Auton Robot (2010) 29: 137–149
DOI 10.1007/s10514-010-9190-3

# Efficient vision-based navigation

## Learning about the influence of motion blur

**Armin Hornung · Maren Bennewitz · Hauke Strasdat**

Received: 18 June 2009 / Accepted: 7 April 2010 / Published online: 23 April 2010


**Abstract** In this article, we present a novel approach to learning efficient navigation policies for mobile robots that use visual features for localization. As fast movements of a mobile robot typically introduce inherent motion blur in the acquired images, the uncertainty of the robot about its pose increases in such situations. As a result, it cannot be ensured anymore that a navigation task can be executed efficiently since the robot's pose estimate might not correspond to its true location. We present a reinforcement learning approach to determine a navigation policy to reach the destination reliably and, at the same time, as fast as possible. Using our technique, the robot learns to trade off velocity against localization accuracy and implicitly takes the impact of motion blur on observations into account. We furthermore developed a method to compress the learned policy via a clustering approach. In this way, the size of the policy representation is significantly reduced, which is especially desirable in the context of memory-constrained systems. Extensive simulated and real-world experiments carried out with two different robots demonstrate that our learned policy significantly outperforms policies using a constant velocity and more advanced heuristics. We furthermore show that the policy is generally applicable to different indoor and outdoor scenarios with varying landmark densities as well as to navigation tasks of different complexity.

**Keywords** Navigation · Reinforcement learning · Vision · Motion blur

## 1 Introduction

Completing navigation tasks reliably and efficiently is one of the most essential objectives for an autonomous robot. As a precondition for finding the way to a target location, the robot needs to know its pose in the environment. Especially in the case of small robots with a limited payload, such as humanoids or unmanned aerial vehicles, compact and lightweight cameras are often used as the only sensor. However, the movements of a mobile robot typically introduce motion blur in the acquired images, with the amount of degradation depending on camera quality, on the lighting conditions, and on the movement velocity. To illustrate this, typical images of a floor patch observed with a downward-looking camera on a wheeled mobile robot moving at different speeds are depicted in Fig. 1. With an increasing velocity the image is

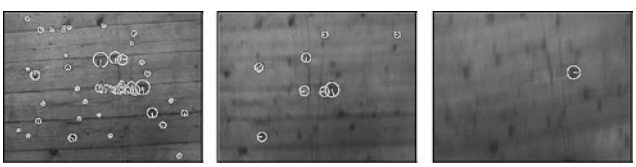


**Fig. 1** Visual features observed in the same floor patch at different velocities (0.05 m/s, 0.4 m/s, 1.0 m/s), with motion blur of increasing magnitude

This work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8.

**Electronic supplementary material** The online version of this article (doi:10.1007/s10514-010-9190-3) contains supplementary material, which is available to authorized users.

A. Hornung (✉) · M. Bennewitz
Department of Computer Science, University of Freiburg,
Georges-Koehler-Allee 79, 79110 Freiburg, Germany
e-mail: hornunga@informatik.uni-freiburg.de

M. Bennewitz
e-mail: maren@informatik.uni-freiburg.de

H. Strasdat
Department of Computing, Imperial College London,
180 Queen's Gate, South Kensington Campus,
London SW7 2AZ, UK
e-mail: strasdat@doc.ic.ac.uk

highly affected by motion blur. As a result, the reliability of feature detection and matching and, thus, the localization accuracy typically decreases. An uncertain localization may result in a wrong pose estimate which, in turn, may prevent the robot from executing the navigation task reliably and efficiently.

In this article, we present a novel approach to vision-based navigation that implicitly takes the influence of motion blur, which results from fast movements, into account. Our method applies reinforcement learning (RL) to determine actions the robot should execute to reach its destination fast and reliably. We use an unscented Kalman filter to track the pose of the robot and include the uncertainty of the state estimate into the state representation of the augmented Markov decision process (MDP) modeling the navigation task. The state representation comprises further features such as the estimated distance and the estimated relative angle of the robot to the goal location. The action space of the augmented MDP consists of the velocities to be chosen by the robot. We apply Sarsa($\lambda$) RL (Wiering and Schmidhuber 1998) to determine the optimal policy which minimizes the time to reach the destination. In contrast to previous works that aim at minimizing the uncertainty in the belief distribution (Huynh and Roy 2009; He et al. 2008; Kollar and Roy 2006; Roy et al. 1999), our approach enables the robot to reach the destination as fast as possible. Thereby, the robot learns to actively avoid delays caused by localization errors. Furthermore, we show that it is possible to compress the learned policy using a clustering technique.

Experiments carried out in simulation and with real robots demonstrate that the learned policy significantly outperforms standard navigation strategies such as constant velocity policies or dual-mode controllers (Cassandra et al. 1996). Additionally, we provide experiments showing that the learned policy can be transferred to different indoor and outdoor scenarios with different landmark densities and to navigation tasks with multiple waypoints.

Note that the purpose of our work is not to compute a path for the robot. We assume that a sequence of subgoals or waypoints the robot has to traverse are already given, i.e., computed by an external path planning module. Instead, we aim at determining appropriate velocities for the robot so as to reach the navigation goals reliably and efficiently. We consider here an instance of the more general problem of system dynamics affecting localization accuracy. Our technique is relevant whenever time matters in navigation tasks, i.e., when a robot is desired to execute a task as fast as possible. Examples are delivery tasks with wheeled robots or playing soccer with humanoid robots. In these applications, the robots need an accurate estimate of their pose in order to successfully fulfill the task, and have to trade off velocity against localization accuracy.

The remainder of this article is structured as follows. In the Sect. 2, we first present our vision-based localization system. The navigation task and our learning approach are described in detail in Sect. 3. Afterwards, we introduce our technique for policy compression in Sect. 4. In Sect. 5 we then provide and thoroughly discuss experimental results. Finally, we present related work in Sect. 6.

## 2 Localization

In this section, we describe how we track the robot's pose over time given observations made by the robot and executed motion commands.

### 2.1 The unscented Kalman filter

We apply the *unscented Kalman filter* (UKF) to estimate the pose of the robot in a given map of the environment. The UKF is a recursive Bayes filter to estimate the state $\mathbf{x}_t$ of a dynamic system (Julier and Uhlmann 1997). This state is represented as a multivariate Gaussian distribution $N(\mu, \Sigma)$. The estimate is updated using nonlinear controls and observations $\mathbf{u}_t$ and $\mathbf{z}_t$. The key idea of the UKF is to apply a deterministic sampling technique that is known as the unscented transform to select a small set of so-called sigma points around the mean. Then, the sigma points are transformed through the nonlinear state transition and measurement probability functions, and the Gaussian distributions are recovered from them thereafter. The UKF can better deal with nonlinearities and thus leads to more robust estimates compared to other techniques such as the extended Kalman filter. Besides Monte Carlo localization, Kalman filter-based localization is one of the standard techniques applied in mobile robotics.

### 2.2 Vision-based pose estimation

A control $\mathbf{u}_t$ for the UKF is obtained from the robot's motion. We use an odometry motion model here, utilizing the data from the robot's wheel encoders (Thrun et al. 2005).

As observations $\mathbf{z}_t$, we extract Speeded-Up Robust Features (Bay et al. 2006) from the camera images as visual landmarks, as depicted in Fig. 1. Extracted descriptors of these features are then matched to landmarks in a map. This map was constructed beforehand and contains the global 2D positions and descriptors of the landmarks on the floor. Whenever the robot matches a perceived feature to a landmark in the map, it integrates the relative 2D position of the landmark as observation $\mathbf{z}_t = (r_t, \varphi_t)$ in the UKF in order to estimate its pose $\mathbf{x}_t = (x_t, y_t, \theta_t)$. Here, $r_t$ and $\varphi_t$ are the polar coordinates of the landmark relative to the robot, $x_t$ and $y_t$ are the global position of the robot, and $\theta_t$ is the robot's orientation.

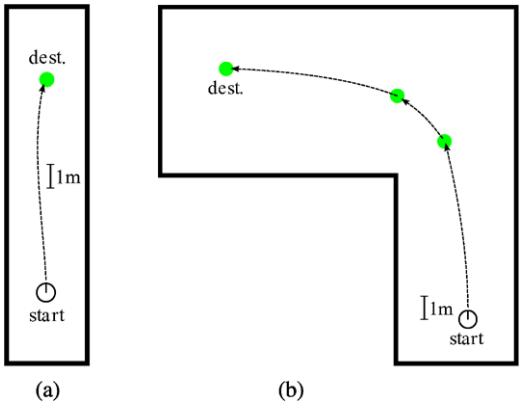



**Fig. 2** Navigation environments for learning (**a**) and evaluation (**a**) and (**b**)

## 3 Learning navigation policies

In this section, we first describe the navigation task we consider and then formulate it as a reinforcement learning problem.

### 3.1 Navigation task

The objective of the robot is to reach its navigation goal reliably and as fast as possible. We assume that a path of waypoints is given which can be obtained by a path planner such as $A^*$. During learning, we consider the scenario of a robot navigating from its current position to one such intermediate goal location, which is in viewing distance (see Fig. 2(a)). The learning task is completed as soon as the distance between the robot's true position and the goal location is below a certain threshold.

After learning, the policy is then also applied to a more general scenario containing several waypoints on a longer path (see Fig. 2(b)). In this application, once the robot's pose estimate is sufficiently close to the current goal point, the next waypoint on the path is regarded as the new goal point. The task is finished as soon as the distance between the robot's true position and the final destination is below a certain threshold.

We employ a straightforward controller which steers the robot to the next goal point, based on the current most-likely pose estimate $\mathbf{x}_t = (x_t, y_t, \theta_t)$ and a desired target velocity $v_{\text{target}}$ (which corresponds to the chosen action, see Sect. 3.6). Depending on the angle $\varphi$ to the next goal point, the translational and rotational velocities $v$ and $\omega$ are set in the following way. When $|\varphi| \geq \frac{\pi}{2}$, $v$ is set to zero and the robot orients itself towards the goal. Otherwise, $v$ is set to the desired target velocity $v_{\text{target}}$ and $\omega$ is set depending on $\varphi$.

The overall velocity influences the visual perception of the robot because the observed scene is affected by motion blur. The faster the robot moves, the more its visual perception is degraded. This has a direct impact on feature extraction and matching and, thus, on the localization performance. By moving slowly, the negative impact of motion blur can be avoided, but the robot needs more time to finish the navigation task. We formulate the problem of trading off velocity against localization accuracy as a reinforcement learning task.

### 3.2 Reinforcement learning

In reinforcement learning, an agent seeks to maximize its reward by interacting with the environment (Sutton and Barto 1998). Formally, this is defined as a *Markov decision process* (MDP) using the state space $\mathcal{S}$, the actions $\mathcal{A}$, and the rewards $\mathcal{R}$. By executing an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$, the agent experiences a state transition $s_t \to s_{t+1}$ and obtains a reward $r_{t+1} \in \mathcal{R}$. The overall goal of the agent is to maximize its return $R_t$ given by

$$R_t = \sum_{i=t+1}^{T} r_i, \tag{1}$$

where $T$ is the time when the final state is reached. One finite sequence of states $s_0, \ldots, s_T$ is called an *episode*.

The decision of which action to take in a certain state is governed by the policy

$$\pi(s, a) = p(a|s) \quad \forall s \in \mathcal{S}, \tag{2}$$

which denotes the probability of taking action $a$ in state $s$. The *action-value function*, also called *Q-function*, for a policy $\pi$ is defined as

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s,\ a_t = a\}, \tag{3}$$

which denotes the expected return of taking action $a$ in state $s$ and following policy $\pi$ afterwards. The optimal policy maximizes the expected return, which corresponds to the maximum $Q$-value for each state-action pair.

A reinforcement learning problem is solved by finding the optimal policy, for example, through temporal difference (TD) learning (Sutton and Barto 1998). TD learns model-free, which means that transition probabilities between states do not need to be defined but are experienced from samples. *Sarsa* is one such TD learning algorithm (Rummery and Niranjan 1994).[1] The estimate of the Q-function is continuously updated in Sarsa. $Q^{\pi}(s, a)$ is learned on-policy, which means that the current behavior policy $\pi$ is

[1] In their original work, Rummery and Niranjan called the algorithm *Modified Connectionist Q-Learning*. The name "Sarsa" was introduced by Sutton (1996).

learned and updated. The current estimate is updated based on its old values, the new reward, and the new value:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\big). \tag{4}$$

The parameters $\alpha$ and $\gamma$ are step size and discounting factor, respectively. This form of Sarsa uses only the immediate reward $r_{t+1}$ and belongs to the class of TD(0) learners.

By looking further into the future, a more accurate estimate of $R_t$ can be obtained. We use Sarsa($\lambda$), an extension of Sarsa that averages over a number of future rewards (Wiering and Schmidhuber 1998). In detail, an $n$-step return

$$R_t^{(n)} = \sum_{i=1}^{n} \gamma^{i-1} r_i + \gamma^n Q_t(s_{t+n}, a_{t+n}) \tag{5}$$

looks $n$ steps into the future and takes the future rewards into account. Sarsa($\lambda$) uses a number of these $n$-step returns, weights them by $\lambda^{n-1}$, and computes the average, yielding the so-called $\lambda$-return

$$R_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}. \tag{6}$$

The parameter $\lambda \in [0, 1]$ determines the decay of the impact of future rewards. For $\lambda = 0$, only the immediate rewards are used, which corresponds to the previously discussed TD(0) learning.

Throughout the learning phase, we use an $\varepsilon$-greedy action selection. This selection method chooses the action with the highest $Q$-value with probability $\varepsilon$ and all non-greedy actions with equal probability in each step.

### 3.3 Augmented Markov decision process

In an MDP, it is assumed that the agent is able to uniquely determine its state. If the belief about the state is represented by a probability distribution, the system is ideally modeled by a partially observable MDP (POMDP, Sondik 1971). Since POMDPs require an explicit modeling of the probability distribution of the state, they are computationally hard to solve and intractable for most real-world tasks. It is a common practice to use approximations instead (Lovejoy 1991; Roy and Gordon 2002). In cases where the underlying distribution is modeled by a unimodal distribution as in our case, the so-called *augmented MDP* (Roy and Thrun 1999) can be used as an efficient approximation. The belief of the state is then represented by its most-likely estimate and the task is modeled as an MDP. The uncertainty of the underlying belief distribution is taken into account by including the corresponding entropy in the state representation.

The probability distribution of the robot's pose given all previous odometry information and visual observations is estimated by an UKF. For the MDP, we define the state space $\mathcal{S}$, the set of actions $\mathcal{A}$, and the rewards $\mathcal{R}$ as follows.

### 3.4 State space $\mathcal{S}$

The complete state of the robot consists of the global pose estimate $\mathbf{x}_t$, the current velocity, and a characterization of the environment including the goal location and the landmarks. However, this complete state representation is impractical to consider for reinforcement learning. Learning in this complete description would take too long and generalization would be hard to achieve.

Thus, we define a set of features based on the complete state which characterizes the state sufficiently detailed and as general as needed for learning. Based on the current, most-likely pose estimate $\mathbf{x}_t = (x_t, y_t, \theta_t)$ and the environment, we define the following features:

- The Euclidean distance to the next goal point $(g_x, g_y)^T$

$$d = \sqrt{(g_x - x_t)^2 + (g_y - y_t)^2}. \tag{7}$$

- The angle relative to the next goal point

$$\varphi = \operatorname{atan2}(g_y - y_t, g_x - x_t) - \theta_t. \tag{8}$$

  In combination with $d$, this completely characterizes the relative position of the next goal point which has to be reached. In the multiple-waypoint scenarios, the next waypoint is regarded as goal point.
- The uncertainty of the localization, represented in terms of the differential entropy of the pose:

$$h = \frac{1}{2} \ln((2\pi e)^3 \cdot |\det(\Sigma)|). \tag{9}$$

  This measures how well the robot is localized: A higher entropy corresponds to a higher pose uncertainty. Note that we use the *differential* or continuous entropy here as opposed to the *Shannon* entropy, because the robot's pose estimate is continuous. Since the probability density function can be greater than one, $h$ can have negative values.

We experimentally found these features $d, \varphi$, and $h$ to be most relevant and sufficient for completing the task. Other combinations of them, also including the current velocity and the landmark density in the state representation, did not lead to a significant improvement of the robot's performance.

We represent the state-action space by a radial basis function (RBF) network, which is a linear function approximator (Doya 2000). The continuous features of the state are approximated by a discrete, uniform grid. In between the centroids of the grid, the state is linearly interpolated with a

Gaussian activation function. In contrast to a strictly discrete representation as feature table, the RBF network suffers less from the effects of discretization.

### 3.5 Possible extensions of the state space

A probably straightforward extension is to incorporate semantic information into the state space. For example, if there are different surfaces in the environment, such as carpet and concrete, this information can be easily included into the state space. The quality of the odometry of the robot typically depends on the surface. In this way, this knowledge can be considered during action selection since it has an influence on the resulting pose uncertainty.

The information about the surface can be estimated using vibration-based approaches which have been applied in outdoor environments (Wurm et al. 2009; Weiss et al. 2006) or it can be integrated into the environment representation by hand. Given the pose estimate of the robot, the semantic information can then be directly deduced from the map. By including the semantic label into the state space, the approach allows the robot to select the motion commands that are best suited in its current situation.

### 3.6 Action set $\mathcal{A}$

The actions the robot can choose from correspond to restricting the overall velocity of the navigation controller to the so-called target velocity $v_{\text{target}}$. The rotational and translational velocities of the robot are selected in a way such that the resulting movement of the camera respects this value $v_{\text{target}}$. Choosing $v_{\text{target}}$ (in m/s) is the action the robot learns via reinforcement learning. We define the possible actions as

$$\mathcal{A} = \{0.1, 0.2, 0.3, 0.4, 1.0\}. \tag{10}$$

We determined this discretization according to the effect of motion blur on the landmark observation probability (see Fig. 5). In our setting, velocities higher than 0.4 m/s blur the image almost beyond recognition.

### 3.7 Rewards $\mathcal{R}$

We define the immediate reward at time $t$ as

$$r_t = \begin{cases} 100 & \text{if } t = T \\ -\Delta_t & \text{otherwise,} \end{cases} \tag{11}$$

where $T$ is the final time step and $\Delta_t$ is the time interval between the update steps. The final state is reached when the robot's true pose is sufficiently close to the destination. This has the effect that the robot is driven to reach the destination as fast as possible.

We do not model an explicit punishment for delocalization or running into a wall. We assume that the robot has some sensors for obstacle avoidance on board, such as bumpers, infrared, or sonar. When the robot is in danger of running into an obstacle, it is immediately stopped by the obstacle avoidance. The time it takes to stop, re-localize, and accelerate is the implicit punishment for getting off the track, which is typically a few seconds.

## 4 Policy compression

A policy learned using the presented framework is represented by a table whose size depends on the discretization of the state space. For each given state tuple $(d, \varphi, h)$ of distance to the next goal point, angle, and entropy, the optimal velocity $v_{\text{target}}$ can be obtained from that table.

While the learned table of the policy contains entries for all values of $(d, \varphi, h)$ within the discretized ranges, not all of these values are relevant since some never appear in practice. Thus, it is desirable to find a more compact representation of the learned policy, which might be interesting when implementing it on systems with memory constraints.

In contrast to methods of dimension reduction in particular during learning (Uther and Veloso 1998; Rubinstein and Kroese 2004; Menache et al. 2005; Satoh 2006), we aim at a data reduction of the representation of the learned policy when it is executed, i.e., the learned *behavior* of the robot.

### 4.1 Formulation as classification problem

We treat the problem of finding a compressed policy as a classification problem on the visited state space, i.e., we use labeled samples available from the robot following the learned policy. In particular, the robot follows the learned policy for 100 episodes. For each visited state $(d, \varphi, h)$, we regard the chosen velocity $v_{\text{target}}$ as the classification of the state. This labeled data contains only the regions of the state space that are relevant for the task.

We hereby abstract from the RBF network as an underlying function approximation of the state space. Any kind of function approximator can be used during the learning phase instead. Afterwards, the learned policy is used greedily and our compression method can be applied to the visited state space and the chosen actions.

### 4.2 $X$-means clustering

We use $X$-means clustering (Pelleg and Moore 2000) to find a number of clusters which approximate the data $d, \varphi, h$, and $v_{\text{target}}$. $X$-means, which is an extension of $K$-means clustering, finds the best number of clusters according to the Bayesian information criterion (BIC), also known as

Schwarz criterion. We cluster input data points into $K$ clusters. Initially, $K$ random cluster means are chosen. Based on the Euclidean distance to these means, the data are iteratively assigned to the closest mean and the mean is adjusted to the centroid of the cluster. The BIC of a model $M_j$ for input data $D$ is given as

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \log |D|, \tag{12}$$

where $\hat{l}_j(D)$ is the log-likelihood of the data according to the model and $p_j$ is the number of parameters in the model (Kass and Wasserman 1995). A model hereby corresponds to a $K$-means clustering solution for a given $K$.

After the number of clusters and the locations of their means are found, each data point $(d, \varphi, h, v_{\text{target}})$ is assigned to the cluster that is closest by means of the Euclidean distance. To account for different scales, such as distance in meters and angle in radians, state vectors and cluster means are normalized within their discretization range. During clustering, the velocity classification of each cluster is set to the average velocity of the samples within that cluster. The cluster means—each labeled with a velocity value—can now be used as a compact representation of the learned policy. During execution, the robot uses the tuple $(d, \varphi, h)$ to look up the closest cluster and its velocity.

## 5 Experiments

We conducted the practical experiments with two different ActivMedia robots, a Pioneer 2-DX8 (Fig. 3) and a Powerbot (Fig. 4). We equipped both robots with a top-mounted ImagingSource DFK 31AF03 camera to observe the ground in front of them and extract Speeded-Up Robust Features (SURF) from the images as visual landmarks. These observations are integrated with odometry information in an UKF for vision-based localization as described in Sect. 2. Additionally, a SICK laser range finder was mounted on the robots and used for obstacle avoidance as well as for providing a ground truth pose estimate needed for evaluation. The indoor environment for the Pioneer robot was a hallway with wood parquet floor and a distance of approximately eight meters between the starting pose and the destination. The Powerbot robot was employed in a paved outdoor environment with four waypoints and a total path length of approximately 30 meters. To make the indoor experiments more challenging, one tire of the Pioneer robot was slightly deflated in order to introduce a systematic error in the odometry.

We learned the policy in simulations. This allows for evaluating different parameter settings for the learning algorithm and for running a large number of learning and evaluation episodes. We modeled the robots and their environments as realistic as possible. This includes the systematic

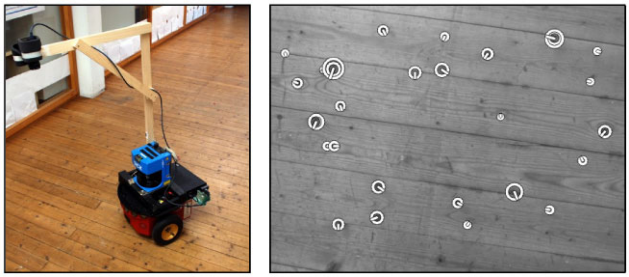


**Fig. 3** Pioneer 2-DX8 robot in the experimental indoor environment (*left*) and an observed floor patch with SURF as visual landmarks (*right*)

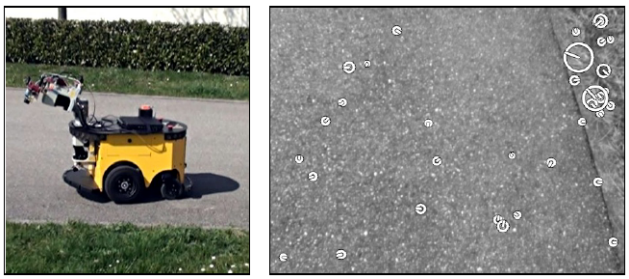


**Fig. 4** Powerbot robot in the experimental outdoor environment (*left*) and an observed ground patch with SURF as visual landmarks (*right*)

error on the odometry of approximately 5° on the yaw angle per meter of translation. Instead of modeling SURF detection and matching explicitly, we used a map of artificial landmarks. The landmark positions were randomly distributed with an average density as in the real map (40 landmarks/m$^2$). Because we wanted to avoid an adaptation of the robot's behavior to a specific environment, landmark positions and the direction of the systematic error were randomized in each new learning and evaluation episode. Inaccuracies in the robot's initial pose estimate were modeled by sampling the initial position and orientation with a small standard deviation. In order to obtain a policy which takes motion blur into account, we modeled motion blur as an effect on the probability of an observation $z$ given the current velocity $v$, i.e., we determined the probability that a feature which is in the robot's field of view is detected given $v$. This dependency $p(z|v)$ was experimentally estimated using real data, as displayed in Figs. 1 and 5. We approximated the measured values by a fitted sigmoid function

$$p(z|v) = s_1 \cdot \left(1 - \frac{1}{(1 + e^{-s_2 v + s_3})}\right) + s_4 \tag{13}$$

with parameters $s_1, \ldots, s_4$. For our hardware setting, we obtained the values $s_1 = 1.0068$, $s_2 = 24.1437$, $s_3 = 5.3991$, and $s_4 = 0.0002$.

For the learning parameters, we used $\alpha = 0.2$, $\gamma = 0.95$, $\lambda = 0.85$, and $\varepsilon = 0.1$ (see Sect. 3.2). These values were experimentally determined and yielded good learning results.

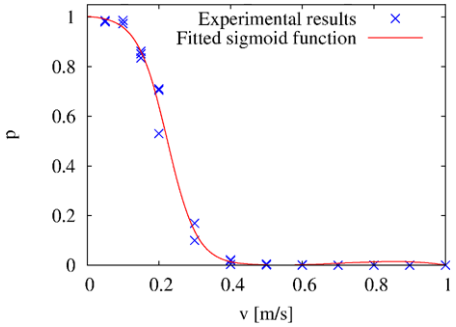



**Fig. 5** Experimentally determined observation model $p(z|v)$. The measured data (*blue crosses*) are approximated by a sigmoid function (*red line*)

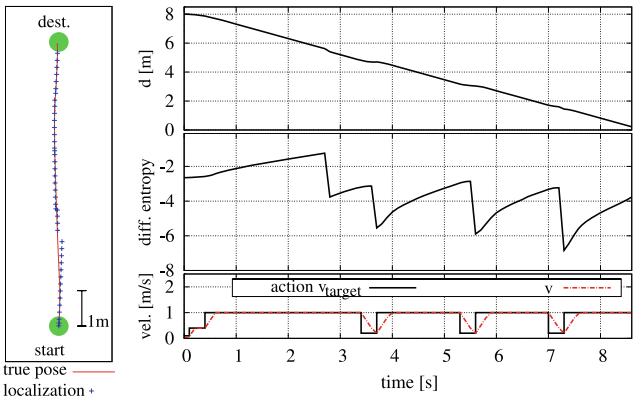



**Fig. 6** A typical example of the trajectory of a learned policy (*left*) and the dimensions *distance* and *entropy* of the state space with actions over time (*right*). The robot maximizes its velocity until its uncertainty gets too high, indicated by a high value of the differential entropy. To re-localize, it then slows down. As soon as the uncertainty decreases as an effect of localization, it accelerates again. As the robot approaches the goal location, it slows down more frequently

We found 500 learning episodes to be sufficient for convergence of the policy. We then evaluated it by using a greedy action selection in 100 evaluation episodes, measuring the average time from the starting pose to the destination and a 95% confidence interval. All following statements concerning significance are with respect to a $t$-test with 95% confidence.

In the following, we present and discuss our experimental results to evaluate the learned policy in terms of performance and generalizability. We present simulation experiments as well as results obtained with real robots.

## 5.1 Qualitative analysis

A typical trajectory and the corresponding state space over time of the learned policy is displayed in Fig. 6. The robot optimizes its time to reach the destination by driving at maximum speed as long as it is confidently localized. When there is risk of getting lost, indicated by a high entropy, it slows down in order to observe landmarks. As mentioned in Sect. 3.4, the differential entropy can have negative values. Note that for different values of the distance $d$, different levels of the entropy are learned to be important. As the robot gets closer to the goal, it frequently slows down so that the target is reliably reached.

## 5.2 Quantitative analysis

We now quantitatively compare a policy learned using our approach to two standard methods of setting the target velocity $v_{\text{target}}$. In the absence of motion blur, the best choice to minimize the time to the destination would be the highest possible velocity. In our scenario, however, there is not an immediate benefit of driving at maximum speed. If a high velocity is chosen, the robot quickly gets lost which leads to a longer overall journey time since the robot has to stop and re-localize later on.

### 5.2.1 *Comparison to constant velocity*

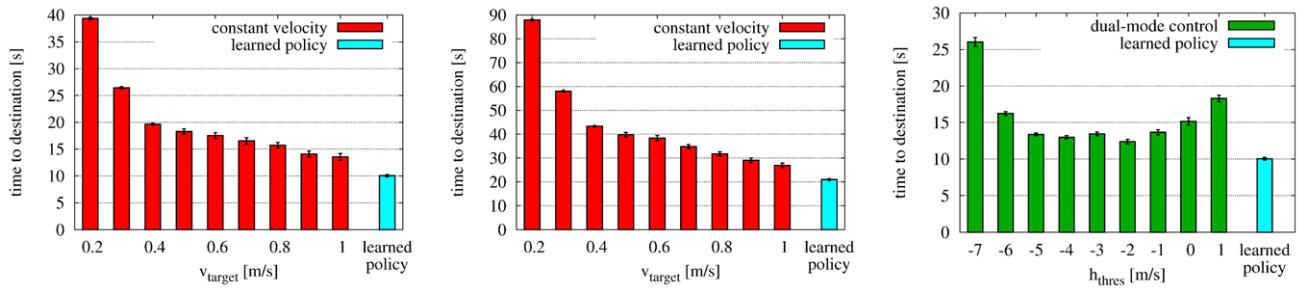
A naive approach is to set a constant target velocity $v_{\text{target}}$. Figure 7a displays an evaluation of following a constant velocity from $v_{\text{target}} = 0.2$ m/s to 1 m/s, compared to our learned policy. Up to 0.4 m/s, an increased velocity directly improves the time to destination. For higher velocities, the robot is no longer able to perform observations, regularly gets lost on its path, and has to stop in order to avoid collissions and to re-localize. Despite this, there is still a small improvement in the average time to destination. This means the robot accepts the risk of nearly colliding and getting lost, in favor of a faster speed.

But even when choosing the best policy of constant velocity, our learned approach is significantly better. While the average time to destination at 1 m/s is $13.56\,\text{s} \pm 0.61\,\text{s}$ (95% confidence interval), the robot is able to finish the task with our learned policy in $10.04\,\text{s} \pm 0.18\,\text{s}$, which corresponds to a reduction of 26%.

### 5.2.2 *Comparison to a dual-mode controller*

A more advanced approach is to employ a *dual-mode controller* as introduced by Cassandra et al. (1996). Similar to our learned policy, the entropy is used to decide on which action to take. When the entropy is above a threshold $h_{\text{thres}}$, an action to reduce the uncertainty is selected, otherwise a greedy action is chosen. These actions are $v_{\text{target}} = 0.1$ and $v_{\text{target}} = 1.0$ in our scenario, respectively. Figure 7c displays the resulting times for various values of $h_{\text{thres}}$ compared to the learned policy.

Using the dual-mode controller, we achieve best results for $h_{\text{thres}} = -2$, resulting in a time to reach the destination of $12.39\,\text{s} \pm 0.31\,\text{s}$. The learned policy still yields a significant reduction of 17%.




(a) Constant velocity in the simulated short scenario (path length: 8 m, two waypoints, see Fig. 2(a)).

(b) Constant velocity in the simulated longer scenario (path length: approx. 17 m, four waypoints, see Fig. 2(b)).

(c) Dual-mode control policies at various thresholds for the entropy in the simulated short scenario (path length: 8 m, two waypoints).

**Fig. 7** Comparison of constant velocity policies and the dual-mode controller to our learned policy in simulations. Each policy is displayed with mean and 95% confidence interval over 100 runs. The learned policy is significantly better than each other policy

**Table 1** A policy learned in an environment with 40 landmarks/m$^2$ is evaluated in environments with 10, 40, and 70 landmarks/m$^2$ and is compared to a policy learned in the specific test environment. According to the results of the $t$-test, there is no significant difference in the performance which shows that the learner did not overfit to the learning environment

| Density in test environment | Density in learning environment | |
|---|---|---|
| | 40 | Matching test env. |
| 10 | $11.19 \pm 0.44$ s | $10.78 \pm 0.16$ s |
| 40 | $10.42 \pm 0.24$ s | |
| 70 | $10.32 \pm 0.25$ s | $10.56 \pm 0.18$ s |

**Table 2** A policy learned for a path length of $d = 8$ m is evaluated at shorter path lengths and is compared to policies learned for the specific lengths. Small differences are apparent at short lengths, where it would be sufficient to drive blindly ahead to the goal without slowing down

| $d$ in test environment | $d$ in learning environment | |
|---|---|---|
| | 8 m | Matching test env. |
| 8 m | $10.62 \pm 0.31$ s | |
| 6 m | $8.31 \pm 0.21$ s | $8.28 \pm 0.25$ s |
| 4 m | $6.87 \pm 0.32$ s | $6.16 \pm 0.27$ s |
| 2 m | $4.20 \pm 0.25$ s | $3.27 \pm 0.19$ s |

To summarize, our learning approach outperforms the dual-mode controller which represents a special case of our approach by just considering the entropy and choosing only between two actions. This indicates that a more complex learning framework such as ours is indeed necessary in order to gain improved performance.

### 5.3 Generalization over landmark density

We will now examine how the density of landmarks affects the learned policies, and how one can generalize over different environments. To do so, we evaluated policies learned in an environment with an average landmark density of 40 landmarks/m$^2$ in significantly sparser (10 landmarks/m$^2$) and denser environments (70 landmarks/m$^2$). We compared the performance to a policy learned in an environment with the same landmark density as the respective test environment, which we expect to yield the best results.

In order to obtain general results, we compare 50 independently learned policies, each learned in 500 learning episodes and evaluated over 100 test episodes. The resulting times are displayed in Table 1. There is no significant difference between using a fixed density of 40 landmarks/m$^2$ during learning and the optimal case where the landmark density of the learning environment fits the test environment. This can be seen as an indication that our learned policy does not overfit to the learning environment and that it can be applied to different environments, without the need of explicitly accounting for this in the learning scenario.

### 5.4 Application to general navigation paths

So far, we considered the scenario of a rectilinear path with a fixed length of eight meters. We now show how a policy learned in this environment can be used in scenarios with general paths, i.e., paths of different lengths and containing multiple waypoints that have to be passed by the robot.

We evaluated the robustness of the policy towards shorter paths by comparing it to policies learned in the specific test environment. Applying the policy learned from a longer path to shorter paths is straightforward. Again, 50 independently learned policies were compared for each path length. Table 2 displays the results. At a path length of six meters, there is no significant difference. However, shorter scenarios with two or four meters distance produce small, but significant different outcomes. In these cases, a policy tailored to the specific short path length seems to be better. Especially for the distance of two meters, it would be sufficient to drive blindly at maximum speed, since it is unlikely not to reach the destination with sufficient accuracy.

We now consider paths which consist of a number of successive waypoints. This does not only allow for longer trajectories but also for more complex, non-straight paths and navigation tasks involving multiple waypoints that have to be reached subsequently. Figure 7(b) displays an evaluation of following a constant velocity policy compared to a policy learned in the short two-waypoint scenario. The total path length in this evaluation scenario over four waypoints is approximately 17 meters (see Fig. 2(b)). As in the two-waypoint scenario, the learned policy significantly outperforms the best constant velocity policy. This demonstrates that the learned policy generalizes over different path lengths.

### 5.5 Verification on real robotic systems

We now transfer the results from simulations into the real world. We apply the policy learned in simulation on two real robots. We first employ the Pioneer robot as shown in Fig. 3 in an indoor environment similar to the environment depicted in Fig. 2(a). Each policy is evaluated in 10 test runs consisting of navigating from the start location to the destination. The resulting navigation times are shown in Fig. 8.

Similar to the results from simulations, the learned policy outperforms any policy of constant velocity by more than 25% and is significantly better. When looking at the trajectories generated by the policies qualitatively, the results are also similar to the simulated ones (Fig. 9). At a slow constant velocity, the robot stays close to the optimal path of the straight-line connection between start and destination. When driving faster at 0.8 m/s, the robot is not able to observe landmarks and quickly gets lost with the result of a near-collision with the wall. Contrary to that, the robot is not stopped by the obstacle avoidance when following the learned policy. When the robot is in risk of getting lost, it immediately slows down to re-localize. As a result, the robot reaches its destination reliably and fast.

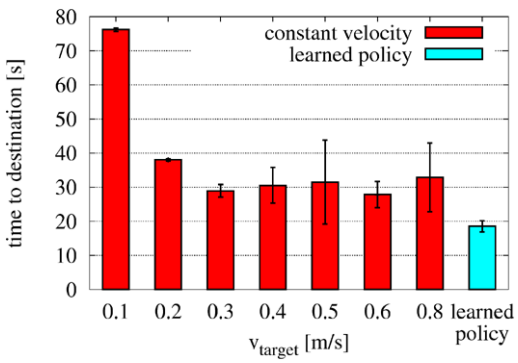



**Fig. 8** Comparison of constant velocity policies to a learned policy evaluated in the real indoor scenario (path length: 8 m, two waypoints). Each policy is displayed with mean and 95% confidence interval over 10 runs. The learned policy is significantly better than each policy with a constant velocity

Furthermore, we demonstrate the applicability to multiple-waypoint outdoor environments by employing the Powerbot (Fig. 4) in an environment similar to the one depicted in Fig. 2(b). As shown in Fig. 10, the robot successfully navigates to the destination using the learned policy to trade off a fast velocity against an accurate localization. Due to the open environment, the robot only finds the destination at slow constant velocities. When driving faster than 0.3 m/s,

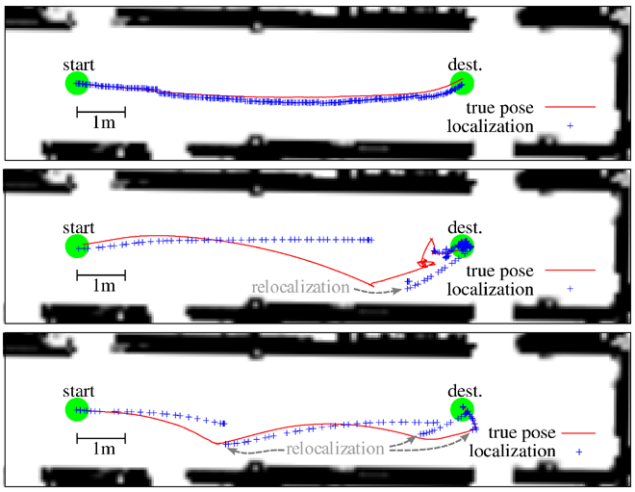



**Fig. 9** Comparison of real robot trajectories at constant velocity (top: 0.2 m/s, middle: 0.8 m/s) and variable velocity, i.e., following the learned policy (*bottom*). Since the localization is plotted with a constant time interval, the velocity of the robot can be inferred from the distance of the localization points (the larger the distance the higher the velocity). This behavior is also displayed in Animation 1 (available online)

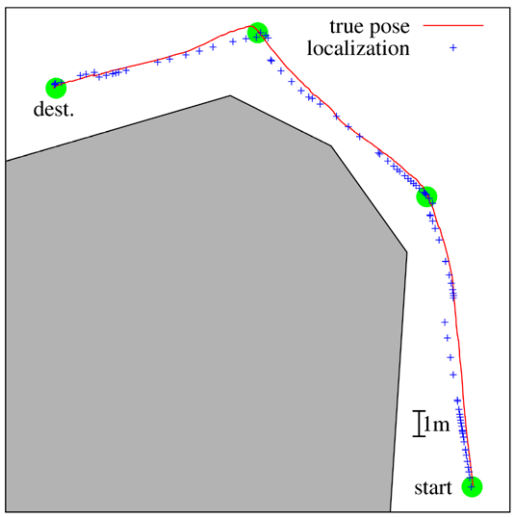



**Fig. 10** Trajectory when following the learned policy in a real multiple-waypoint outdoor environment. The *gray area* is terrain not traversable by the robot. Since the localization is plotted with a constant time interval, the velocity of the robot can be inferred from the distance of the localization points. This behavior is also displayed in Animation 2 (available online)

it cannot obtain reliable observations anymore, resulting in a complete delocalization which usually leads to leaving the mapped area. In these cases, the robot needs to be stopped by hand.

To summarize, the policy learned in simulations could be successfully applied on real robots and performs significantly better than the naive approach of driving with a constant velocity.

### 5.6 Policy compression

The policy we learned is represented by a table whose size depends on the discretization of the feature space. As discussed in Sect. 4, the robot then followed this learned policy in order to obtain labeled samples. These labeled data points are shown in Fig. 11, with the resulting clustering found via $X$-means in Fig. 12. In total, four clusters with three different average velocities were found (0.21 m/s, 0.26 m/s, $2 \times 1$ m/s). These four clusters lead to a representation which is significantly smaller than the full initial table. For comparison, a careful reduction of the state space discretization by hand (with no significant loss of performance) lead to a table which still had size $10 \times 3 \times 4 = 120$.

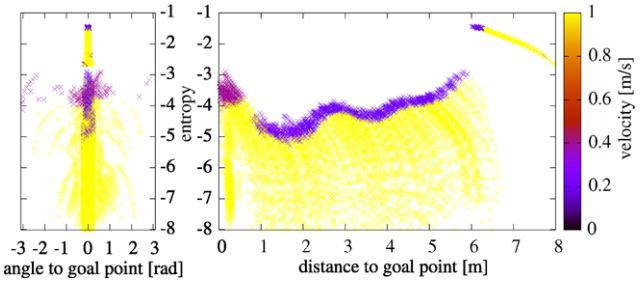



**Fig. 11** Evaluation of the learned policy throughout 100 simulated test episodes. For visualization, the three-dimensional state space is projected into the dimensions angle/differential entropy (*left*) and distance/differential entropy (*right*). Color (grayscale) denotes the action as velocity $v_{\text{target}}$ set by the policy

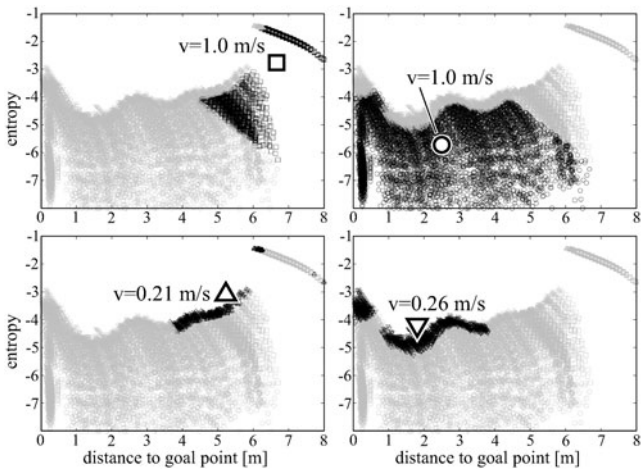



**Fig. 12** $X$-means clustering of the learned policy (see Fig. 11). Observed values of distance, entropy, angle, and the corresponding velocities are used for clustering, while only the projection on distance and entropy is shown here. Four clusters with three different velocity classifications are found, highlighted in the single plots

Note that the velocity distribution and the resulting clustering also illustrates that our navigation task is too complex to be solved with a dual-mode controller. The entropy alone yields not enough information to select an optimal velocity.

To decide on which velocity to set, the robot now uses the approximated policy which consists of the four clusters instead of the table-based action selection. For each visited state $(d, \varphi, h)$, the velocity of the closest cluster is selected. Using the policy represented by the original table, the robot needs 10.04 s $\pm$ 0.18 s to finish the task. Using the approximated representation, it is able to finish the task within 10.87 s $\pm$ 0.70 s. Thus, there is no significant performance difference between the two representations. This shows that we were able to compress the learned policy to a significantly smaller representation with no loss of performance in the task.

It is worth noting that this compression is at the cost of a slightly increased query time when using the policy on a system. Instead of directly querying a table entry, the nearest cluster mean needs to be found. Since the number of cluster means is typical small, however, this only leads to a minimal increase of computational effort.

## 6 Related work

In the last few years, various frameworks have been presented which employ active methods in the context of localization and navigation. Kollar and Roy (2006) use reinforcement learning to optimize the robot's trajectory during exploration. Similar to our approach, the authors learn optimal parameters of the navigation controller. While we consider the problem of reaching the destination reliably and as fast as possible, Kollar and Roy learn the translational and rotational behavior which minimizes the uncertainty in SLAM (simultaneous localization and mapping). Huynh and Roy (2009) generate control laws by combining global planning and local feedback control to obtain trajectories which minimize the pose uncertainty during navigation. Cassandra et al. (1996) introduced *dual-mode controllers* as heuristics for POMDPs. A threshold on the entropy as a measure of the uncertainty determines whether a greedy action or an action reducing the uncertainty is selected.

A different method of minimizing the uncertainty about the state of the robot is to plan a path for the robot which takes the information gain into account. Roy et al. (1999) presented an approach for this called *coastal navigation*. Recently, He et al. (2008) have applied this technique to

a quadrotor helicopter for indoor navigation with a short-range laser range finder. Bryson and Sukkarieh (2006) suggested a framework for unmanned aerial vehicle localization and exploration in unknown environments. They also use an intelligent path planning scheme in order to maximize the quality of the resulting SLAM estimate. Similarly, Martinez-Cantin et al. (2009) proposed a Bayesian optimization method that trades off exploration and exploitation for online path planning.

Strasdat et al. (2009) developed a reinforcement learning approach for the problem of landmark selection in visual SLAM. They learn a policy on which new landmark to integrate into the environment representation in order to improve the navigation capabilities of the robot. The motivation behind their work is the application on memory-constrained systems where it is not possible to maintain all observed landmarks in the belief. Michels et al. (2005) proposed to learn a control policy for high speed obstacle avoidance of a remotely controlled car. Based on depth estimation with a monocular vision system, steering directions are learned. The authors focus on obstacle avoidance whereas we consider the effect of fast movements on the observation quality and adapt the speed accordingly. Kwok and Fox (2004) apply reinforcement learning to increase the performance of soccer-playing robots by active sensing. In their approach, the robot learns where to point its camera in order to localize relevant objects.

Several authors proposed techniques for state space reduction and basis function optimization in order to speed up reinforcement learning. Most notably is the work of Menache et al. (2005). They presented a method for optimizing the basis functions during the TD learning process using either a gradient-based approach or the cross entropy method (Rubinstein and Kroese 2004). The compression techniques proposed by Satoh (2006) concentrate on the *curse of dimensionality* and how to obtain an appropriate lower dimensional representation of the feature space. Uther and Veloso (1998) proposed a tree-based discretization for state space compression by determining relevant, contiguous parts of a large domain. While all these methods mainly focus on gaining a speed-up during learning, our approach, which compresses an already trained policy by clustering, is motivated by the storage problem. The question we address is: How can we represent the learned policy in a most compact way so that it becomes applicable on memory-constrained systems and so that, at the same time, the compression does not lead to a loss of performance.

Bennewitz et al. (2006) developed a localization method based on visual features and presented experiments with a humanoid robot. The authors mentioned the impact of motion blur on feature extraction, but did not address the problem specifically. Instead, their robot interrupted its movement at fixed intervals in order to perform observations. To overcome the problem of motion blur in the context of humanoid robots, Ido et al. (2009) explicitly consider the shaking movements of the head while walking and acquire images only during stable phases of the gait.

Pretto et al. (2009) proposed an additional image processing step prior to feature extraction, in particular for humanoid robots. The authors estimate the direction of the motion blur for image patches and developed a novel feature detection and tracking scheme. While their approach increases the matching performance, motion blur cannot be completely removed by filtering. However, such a pre-processing technique could be easily combined with our learning approach in order to further improve the navigation performance of the robot.

Miura et al. (2006) presented a method for adaptive speed control in partially unknown environments. In this approach, the velocity is chosen to be as fast as possible while still being safe in the sense that potential collisions with obstacles are avoided. The authors use heuristics which depend on the distance of the robot to unexplored areas and empirically determined safety margins around obstacles. In contrast to our work, the uncertainty of the robot about its pose is not explicitly considered. Similarly, reactive collision avoidance systems assume the pose of the robot to be known. These systems compute translational and rotational velocities for the robot, thereby considering the progress towards the goal, the current velocities, and the distance to obstacles (Stachniss and Burgard 2002; Brock and Khatib 1999; LaValle and Kuffner 1999; Schlegel 1998; Fox et al. 1997; Simmons 1996). Our approach can be seen as orthogonal to these techniques. It could be combined with them, e.g., by relating the size of the search space considered in these approaches to the uncertainty of the pose estimate.

To the best of our knowledge, we developed the first technique which learns about the influence of motion blur on the observations and hence on the localization performance, generating a policy to reach the destination reliably and as fast as possible. This article is an extension of our previous work (Hornung et al. 2009). The new contribution lies in the application of our approach to general scenarios also involving several goal points which have to be reached subsequently. Furthermore, we discussed technical issues in more detail and presented additional experiments carried out with two different robot platforms.

## 7 Conclusion and future work

In this article, we presented an approach which enables a robot to generate efficient policies for vision-based navigation. Typically, the quality of the pose estimate seriously decreases during fast movements. This is due to motion blur introduced in the images and the resulting high noise in the

feature observations. As a result of a wrong pose estimate, the robot may not be able to accomplish its navigation task efficiently and successfully.

We formulate the task of navigating to a target location reliably and, at the same time, as fast as possible as a reinforcement learning problem. While the robot applies the learned policy, it avoids delays caused by localization errors and implicitly takes the effect of motion blur on the feature detections into account when choosing appropriate velocities. As we showed in simulated and real-world experiments, our learned policy significantly outperforms strategies which apply a constant velocity and the more advanced dual-mode controllers with respect to the time to reach the destination. We used different indoor and outdoor scenarios to show the general applicability of the learned policy in terms of landmark density and path length.

Furthermore, we applied a clustering approach on the visited state space to compress the learned policy. In our experiments, the approximation yielded a similar performance as the original learned policy. This is especially valuable for memory-constrained systems such as lightweight UAVs.

In the future, we plan to apply our approach to walking humanoid robots or fast moving UAVs. Active policies for vision-based navigation are especially promising for such platforms since they have a high trade-off between accuracy and speed. The related navigation tasks are in general more complex than those presented in this article. For example, the motion of an UAV has typically 6 degrees of freedom (DOFs) where the movement along each dimension has a different impact on motion blur. For a moving humanoid robot, at least 20 DOFs have to be controlled. Thus, the resulting search spaces are high-dimensional. Approximations and dimension reduction techniques to explore them during reinforcement learning will be needed for efficient learning. The adaptation of our presented learning scheme to these more complex platforms is a challenging problem for future research.

**Acknowledgements** We would like to thank Gerhard Neumann for making the *Reinforcement Learning Toolbox* available (Neumann 2005). Further thanks go to Andreas Karwath for insightful discussions and hints about machine learning techniques.

## References

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: speeded-up robust features. *Proc. of the European Conf. on Computer Vision*, *110*(3), 346–359.

Bennewitz, M., Stachniss, C., Burgard, W., & Behnke, S. (2006). Metric localization with scale-invariant visual features using a single perspective camera. In H. I. Christiensen (Ed.), *Springer tracts in advanced robotics: Vol. 22*, *European robotics symposium 2006*. Berlin: Springer.

Brock, O., & Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA*.

Bryson, M., & Sukkarieh, S. (2006). Active airborne localisation and exploration in unknown environments using inertial SLAM. In *IEEE Aerospace Conference*.

Cassandra, A. R., Kaelbling, L. P., & Kurien, J. A. (1996). Acting under uncertainty: discrete Bayesian models for mobile-robot navigation. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS* (pp. 963–972).

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, *12*(1), 219–245.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*, 23–33.

He, R., Prentice, S., & Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA* (pp. 1814–1820).

Hornung, A., Strasdat, H., Bennewitz, M., & Burgard, W. (2009). Learning efficient policies for vision-based navigation. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS*.

Ido, J., Shimizu, Y., Matsumoto, Y., & Ogasawara, T. (2009). Indoor navigation for a humanoid robot using a view sequence. *Int. Journal of Robotics Research*, *28*(2), 315–325.

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *Int. symposium on aerospace/-defense sensing, simulation and controls*, pp. 182–193.

Kass, R. E., & Wasserman, L. (1995). A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. *Journal of the American Statistical Association*, *90*(431), 928–934.

Kollar, T., & Roy, N. (2006). Using reinforcement learning to improve exploration trajectories for error minimization. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA* (pp. 3338–3343).

Kwok, C., & Fox, D. (2004). Reinforcement learning for sensing strategies. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS* (vol. 4, pp. 3158–3163), 28 Sept.–2 Oct.

LaValle, S. M., & Kuffner, J. J. (1999). Randomized kinodynamic planning. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA* (pp. 473–479).

Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, *39*(1), 162–175.

Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., & Doucet, A. (2009). A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Journal of Autonomous Robots*, *27*(2), 93–103.

Menache, I., Mannor, S., & Shimkin, N. (2005). Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, *134*(1), 215–238.

Michels, J., Saxena, A., & Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. of the int. conf. on machine learning—ICML* (pp. 593–600). New York: ACM.

Miura, J., Negishi, Y., & Shirai, Y. (2006). Adaptive robot speed control by considering map and motion uncertainty. *Journal of Robotics & Autonomous Systems*, *54*(2), 110–117.

Neumann, G. (2005). The reinforcement learning toolbox, reinforcement learning for optimal control tasks. Diplomarbeit, Technischen Universität (University of Technology) Graz, May 2005.

Pelleg, D., & Moore, A. (2000). X-means: extending K-means with efficient estimation of the number of clusters. In *Proc. of the int. conf. on machine learning—ICML* (pp. 727–734). San Mateo: Morgan Kaufmann.

Pretto, A., Menegatti, E., Bennewitz, M., Burgard, W., & Pagello, E. (2009). A visual odometry framework robust to motion blur. In *Proc. of the IEEE int. conf. on robotics & automation (ICRA)*.

Roy, N., & Gordon, G. (2002). Exponential family PCA for belief compression in POMDPs. In S. Becker, S. Thrun, K. Obermayer (Eds.), *Proc. of the conf. on neural information processing systems—NIPS* (pp. 1043–1049), Vancouver, Canada, December 2002.

Roy, N., & Thrun, S. (1999). Coastal navigation with mobile robots. In *Proc. of the conf. on neural information processing systems—NIPS* (vol. 12, pp. 1043–1049).

Roy, N., Burgard, W., Fox, D., & Thrun, S. (1999). Coastal navigation–mobile robot navigation with uncertainty in dynamic environments. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA* (vol. 1, pp. 35–40).

Rubinstein, R. Y., & Kroese, D. P. (2004). *The cross-entropy method: a unified approach to combinatorial optimization, monte-carlo simulation and neural computation*. Berlin: Springer.

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems (Technical report CUED/F-INFENG/TR 166). Cambridge University, Cambridge, UK, September 1994.

Satoh, H. (2006). A state space compression method based on multivariate analysis for reinforcement learning in high-dimensional continuous state spaces. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E89-A*(8), 2181–2191.

Schlegel, C. (1998). Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot. In: *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS*.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA*.

Sondik, E. J. (1971). *The optimal control of partially observable Markov decision processes*. Ph.D. thesis, Stanford University, Stanford, USA.

Stachniss, C., & Burgard, W. (2002). An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS* (pp. 508–513), Lausanne, Switzerland.

Strasdat, H., Stachniss, C., & Burgard, W. (2009). Which landmark is useful? Learning selection policies for navigation in unknown environments. In *Proc. of the IEEE int. conf. on robotics & automation—ICRA*.

Sutton, R. S. (1996). Generalization in reinforcement learning: successful examples using sparse coarse coding. In *Proc. of the conf. on neural information processing systems—NIPS* (pp. 1038–1044). Cambridge: MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Adaptive computation and machine learning reinforcement learning: an introduction*. Cambridge: MIT Press.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. Cambridge: MIT Press.

Uther, W. T. B., & Veloso, M. M. (1998). Tree based discretization for continuous state space reinforcement learning. In *Proc. of the national conference on artificial intelligence—AAAI* (pp. 769–774).

Van Huynh, A., & Roy, N. (2009). icLQG: combining local and global optimization for control in information space. In *Proc. of the IEEE international conference on robotics and automation—ICRA*.

Weiss, C., Fröhlich, H., & Zell, A. (2006). Vibration-based terrain classification using support vector machines. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS*.

Wiering, M., & Schmidhuber, J. (1998). Fast online Q(λ). *Machine Learning*, *33*(1), 105–115.

Wurm, K. M., Kuemmerle, R., Stachniss, C., & Burgard, W. (2009). Improving robot navigation in structured outdoor environments. In *Proc. of the IEEE/RSJ int. conf. on intelligent robots and systems—IROS*.

**Armin Hornung** received his Diplom (Master's) degree in computer science with specialization in artificial intelligence and robotics from University of Freiburg, Germany in 2009. He is currently a Ph.D. student in the Humanoid Robots Laboratory at the University of Freiburg. His research interests include autonomous navigation, learning, and humanoid robots.

**Maren Bennewitz** studied Computer Science at the University of Bonn and received her M.Sc. degree in 1999. She got her Ph.D. in Computer Science from the University of Freiburg in 2004. From 2004 to 2008, she was a Postdoc in the humanoid robots lab at the University of Freiburg which she heads since October 2008 as an assistant professor. In the last few years, her research has been focused on multimodal human-robot interaction and state estimation problems. Currently, she is working on navigation with humanoid robots in complex indoor environments.

**Hauke Strasdat** received his Master's degree in Applied Computer Science from University of Freiburg, Germany in 2007. He is currently a Ph.D. student in the Department of Computing, Imperial College London. His research interests include robot vision, SLAM, machine learning, and optimization.