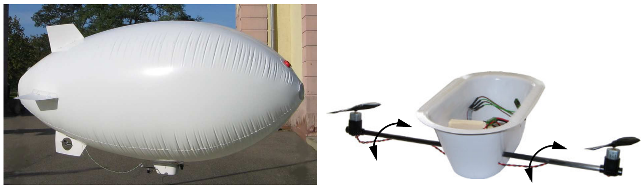

Fig. 1. Blimp used to evaluate our approach (left image). The right image shows the gondola with two rotors for pitch and thrust control. They can be rotated by 180 degrees.

The paper is organized as follows. After discussing related work in the following section, we will describe the properties of our blimp in Section III. In Section IV we will then introduce our approach to learning the control policy of the blimp. After that, we will present how to learn such a policy online on the blimp in Section V. Finally, we will present our experimental results obtained on a real robot and in simulation in Section VI.

II. Related Work

The problem of controlling a blimp has been studied intensively in the past. For example, Varella Gomes and Ramos [19] as well as Hygounenc *et al.* [8] describe the physical principles of airship operations and use the non-linear dynamics to control several flight phases from takeoff to landing. Zhang and Ostrowski [21] use a vision-guided blimp combined with a PID controller. Fukao *et al.* [6] discuss image-based tracking control for an indoor blimp. Wyeth and Barron [20] present a low level reactive controller and Rao *et al.* [12] use a fuzzy controller. Compared to these approaches, our algorithm does not assume prior knowledge about the dynamics or pre-defined controllers. Instead, it learns the control policy online and does not require a-priori information about the payload, the temperature, or the air pressure. Furthermore, autonomous

*University of Freiburg, Department for Computer Science, D-79110 Freiburg. †University of Freiburg, Department of Microsystems Engineering, D-79110 Freiburg

blimps have been presented in different application scenarios, for instance by Jung and Lacroix [10], Fukao *et al.* [5], and Green *et al.* [7].

The approach that is most closely related to the one described in this paper has recently been presented by Ko *et al.* [11]. They also deal with the problem of learning to control an autonomous blimp and choose a similar set of methods for this task. In contrast to our approach, however, they make extensive use of models of the dynamics, both for describing an analytical motion model as well as for the controller itself. Gaussian processes are used to learn the residuals of these models in a supervised manner using motion capture data as ground truth. In contrast to their work, we aim at learning the control policy directly, without representing an explicit motion model or based on simulation. Using only the real hardware as source of information about the dynamics, our system learns a non-parametric representation for the Q-Function while the system is flying.

Additionally, Gaussian processes have been applied to the reinforcement learning problem in various ways. Engel *et al.* [2] approached the problem from the viewpoint of temporal difference learning (GPTD) and later extended this scheme to be able to deal with stochastic state transitions to improve action selection and to learning Q-values without an explicit transition model (GPSARSA) [3]. Their approach was successfully applied to the problem of learning complex manipulation policies [4]. Continuous state spaces have also been considered by Smart and Kaelbling [17]. Rasmussen and Kuss [13] present a different approach, in which the system dynamics and the value function are learned using separate Gaussian processes. In contrast to their work, we show that our approach can be directly applied on a real robot and be utilized to learn a policy online.

III. The Autonomous Blimp

The approach presented in this paper has been targeted for the system depicted in the left image of Figure 1. It is based on a commercial $1.8\,\mathrm{m}$ blimp envelope, which is described in detail in [16]. The blimp is steered by three motors. One motor is mounted in the tail fin to control the yaw. The other two are mounted on each side of the gondola to control the pitch and thrust. The latter two are also attached to a shaft which can be rotated up to 180 degree by a servo (see right image of Figure 1). The gondola includes an Intel XScale PXA270 based system-on-a-chip with 600 MHz and 128 MB RAM. An on-board 32 MB flash memory serves as storage for the Linux operating system. This board is used to control the speed of the motors, to change the position of the servo, and to process the sensor data. For our experiments we used a downward-facing ultrasound sensor at the bottom of the gondola. The measurements of this sensor are integrated on-board by a Kalman filter which sequentially estimates the height and the vertical velocity of the vehicle. The total weight of the gondola and hardware is about 200 grams and the complete system is powered via a 3.7 V/1500 mAh lithium polymer battery.

IV. Reinforcement Learning

Reinforcement learning is based on the idea that an agent interacts with a potentially unknown environment and gets rewarded or penalized according to the actions it performs. In general, the agent receives rewards for actions that are beneficial in certain states for achieving a long-term goal. The agent thus seeks to behave in a way that maximizes its numerical reward. A reinforcement learning task can be defined by a tuple $\{S, A, \delta, r\}$, consisting of states S, actions A, a transition function $\delta : S \times A \to S$, and a reward function $r : S \times A \to \mathbb{R}$, which defines the immediate reward to be received when executing action a in state s. The goal is to determine a policy function $\pi : S \to A$, which maps each state s to an action a, such that the future expected reward is maximized. As described by Sutton and Barto [18], the expected long time reward in state s_t can be expressed recursively as

$$R^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i} , \tag{1}$$

where $\gamma \in (0, 1]$ is a discount factor. In general, the sequence of rewards r_{t+i} is obtained by starting in state s_t and then iteratively applying a policy π for selecting the next action $a_t = \pi(s_t)$.

A. On-Policy Monte Carlo Control

Several approaches have been proposed to solve the reinforcement learning problem by maximizing the expected long time reward as stated in (1). In this paper we apply the Monte Carlo (MC) method, which has the advantage that it allows the agent to learn directly from experience while interacting online with a completely unknown environment. This enables us to learn without prior knowledge and also in situations in which no simulation environment is available. Formally, we seek to learn the state-action value function $Q(s, a) : S \times A \to \mathbb{R}$, representing the expected future reward when selecting action a in state s. To gather training data, we iteratively generate episodes $e_1, \ldots, e_T$. Each episode $e_t = \big((s_1, a_1), \ldots, (s_N, a_N)\big)$ consists of state-action pairs, which are in turn selected using a given policy. An episode ends when a pre-defined goal state is reached, which typically yields maximum reward or a maximum episode length is exceeded. After the execution of an episode, the Q-values are derived by averaging over the expected long time rewards

$$Q(s, a) = \overline{R(s, a)} , \tag{2}$$

where $\overline{R(s, a)}$ is the average long time reward if action a is executed in state s. In this scheme, the best policy is given by the maximum over the Q-value function

$$\pi(s) = \arg\max_a Q(s, a) . \tag{3}$$

A commonly applied strategy for generating the training episodes $e_1, \ldots, e_N$ is the so-called ϵ-greedy approach. Under this scheme one chooses a random action with probability $\epsilon \in (0, 1]$ and, otherwise, selects the currently best action as defined by (3).

B. Learning the Q-Function

A crucial choice for any learning task is the type of representation that is to be used for the core concepts. In the reinforcement learning task outlined above, we seek to learn the Q-function from sampled action sequences. Most existing approaches to the problem represent the Q-function using a discrete approximation over the space of state-action pairs. This, however, can lead to discretization errors or, when fine-grained grids are used, requires a huge amount of memory and a time-consuming exploration process. Therefore, function approximation techniques that directly operate on the continuous space such as neural networks [1], [15], kernel methods [9], or Gaussian processes [13], [3] have been proposed as powerful alternatives to the discrete approximations of the continuous Q-function. From a regression perspective, these techniques seek to model the dependency

$$q_i = Q(s_i, a_i) + \epsilon_i \tag{4}$$

for the unknown function Q and independent and identically, normally distributed noise terms ϵ_i, given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, q_i)\}_{i=1}^{D}$ of state-action pairs $\boldsymbol{x}_i = (s_i, a_i) \in S \times A$ and estimates of their Q-value q_i.

Gaussian processes can be seen as a generalization of weighted nearest neighbor regression [14], where the dependency of a function value on its local neighborhood is described using parameterized covariance functions. We imply in this work the squared exponential

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sigma_f^2 \exp\left(-\sum_{j=1}^{d} \frac{(x_j - y_j)^2}{2\ell_j^2}\right), \tag{5}$$

where ℓ is the length-scale and σ_f the signal variance. In our model, the inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ with $\boldsymbol{x}, \boldsymbol{y} \in S \times A$ are d-dimensional vectors, which are constructed by concatenating the corresponding state and action vectors. Given a set of data samples $\mathcal{D}$ from the Q-function and the hyper-parameters $\boldsymbol{\theta} = (\sigma_f, \ell_1, \ldots, \ell_d)$ of the covariance function, arbitrary Q-values q^* can be predicted for a new input location $\boldsymbol{x}^*$ by

$$q^* = \boldsymbol{k}^* \left(K + \sigma_n^2 I\right)^{-1} \boldsymbol{q}, \tag{6}$$

where $K \in \mathbb{R}^{D \times D}$ is the covariance matrix for the training points, $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{k}^* \in \mathbb{R}^D$, represents the covariances between training points and the query point $\boldsymbol{x}^*$, $k_i^* = k(\boldsymbol{x}^*, \boldsymbol{x}_i)$, and $\boldsymbol{q} = (q_1, \ldots, q_D)$ represents the Q-values from the training set. For details about Gaussian process regression, we refer to the book of Rasmussen and Williams [14] or the numerous other introductory works available. The learning task consists of two parts, i.e., the gathering of training data $\mathcal{D}$, which has been described in the previous section, and the adaptation of the hyper-parameters $\boldsymbol{\theta}$ of the covariance function as well as the global noise parameter σ_n. In the Gaussian process framework, these parameters can be optimized by maximizing the marginal data likelihood of the observed training data $\mathcal{D}$. In our current system we optimize these parameters using gradient descent.

TABLE I

MONTE CARLO LEARNING USING GAUSSIAN PROCESSES TO APPROXIMATE THE Q-FUNCTION.

- Input: $a_{\text{min}}, a_{\text{max}}, \epsilon, \gamma$
- Repeat until convergence:
    1) Generate an episode $(s_1, a_1), \ldots, (s_N, a_N)$ of state-action pairs using the ϵ-greedy policy.
    2) For each pair (s_n, a_n) appearing in the episode calculate the expected long time reward

       $$R(s_n, a_n) = \sum_{k=0}^{N-n} \gamma^k r_{s_{n+k}, a_{n+k}},$$

       where $\gamma \in (0, 1]$ and r represents the immediate reward.
    3) Add all $R(s_n, a_n)$ to the training set $\mathcal{D}$ of the Gaussian process GP and optimize the hyper-parameters $\boldsymbol{\theta}$ and σ_n.
- The best policy π for a state s' is given by

  $$\pi(s') = \arg\max_a GP(s', a),$$

  where $a \in [a_{\text{min}}, \ldots, a_{\text{max}}]$.

To avoid the blimp having to wait until the parameters have been updated, we realized a thread-based implementation which performs the optimization in parallel to the learning task. In the simulation experiments, however, the optimization is carried out after each round of 100 episodes.

V. ONLINE POLICY LEARNING ON A REAL BLIMP

For the task of learning online to control an autonomous blimp, several specific constraints have to be met. First, we need to specify the reward function, which implicitly defines how we want the blimp to behave and react. Second, we need to define how the episodes are created during continuous online learning without restarting the blimp. Finally, we need to define the state and action space and need to explicitely constrain the state space to limit exploration.

We consider the task of stabilizing the blimp at a given altitude h^* without knowing the specific dynamics or any parameters of the environment. In practice, altitude control is already a complex task as blimps are very sensitive and their behavior highly depends on outer influences such as payload, battery level, temperature, and air flow. In general, it is hard to determine a globally suitable policy applicable to several conditions. Therfore, we seek to learn the best policy for the current conditions while the blimp is in operation. For this task, we propose to use the on-policy MC approach to reinforcement learning. To deal with the continuous Q-function, we apply Gaussian processes as approximation technique. The resulting algorithm is outlined in Table I.

We define the state space S by $s_t = (d_t, v_t)$, where $d_t = h_t - h^*$ represents the distance to the goal altitude h^*. The terms h_t and v_t are the estimated height and velocity values calculated by the Kalman filter. The velocity v is negative, if the blimp descends. The action space A is naturally limited by the capabilities of the motors, i.e., by an interval $a_t \in [a_{\text{min}}, a_{\text{max}}]$.

To apply MC learning to the blimp, we need to specify how the episodes are created. In our application, we have to sample the episodes $e_1, \ldots, e_N$ from a sequence of measurements $(s_1, a_1), \ldots (s_T, a_T)$ obtained while the blimp is moving trough the environment. For each state-action pair (s_t, a_t) in the sequence an episode $e_t = \big((s_t, a_t), \ldots, (s_{t+p}, a_{t+p})\big)$ is generated consisting of the p successor states. The length p of an episode is defined by the factor γ. An episode ends if the factor γ^p is smaller than a given threshold θ which has been set to 0.1 in our current implementation. The expected long time reward for each state-action pair is finally given by

$$R^\pi(s_t, a_t) \quad = \quad \sum_{i=0}^{p} \gamma^i r_{t+i} \, . \tag{7}$$

To guarantee convergence to the optimal policy, the learning task would either have to be endlessly restarted at random initial states to assure that all state-action pairs are visited, or, like in our work, a continuous policy would have to ensure that all state-action pairs are visited with a non-zero probability.

An additional condition induced by continual movement of the blimp is the limitation of the exploration space. In our case, in which the blimp is never restarted in well-defined starting states, the system may move outside of the safe working environment and, for example, hit the floor or ceiling. Our experiments were performed in a factory building with a ceiling height of 5 m. We avoid collisions, by artificially guaranteeing that the selected action a_t satisfies the equation $d_t \cdot a_t < 0.0$ if the absolute value of the current distance d_t is greater than $1.5\,$m.

Finally, to stabilize the blimp at a given altitude h^*, we have to specify the conditions of the learning task that the agent reaches the goal when it tries to maximize the expected long time reward. A common strategy is a boolean reward function, means, if the goal is reached r^+ is received otherwise r^-, where $r^+ > r^-$. In our specific task, we found that such a boolean reward function requires a longer learning phase than a reward function that is proportional to the distance to the goal

$$r(s_t, a_t) \quad = \quad -|d_t|. \tag{8}$$

A common method to learn the optimal policy is the so called actor-critic [18] which separates the policy from the value function. One structure is used to select the action whereas the other represent the actual learned progress. This guarantees that the policy is fixed while learning the Q-values. In our current system, we use one structure to simultaneously represent the values and the current ϵ-greedy policy. In experiments not reported here, we compared these two methods and achieved no significant difference in the resulting policies.

VI. Experiments

The approach described above has been implemented and tested in simulation as well as on a real robot. The goal of the experiments is to demonstrate that our approach learns to control a real blimp online and within a short time-frame without any prior information about the dynamics of the vehicle. Furthermore, we describe results indicating the improvement obtained by utilizing Gaussian processes for function approximation without loosing precision. Finally, we compare the behavior of our policy learned online with a manually tuned PD^2-T_2 controller.

The real-world experiments described in this section have been conducted with the small-scale blimp described in Section III. The challenge is to control the altitude without knowing the specific dynamics of the blimp or any parameters of the environment. In practice, height control already is a complex task as blimps are very sensitive to outer influences like payload or air flow. Additionally, it has to be taken into account that the resolution of the ultrasound sensor is 1 cm and since the velocity is very small the estimation is susceptible to errors. Therefore, the estimated state of the blimp is extremely noisy and accordingly the results of potential actions can hardly be predicted.

To evaluate the progress of the learning processes we derived the dynamics of the blimp based on standard physical aeronautic principles [19]. Since we only took the movement in the vertical direction into account, the effects to all other dimensions were ignored. Afterwards, the produced forces of the motors depending on the control signals were determined on a measuring system and the parameters of the dynamics were optimized based on several test flights. We applied the resulting dynamics to move the blimp in simulation as well as to determine a policy we can use as a base-line in our experiments. This policy was computed using dynamic programming [18] with a transition function based on the dynamics. Please note, that this policy is optimal with respect to the dynamics, but not in general for all conditions. As can be seen from the experiments, this policy is well-suited to evaluate the learning progress.

In all experiments only the two motors mounted at the gondola were used. They were oriented upwards to control the altitude and the maximum values a_{min} and a_{max} for the actions were set to -0.8 and 0.8 respectively. These values represent the engine speed in percent. We also analyzed different time intervals between the processed measurements and obtained the best results for a time delay of 1 second. Whereas shorter intervals made it nearly impossible to predict the effects of an action, larger intervals complicated the learning in environments of limited height like our factory building as the blimp typically quickly reaches the maximum height or gets too close to the floor.

A. Simulated Learning

In the first experiments we analyzed whether the integration of a Gaussian process for function approximation yields an improvement of the learning process. To perform this experiment, we simulated the movement of the blimp based on the dynamics with additional noise and learned control polices using the standard MC approach and our MC approach with Gaussian processes. Although the time-complexity of

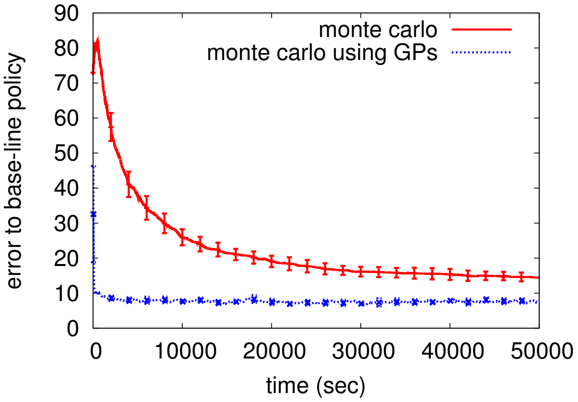

Fig. 2. The sum of squared differences between the present best actions and the base-line policy for the standard Monte Carlo learning approach and our approach using Gaussian processes.

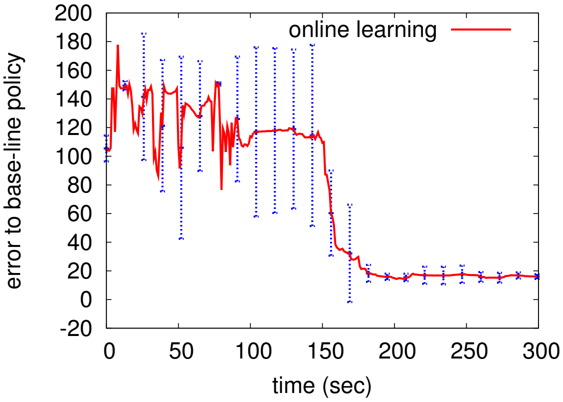

Fig. 3. Learning rate of our Monte Carlo learning approach using Gaussian processes applied online on the real blimp.

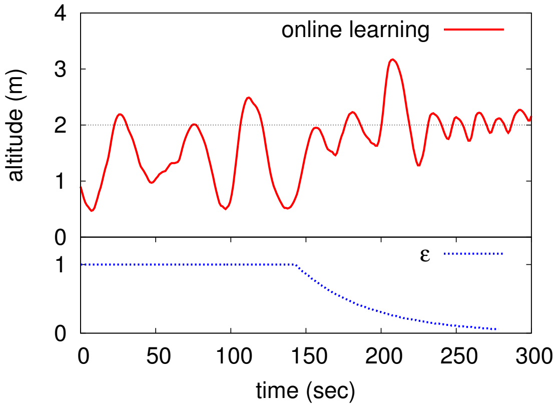

Fig. 4. The upper graph shows the altitude of the blimp while it learns to stabilize at 2 m. The lower graph plots the schedule of the ϵ-greedy policy.

inverting the covariance matrix in Gaussian processes can be speed up by the fact that they are symmetric and positive definite, the number of samples $\mathcal{D}$ has to be limited in the case of long runs. Therefore, we average two samples if their euclidian distance is smaller than a given threshold of 0.01. Thus, we follow the idea of MC but still adhere the property of continuous spaces. Incidentally, for short-term runs the reduction of the samples is needless since the calculation of the inverse matrix with $1,000$ samples requires less than one second.

Figure 2 plots the sum of the squared differences to the base-line policy averaged over multiple runs. As can be seen from the figure, our approach using Gaussian processes for approximating the Q-function converges significantly ($\alpha = 0.05$) faster and also provides a significantly better policy up to $50,000$ seconds. This illustrates that the approximation provided by the Gaussian process is extremely beneficial when only a few states have been visited. Furthermore, it also yields a good performance in the long-term without sacrificing precision despite the reduction of the sample space. Due to space restrictions we only plot the graph up to time-step $50,000$. Note that even up to $1,000,000$ time-steps the error of the standard MC approach never fell below the error obtained with our method.

B. Learning Online on a Real Blimp

This experiment is designed to demonstrate that our MC approach to reinforcement learning using Gaussian processes for approximating the Q-function can be applied online on a real blimp. It also illustrates that our approach allows us to efficiently learn on a completely model-free, real system with unknown dynamics. To perform this experiment we ran the blimp in a factory building with a vertical exploration space of 5 m. Figure 3 illustrates the average learning rate over multiple runs compared to the base-line policy calculated from the blimp dynamics. As can be seen from the graph, our approach is able to learn a good policy already after only a few states have been visited. Note, that every second only one new state is inspected. At the beginning, the learning rate is very instable which is due to the fact that the low number of samples typically introduce a high variance in the Gaussian process. As the number of samples increases, the learning rate stabilizes and converges quickly to an accurate estimate, which is close to the base-line policy.

Figure 4 shows a typical evolution of the height of the vehicle and the schedule of the ϵ-greedy policy for one learning experiment. As can be seen, at the beginning the blimp is exploring the states and with descending ϵ the blimp is gradually exploiting the current policy and finally stabilizing at the goal of 2 m. Figure 5 depicts the final policy learned during this run, which lasted 300 seconds.

C. Comparison to a Manually Tuned $\text{PD}^2\text{-T}_2$ Controller

The final experiment is designed to illustrate the benefit of online learning on real systems compared to a manually tuned $\text{PD}^2\text{-T}_2$ controller, which is a PD controller with two D terms and two delay elements. This $\text{PD}^2\text{-T}_2$ has been developed based on the dynamics and characteristics of our blimp system and the parameters of the controller have been optimized in simulation. Additionally, we integrated a virtual PWM module to deal with the dead-zone of the non-linear actor function. Figure 6 shows the behavior of the blimp using the $\text{PD}^2\text{-T}_2$ and the controller learned with our

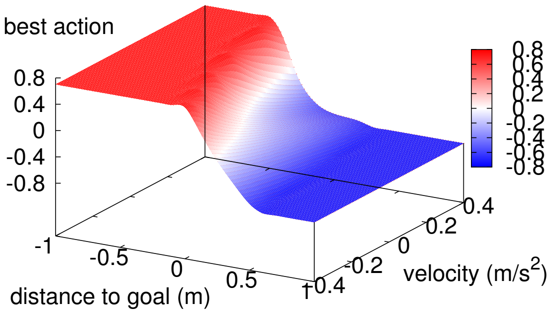

Fig. 5. Policy learned online on the real blimp with our Monte Carlo learning approach using Gaussian processes within three minutes.

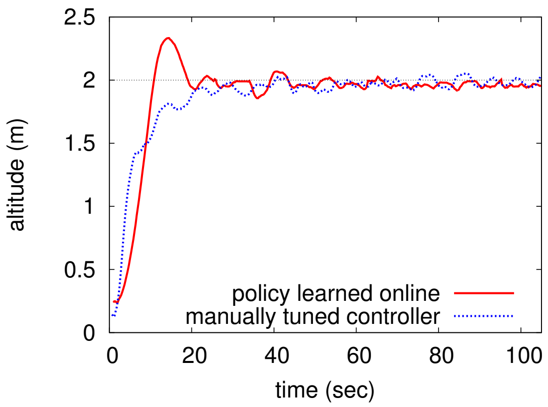

Fig. 6. Altitude progress obtained by applying the online learned policy and the policy of the manually tuned controller while the blimp stabilizes at 2 m.

approach over a time period of three minutes. As can be seen, the policy learned online stabilizes the blimp as efficient as the PD^2-T_2 controller and yields a similar behavior. It can also be observed that the controller learned online is not penalized by the reward function for overshooting and thus behaves different to approach the goal.

VII. Conclusions and Further Work

In this paper we presented an approach to control the height of a blimp using reinforcement learning. Our approach applies Monte Carlo reinforcement learning and utilizes Gaussian processes for dealing with the continuous state-action space. Our approach does not require any pre-defined models about the dynamics of the blimp and is able to learn a policy online on a real blimp. As a result, our approach can be applied whenever the blimp is switched on and does not require any tuning of parameters whenever internal or external conditions change.

Our approach has been implemented and evaluated on a real blimp and has been compared to policies learned based on prior information of the system dynamics. Experimental results demonstrate that our approach can quickly learn an effective policy and also performs equally well as a manually tuned PD^2-T_2 controller.

Acknowledgment

This work has partly been supported by the DFG within the Research Training Group 1103, by the EC under contract number FP6-IST-34120-muFly (action line: 2.5.2.: micro/nano based subsystems) and BP6-004250-CoSy, and by the German Ministry for Education and Research (BMBF) through the DESIRE project.

References

[1] R. Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control.* PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[2] Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian Process approach to temporal difference learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2003.

[3] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2005.

[4] Y. Engel, P Szabo, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[5] T. Fukao, K. Fujitani, and T. Kanade. An autonomous blimp for a surveillance system. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[6] T. Fukao, K. Fujitani, and T. Kanade. Image-based tracking control of a blimp. In *Proc. of the IEEE Conf. on Decision and Control*, 2003.

[7] W. Green, K. Sevcik, and P. Oh. A competition to identify key challenges for unmanned aerial robots in near-earth environments. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, 2005.

[8] E. Hygounenc, I-K. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project at LAAS/CNRS: Achievements in flight control and terrain mapping. In *International Journal of Robotics Research*, 2003.

[9] N. Jong and P. Stone. Kernel-based models for reinforcement learning in continuous state spaces. In *ICML workshop on Kernel Machines and Reinforcement Learning*, June 2006.

[10] I-K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Int. Conf. on Computer Vision*, 2003.

[11] J. Ko, D. Klein, D. Fox, and D. Hähnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[12] J. Rao, Z. Gong, J. Luo, and S. Xie. A flight control and navigation system of a small size unmanned airship. In *Proc. of the IEEE Int. Conf. Mechatronics and Automation*, 2005.

[13] C. E. Rasmussen and E. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[14] C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[15] M. Riedmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proc. of the European Conference on Machine Learning (ECML)*, 2005.

[16] A. Rottmann, S. Sippel, T. Zitterell, W. Burgard, L. Reindl, and C. Scholl. Towards an experimental autonomous blimp platform. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2007.

[17] W. Smart and L. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2000.

[18] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[19] S. Varella Gomes and J. Ramos. Airship dynamic modeling for autonomous operation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[20] G. Wyeth and I. Barron. An autonomous blimp. In *Proc. of the IEEE Int. Conf. on Field and Service Robotics (FSR)*, 1997.

[21] H. Zhang and J. Ostrowski. Visual servoing with dynamisc: Control of an unmanned blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.