Learning Adaptive Navigation Strategies for Resource-constrained Systems

Armin Hornung[1] and **Maren Bennewitz**[1] and **Cyrill Stachniss**[1] and **Hauke Strasdat**[2] and **Stefan Oßwald**[1] and **Wolfram Burgard**[1]

Abstract. The majority of navigation algorithms for mobile robots assume that the robots possess enough computational or memory resources to carry out the necessary calculations. Especially small and lightweight devices, however, are resource-constrained and have only restricted capabilities. In this paper, we present a reinforcement learning approach for mobile robots that considers the imposed constraints on their sensing capabilities and computational resources, so that they can reliably and efficiently fulfill their navigation tasks. Our technique learns a policy that optimally trades off the speed of the robot and the uncertainty in the observations imposed by its movements. It furthermore enables the robot to learn an efficient landmark selection strategy to compactly model the environment. We describe extensive simulated and real-world experiments carried out with both wheeled and humanoid robots which demonstrate that our learned navigation policies significantly outperform strategies using advanced and manually optimized heuristics.

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 the only available sensor for navigation. 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. Figure 1 depicts two images of patches of a wooden floor recorded with a downward-looking camera on a wheeled robot and a humanoid robot walking through the same corridor. As can be seen, the movements of the robots introduce substantial motion blur to the image, which in practice will lead to a considerable reduction of the accuracy of the position estimation process. While there are methods to reduce the influence of motion blur [22] or limit image acquisition to stable phases of a gait [13], the degradation introduced by motion blur usually cannot be completely eliminated by filtering techniques and cheap cameras typically do not allow for an exact synchronization to the controllers executing the motor commands or the walking gait.

Additionally, small humanoids or unmanned aerial vehicles are often resource-constrained and possess only limited computational power. For truly autonomous navigation in initially unknown environments, however, the robot has to solve the so-called simultaneous localization and mapping (SLAM) problem. This is computationally demanding and the memory requirements increase with the number of landmarks that need to be maintained by the robot. In practice, there are many scenarios in which the number of visible landmarks during a navigation task is significantly larger than the number of landmarks that can be processed efficiently on an embedded device. This leads to the question which landmark should be stored and maintained by the robot to optimally solve the navigation task.

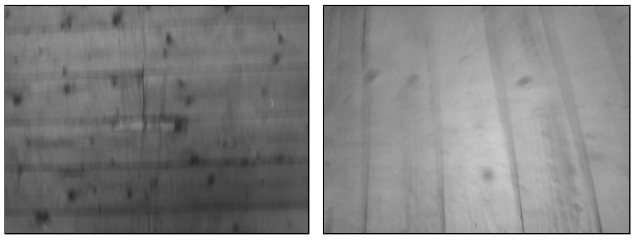


Figure 1. An indoor floor patch observed by a wheeled robot moving at 0.4 m/s (left), and by a walking humanoid robot (right). Significant motion blur is introduced in the captured images, degrading their quality for feature detection.

In general, the goal of the robot is to accomplish its task as fast as possible. However, as faster movements may introduce a higher uncertainty in the pose estimate due to the decreased reliability of the sensor data, they increase the risk of not being able to accomplish the mission. In principle, the robot therefore has to determine the movement speed that provides the optimal trade-off between the time needed to reach a designated target location and the risk of a positioning failure. Such questions often arise on systems with limited computational resources. The systems are usually not able to incorporate all the information into the state estimation processes which introduces a corresponding information selection problem. In this paper, we present a general approach towards learning optimal policies for systems with limited computational or perceptional capabilities, which at the same time are efficient and lead to reliable navigation behaviors. We use reinforcement learning (RL) to learn which navigation actions to execute so as to reach the destination reliably and efficiently. At each time step, the robot decides whether it should decrease the velocity or even stop to increase the quality of its perceptions or to continue moving towards the goal. In previous publications on vision-based navigation with wheeled and humanoid robots [9, 10, 21], we discussed the key concepts of our approach. Besides the localization problem, we investigate in this work how reinforcement learning can be used to decide which landmark to in-

[1] Dept. of Computer Science, University of Freiburg, Germany
[2] Department of Computing, Imperial College London, UK

tegrate during navigation without a known map [28]. We present experiments carried out in simulation and with real wheeled and humanoid robots and demonstrate that the learned policies significantly outperform manually optimized strategies and also techniques using advanced heuristics.

This paper is structured as follows. We first give an overview over related work in Sec. 2, followed by the background about state estimation and reinforcement learning in Sec. 3. Sec. 4 details our learning approach. Finally, in Sec. 5 we present the experimental results.

2 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 [15] 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 [12] generate control laws by combining global planning and local feedback control to obtain trajectories which minimize the pose uncertainty during navigation. Cassandra *et al.* [5] 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. A popular approach in this context is the so-called *coastal navigation* introduced by Roy *et al.* [23]. Recently, He *et al.* [8] have applied this technique to a quadrotor helicopter for indoor navigation with a short-range laser range finder.

Michels *et al.* [18] 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 [16] 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 to localize relevant objects.

Bennewitz *et al.* [3] 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 to make observations. To overcome the problem of motion blur in the context of humanoid robots, Ido *et al.* [13] explicitly consider the shaking movements of the head while walking and acquire images only during stable phases of the gait.

Pretto *et al.* [22] 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 present 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 to further improve the navigation performance of the robot.

Miura *et al.* [19] 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 this paper, we do not only investigate the ability to localize a vehicle but also to build maps under constraint settings. The standard method for SLAM relies on the extended Kalman filter (EKF) [6] or its variants such as the unscented Kalman filter (UKF) [14]. Using these approaches, the computational requirement and memory demand increase at least quadratically with the number of landmarks since the full correlation between the position of all landmarks is taken into account. There are many approximative filtering techniques for SLAM [20, 30]. These methods do not incorporate the full correlation between the landmarks, so that the computational constraints are less restrictive. However, their memory demand increases at least linearly with the number of landmarks used.

Recently, Sala *et al.* [25] presented a graph-theoretic formulation for the selection problem of visual features to perform navigation in known environments. The optimal set of features is defined as the minimal set with which navigation is possible. Zhang *et al.* [32] proposed an entropy-based landmark selection method for SLAM. This method specifies a measure of which visible landmark is best in terms of entropy reduction. However, it only provides a vague guideline for how many features should be selected at a given point in time. Furthermore, Lerner *et al.* [17] presented another quality measure for landmark selection in known environments which is based on the comparison of pose uncertainties. Dissanayake *et al.* [6] suggested a map management which ensures a uniform distribution of landmarks over the traversed area. Apart from landmark selection, other active methods were presented such as maximizing the SLAM estimate by intelligent path planning [4].

3 BACKGROUND

3.1 The Unscented Kalman Filter

The *unscented Kalman filter* (UKF) is a recursive Bayes filter to estimate the state $\mathbf{x}_t$ of a dynamic system [14]. 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 non-linearities and thus leads to more robust estimates compared to other techniques such as the extended Kalman filter.

3.2 Vision-based Localization

In this work, we use the UKF to perform state estimation. In case of localization, it estimates the 3D pose of the robot in a given 2D map of the environment. Besides Monte Carlo localization, Kalman filter-based localization is one of the standard techniques applied in mobile robotics.

A control $\mathbf{u}_t$ for the UKF is obtained from the robot's motion. On wheeled robots, an odometry motion model can be used, utilizing the data from the robot's wheel encoders [31]. Humanoid robots can use the executed motion command as a rough guess, or estimate their movement by integrating the leg joint angles while walking [11].

As observations $\mathbf{z}_t$, we extract *speeded-up robust features* (SURF) [2] from the camera images. Extracted descriptors of these features are then matched to landmarks in a map. This was constructed beforehand for each environment and contains the global 2D positions and SURF 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 $\mathbf{x}_t = (x_t, y_t, \theta_t)$.

3.3 Simultaneous Localization and Mapping

We also use the UKF for the setting when the environment is not known to the robot and the positions of landmarks need to be estimated as well. This problem is widely known as the landmark-based simultaneous localization and mapping (SLAM) problem where one seeks to simultaneously determine the map of the environment and the pose of the robot. We apply the UKF as a probabilistic method to estimate the joint probability distribution over the robot's pose and the landmark locations:

$$p(\mathbf{x}_t, \mathbf{l}_1, \ldots, \mathbf{l}_M \mid \mathbf{u}_1, \ldots, \mathbf{u}_t, \mathbf{z}_1, \ldots, \mathbf{z}_t) \tag{1}$$

Here, $\mathbf{x}_t$ is the pose of the robot at time t and the position of the landmarks $\mathbf{l}_1, \ldots, \mathbf{l}_M$ given all previous motions $\mathbf{u}_1, \ldots, \mathbf{u}_t$ and observations $\mathbf{z}_1, \ldots, \mathbf{z}_t$. Various approaches to estimate this posterior have been presented in the literature.

In this paper, we address the SLAM problem using the UKF by representing the joint state $(\mathbf{x}_t, \mathbf{l}_1, \ldots, \mathbf{l}_M)$ with $\langle \mu, \Sigma \rangle$. This is a standard approach which has been shown to operate successfully in the past. The mean of the jth landmark location (μ_{2j+2}, μ_{2j+3}) is denoted by $\left(l_x^{[j]}, l_y^{[j]}\right)$. Furthermore, we interpret the state transition function as the robot's motion model and assume that range and bearing observations (r, φ) are given so that we can define a corresponding observation model.

3.4 Reinforcement Learning

In reinforcement learning, an agent seeks to maximize its reward by interacting with the environment [29]. 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 $s_t \rightarrow s_{t+1}$ 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

$$R_t = \sum_{i=t+1}^{T} r_i \ , \tag{2}$$

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

$$\pi(s, a) = p(a|s) \ \ \forall s \in \mathcal{S} \ , \tag{3}$$

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

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \ , \tag{4}$$

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

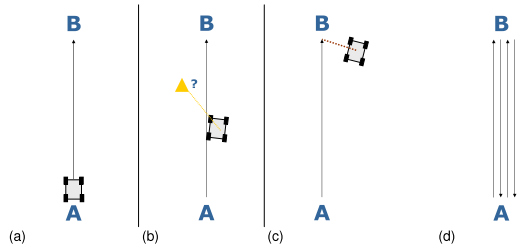


Figure 2. Illustration of the single-goal navigation task (a-c) and the round-trip task (d).

4 LEARNING NAVIGATION POLICIES

In our work, we consider three typical navigation tasks and analyze how to solve them in the reinforcement learning setting. In the first task, the robot has to reach a target location as fast as possible while staying localized using its camera and a given map. To achieve this, the robot has to adapt its travel speed to obtain good feature observations on the one hand, while it has to drive as fast as possible to reach its goal quickly on the other hand. Note that this single goal navigation task can be easily extended to a multi-waypoint path following task. Such a path of waypoints may be given to the robot by a higher level task planner or a path planner such as A^*.

The second and third task address the problem of navigating in an environment without a known map towards a given (relative) location and to perform round-trip navigation tasks, respectively. Here, the challenge is to select a good subset of landmarks to solving the SLAM problem while at the same time taking into account the computational constraints of the system.

Because the belief about the robot's state is represented by a probability distribution in the UKF, the system is ideally modeled by a partially observable MDP (POMDP) [27], which requires an explicit modeling of the probability distribution of the state. This makes POMDPs computationally hard to solve and intractable for most real-world tasks. We use the so-called *augmented MDP* [24] as approximation of the POMDP. Hereby, the belief of the state is 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.

4.1 Navigation Tasks

4.1.1 Navigation Task with a Given Map

Let us consider the following most basic navigation task (Fig. 2(a)). The robot is located at a starting position A and is supposed to reach a goal position B. In this first setting, the robot is supposed to have a map of the environment, that means it knows where the individual landmarks it can observe are located in the environment. However, the robot's motion is affected by drift and the overall motion 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 or stopping from time to time, the negative impact of motion blur can be avoided, but the robot needs more time to finish the navigation task.

Rewards Since we require the robot to reach the goal as quickly as possible, we encode this directly in the reward function. The immediate reward at time t is given as

$$r_t = \begin{cases} C & \text{if } t = T \\ -\Delta_t & \text{otherwise,} \end{cases} \tag{5}$$

where C is some constant, T is the final time step, and Δ_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 in order to maximize its reward.

Note that 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.

Actions The set of available actions in this task is coupled to the mobile robot at hand. On our wheeled robot, we have a basic navigation controller available 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 the desired target velocity v_{target}. Depending on the angle ϕ to the next goal point, the translational and rotational velocities v and ω 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_{target} and ω is set depending on ϕ.

As parameter of the navigation controller which influences the quality of the observed images, we learn the overall velocity limit v_{target} as combination of the translational and rotational velocity. That means that the resulting actions for reinforcement learning can be kept as simple as setting v_{target} (in m/s) to the following values:

$$\mathcal{A} = \{0.1, 0.2, 0.3, 0.4, 1.0\} \ . \tag{6}$$

Regarding the humanoid robot, a discrete set of actions can be directly used to learn the controlling policy to reach the goal fast and reliably. This eliminates the need for a navigation controller that steers the robot to the goal, because the full controller policy is learned. Note that there is still a gait controller running on the humanoid, which translates the walking commands into commands for the joint angles. On our humanoid, we employ the following actions:

- Walk forward: The robot walks 10 cm in forward direction (2 steps).
- Turn left / turn right: The robot turns 23° on the spot in the given direction (2 steps).
- Stand still: The robot interrupts its movement and waits for 0.7 seconds to acquire a good quality image for its localization. This is the time required for the robot's body to stabilize after it has stopped.

We chose these actions since they proved to yield the most reliable and predictable behavior.

4.1.2 *Single-goal Navigation Task Without Known Landmark Locations*

In the second scenario, the landmark locations are not known to the robot and it also has to reach a given location (specified in relative coordinates to the start location). Thus, the robot has to solve a similar task as before but without a map and thus has to estimate the map online (Fig. 2(b-c)) . Here, the problem arises that estimating the map as well leads to a significantly increased overhead in memory and computational load. Thus, the key task of the robot is to select and integrate only landmarks that are useful for the navigation task. We assume that N landmarks are distributed randomly over the environment. When the robot perceives a new landmark, it has to decide whether it should integrate this landmark in the UKF or not. The UKF has a landmark capacity of M landmarks with $M \ll N$.

Rewards The goal is to choose the landmarks in such a way that the distance of the final position of the robot $(x_T, y_T)^\top_{\text{true}}$ and the target position B is minimized. In this scenario, we ignore the impact of the robot's velocity on its perception and the potential problem of missing landmark detections due to motion blur for now. Hence, we define the reward as

$$r_t = \begin{cases} -\left|B - (x_T, y_T)^\top_{\text{true}}\right| & \text{if } t = T \\ 0 & \text{else,} \end{cases} \tag{7}$$

which is the negative Euclidean distance of the robot's true position to the goal B if the training episode reaches the terminal state s_T; intermediate rewards are set to zero. In this task, the terminal state is reached when the robot's estimated position is at the goal B.

Actions In this task, we utilize the existing navigation controller described above with a constant velocity. Thus, the robot only needs to decides whether to integrate a new landmark or not, which is a binary decision:

$$\mathcal{A} = \{a_{\text{reject}}, a_{\text{accept}}\} \tag{8}$$

4.1.3 *Round-trip Task Without Known Landmark Locations*

In the round-trip task, the robot is supposed to reach several subgoals (see Fig. 2 (d)). It starts at A and is supposed to drive to B, back to A and then drive to B and A again. A new subgoal is selected as soon as the position estimate of the robot $(x_t, y_t)^T$ is close to the current subgoal – independent of the robot's true position $(x_t, y_t)^T_{\text{true}}$. In this task, the error in the pose estimate should be minimized over the whole trajectory. For convenience, we specify the return directly as the negative average error over the remaining trajectory

$$R_t = -\frac{1}{|T-t|} \sum_{t'=t}^{T} \left| \begin{pmatrix} x_{t'} \\ y_{t'} \end{pmatrix}_{\text{true}} - \begin{pmatrix} x_{t'} \\ y_{t'} \end{pmatrix} \right| , \tag{9}$$

whereas t specifies the current time and T is the time when the robot reaches its final destination. The actions are identical to the single-goal SLAM task. To simplify things for the second task, landmark selection is only allowed while the robot moves from A to B the first time. The round-trip task is more complex than the previous one. However, it is worth considering since it focuses on the loop-closing problem of SLAM where a robot re-visits previously seen areas in order to correct incremental pose errors. Therefore, this task has a higher practical relevance than the single-goal task.

4.2 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 landmarks and waypoints to reach. 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 a specific navigation task. Based on the current, most-likely pose estimate $\mathbf{x}_t = (x_t, y_t, \theta_t)^\top$ and the environment, we define the following features:

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

$$d = \sqrt{(g_x - x_t)^2 + (g_y - y_t)^2}. \tag{10}$$

- The angle relative to the next goal point

$$\phi = \operatorname{atan2}(g_y - y_t, g_x - x_t) - \theta_t. \tag{11}$$

  In combination with d, this completely characterizes the relative position of the next goal point which has to be reached. In 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:

$$h = \frac{1}{2} \ln \left((2\pi e)^3 \left| \det \left(\Sigma^{3\times 3} \right) \right| \right). \tag{12}$$

  This measures how well the robot is localized: A higher entropy corresponds to a higher pose uncertainty.

In addition, the following features are relevant in the context of landmark integration in SLAM:

- The angle φ_l to the potential new landmark $l^{[\text{new}]}$.
- The number of landmarks already integrated in the UKF

$$m = \left| \{ j \in M : \Sigma_{2j+2} < \infty \wedge \Sigma_{2j+3} < \infty \} \right|, \tag{13}$$

  where Σ_{2j+2} and Σ_{2j+3} are the variances of the jth landmark in the x and y direction.
- The distance of the potential new landmark to the closest landmark already integrated

$$d_l = \min_{\substack{j \in L \text{ with} \\ \Sigma_{2j+2} < \infty \wedge \Sigma_{2j+3} < \infty}} \left| \begin{pmatrix} l_x^{[j]} \\ l_y^{[j]} \end{pmatrix} - \begin{pmatrix} l_x^{[\text{new}]} \\ l_y^{[\text{new}]} \end{pmatrix} \right|. \tag{14}$$

For the localization task with motion blur (see Sec. 4.1.1), we found the features d, ϕ, 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.

In the single-goal and round trip SLAM tasks, we use a combination of all of the above features, and additionally evaluate the effectiveness of including the entropy in the state space.

Since the state space of the features is usually continuous, we need to estimate the Q-function with some function approximator. Either k-nearest neighbor (k-NN) regression [26] or radial basis function (RBF) networks [7] yielded good results in our experiments. In contrast to a strictly discrete representation as feature table, these methods suffer less from the effects of discretization.

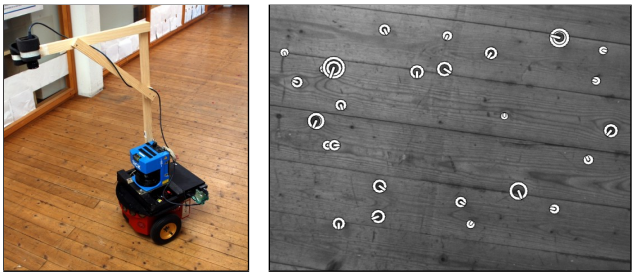


Figure 3. Pioneer 2-DX8 robot in the experimental indoor environment (left) and an observed floor patch with SURF as visual landmarks (right).

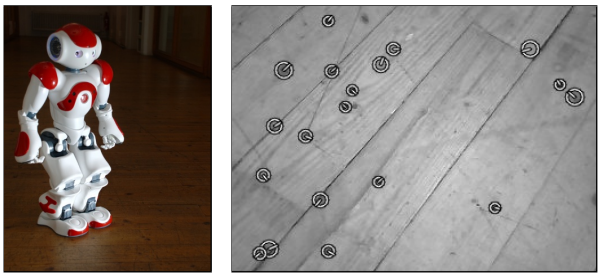


Figure 4. The *Nao* humanoid robot [1] in the experimental indoor environment (left) and an observed floor patch with SURF as visual landmarks (right).

5 EXPERIMENTS

5.1 Navigation Policy for a Known Map

We evaluated our approach for known environments on a wheeled *Pioneer* robot (Fig. 3) as well as on our *Nao* humanoid robot (Fig. 4). The wheeled robot was equipped with a top-mounted camera observing the floor in front of it. Additionally, it carries a laser range finder for obstacle avoidance and to provide a ground truth pose estimate for evaluation.

The humanoid robot is equipped with two small cameras of webcam quality. One of them points to the ground in front of the robot, which we use for localization. In addition, Nao has two ultrasound sensors which can be used for obstacle avoidance. Since the humanoid has no knowledge about its true pose, we use a special marker to allow the robot to identify when it has reached the goal location. This artificial landmark can be reliably detected even while the robot is walking.

5.1.1 Learning the Navigation Policy in Simulation

The policy was learned in simulations. This allowed us to evaluate different parameter settings for the learning algorithms and to run a large number of learning and testing episodes without putting too much strain on the real robots. Each simulated robot and its environment are modeled as close to reality as possible. This includes the motion noise of the robots with a systematic drift to the left or right. We use a map of artificial landmarks whose positions are randomly distributed. To avoid an adaption of a robot's behavior to a specific environment, landmark positions and the direction of the systematic motion error were randomized in each new learning and evaluation episode.

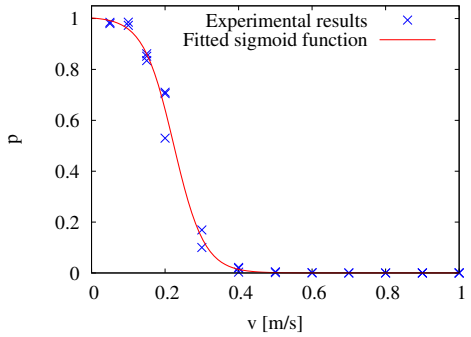


Figure 5. Experimentally determined observation model $p(z|v)$ for a given velocity v of a wheeled robot in our indoor environment. The measured data (blue crosses) are approximated by a sigmoid function (red line).

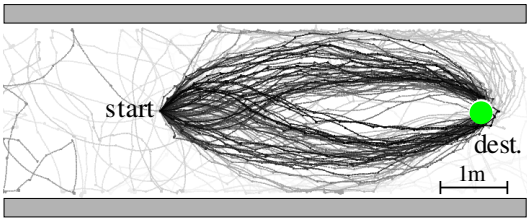


Figure 6. Evolution of the trajectory followed by the humanoid throughout the 1500 learning episodes (each 10th episode is drawn). At the beginning, random exploratory actions are chosen and the robot does not reach the goal within a maximum of 700 seconds (light gray trajectories). Towards the end, the robot navigates successfully and efficiently towards the destination (black trajectories). Note that there is a high noise in the executed motion commands.

In order to obtain a policy which takes motion blur into account, we modeled motion blur in the simulation environment 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 and is approximated by a sigmoid function (Fig. 5). On the humanoid, the amount of motion blur depends on the executed motion command and the current phase of the walking cycle. As we are not able to accurately synchronize the image acquisition with the walking cycle, we use the average observation probability for each walking motion instead.

Note that we model motion blur as effect on the observation probability only in the simulation environment, it is not part of the learning state space or the robot's state estimate. Instead, the robot learns about this effect while interacting with the environment.

Figure 6 shows the evolution of the humanoid's behavior throughout the learning process. As can be seen, in the beginning the robot chose random exploratory actions. It did not reach the goal so that the episodes were aborted after a maximum of 700 seconds (the resulting trajectories are colored light gray). After a certain number of learning trials, however, the robot successfully navigated towards the destination (dark gray / black trajectories). The trajectories were getting more and more efficient towards the end of the learning process. Note that the robot had a systematic error in the executed motion command in each of the episodes.

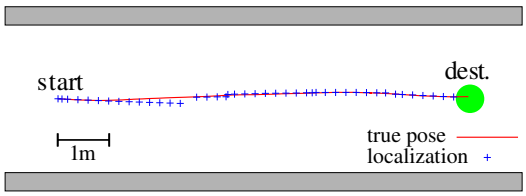


Figure 7. A typical example of the executed trajectory of a learned policy. The corresponding state space is displayed in Fig. 8.

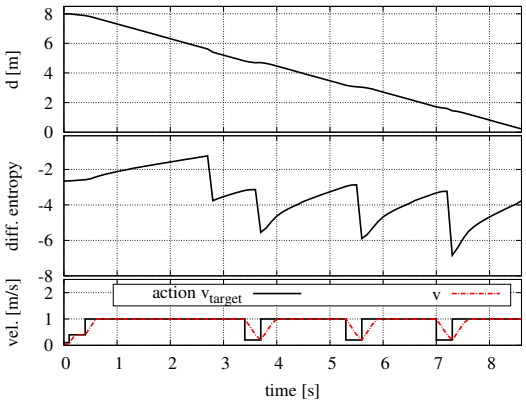


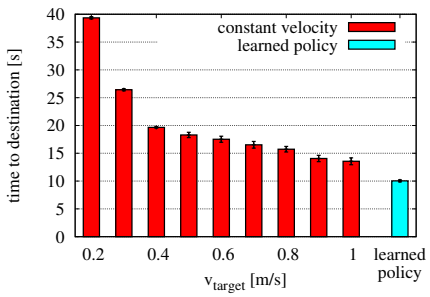
Figure 8. A typical example of the learned policy for a wheeled robot with the dimensions *distance* and *entropy* of the state space. The corresponding trajectory is displayed in Fig. 7. The robot maximizes its velocity until its uncertainty gets too high, indicated by a high value of the differential 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.

5.1.2 *Evaluation of the Learned Policy*

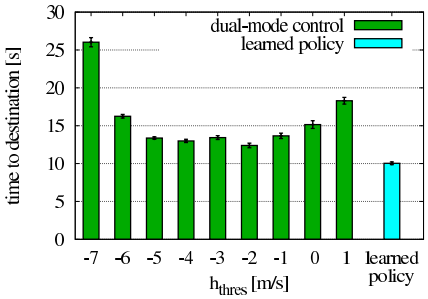
A typical trajectory and the corresponding state space over time of the learned policy for a wheeled robot are displayed in Fig. 7 and 8. The robot optimizes its time to reach the destination by driving at maximum speed as long as it is confidently localized. When there is risk of getting lost, indicated by a high entropy, it slows down in order to observe landmarks. 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. Overall, the robot stays close to the direct connection between start and destination.

Comparison to Constant Velocity A standard approach for a wheeled robot is to set a constant target velocity v_{target}. Figure 9(a) 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 collisions 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.

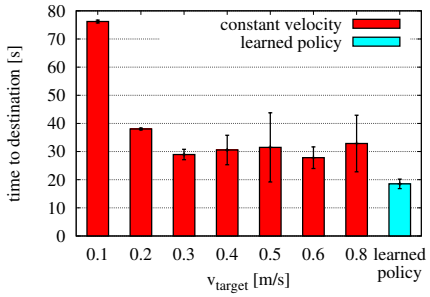
But even when choosing the best policy of constant velocity, our learned approach is significantly better. While the average time to destination at 1 m/s is 13.56 s $\pm$ 0.61 s (95% confidence interval),



(a) Constant velocity in the simulated scenario (100 runs).



(b) Dual-mode control policies at various thresholds for the entropy in the simulated scenario (100 runs).



(c) Constant velocity in the real indoor scenario (10 runs).

Figure 9. Comparison of constant velocity policies and the dual-mode controller to our learned policy in known environments. Each policy is displayed with mean and 95% confidence interval. The learned policy is significantly better than each other policy.

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%.

Comparison to Dual-Mode Controllers A more advanced approach is to employ a *dual-mode controller* as introduced by Cassandra *et al.* [5]. Similar to our learned policy, the entropy is used to decide on which action to take. When the entropy is above a threshold h_{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 9(b) displays the resulting times for various values of h_{thres} compared to the learned policy on the wheeled robot.

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%.

Additionally, we evaluate the performance of a policy learned using our approach for the humanoid robot in comparison to a dual-mode control policy. This dual-mode policy controls the robot to walk forward while the estimated orientation towards the goal is smaller than some threshold $\hat{\varphi}$. Whenever the estimated angular distance to the goal is larger than $\hat{\varphi}$, the robot stops its forward motion and turns towards to goal. We chose the threshold of $\hat{\varphi} = 35°$ since smaller values lead to an oscillating behavior of the robot near the destination, whereas larger values lead to frequent collisions with the walls bounding the corridor. Whenever the uncertainty about its pose exceeds a threshold, the robot stops in order to obtain a good quality image. This threshold was empirically determined to minimize the time to reach the destination while still achieving a success rate of 100%. Thus, we optimized the parameters of this dual-mode controller so as to perform best in our test environment.

It took the humanoid robot $117.18\,\text{s} \pm 8.41\,\text{s}$ to reach the destination with the hand-optimized controller, and only $106.66\,\text{s} \pm 10.05\,\text{s}$ using our learned policy. A t-test with 95% confidence reveals that the learned policy performs significantly better.

5.1.3 *Verification on Real Robotic Systems*

We now transfer the results from simulations into the real world by applying the policy learned in simulation on real robots.

Wheeled Robot We first employ the Pioneer robot (Fig. 3) in an indoor environment. Each policy is evaluated in 10 test runs, each consisting of navigating from the start location to the destination. The resulting navigation times are shown in Fig. 9(c).

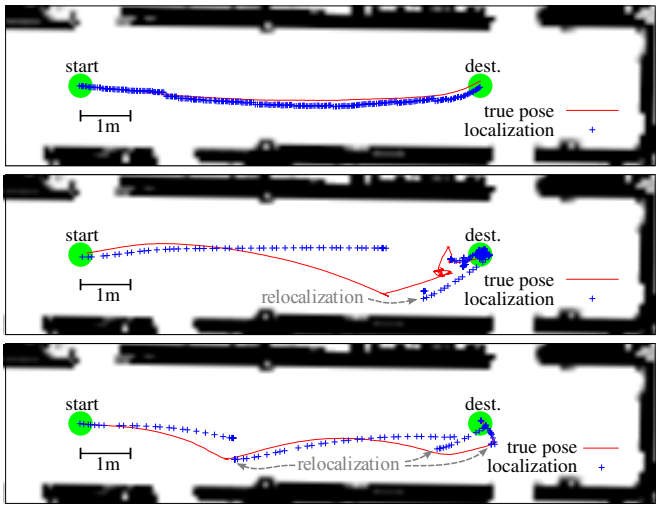


Figure 10. Comparison of real robot trajectories at constant velocity (top: 0.2 m/s, middle: 0.8 m/s) and variable velocity, i.e., following the learned policy (bottom).

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. 10). 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. Note that there is a systematic drift to the right in the robot's motion. Contrary to that, the robot does not need to be stopped by the obstacle avoidance while following the learned policy. When it in danger of getting lost, it immediately slows down to re-localize. As a result, the robot reaches its destination reliably and quickly.

Humanoid Robot Finally, we performed experiments with our real humanoid (Fig. 4) navigating in our hallway environment. We

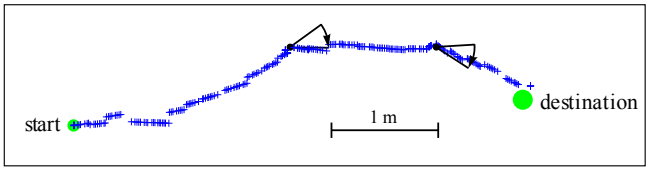


Figure 11. Estimated trajectory of the humanoid while executing the learned navigation policy in our hallway. The robot walks forward with an error resulting from a drift to the left. Whenever it seems appropriate according to its belief, the robot executes a turning action to re-align with the goal. The robot stops as soon as it recognizes the goal landmark.

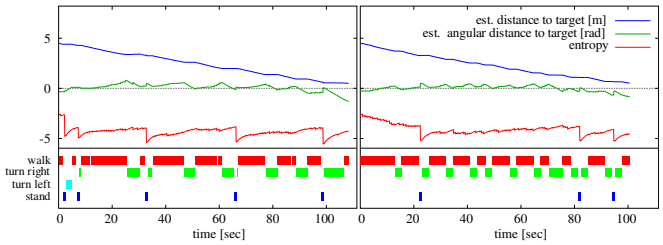


Figure 12. The state features over time during typical runs with a hand-optimized controller (left) and the learned navigation policy (right) in simulation. The hand-optimized controller stops the robot in regular intervals to decrease the uncertainty, whereas the learned policy adapts the observation frequency according to the current state.

conducted test runs both with the hand-optimized controller and with the policy learned in simulation. The robot needs 93.16 s $\pm$ 10.14 s using the hand-optimized controller compared to 86.53 s $\pm$ 8.60 s using the learned policy, so the learned policy outperforms the hand-optimized controller by 7.1%. Again, a t-test with 95% confidence shows that the learned policy performs significantly better.

Figure 11 depicts a typical trajectory of the Nao robot while executing the learned navigation strategy (the drawn poses were estimated by the localization system). As can be seen, the robot rotates from time to time to compensate for its motion drift and to re-align with the goal. This learned policy is not adapted to the specific drift direction.

Note that in these experiments, we used slow walking patterns as the Nao's stability was highly reduced when walking faster in the current implementation. Accordingly, the acquired images are only moderately blurred. The robot can still match an average of 3.25 features per frame while moving, and actions to reduce the uncertainty are rarely executed. Thus, the efficiency gain of the learned controller compared to the hand-optimized controller results mainly from choosing the navigation actions more foresightedly, which leads to shorter paths.

In future implementations, we will optimize the humanoid's gait, so faster walking patterns will be used. While this will enable the humanoid to potentially reach its goal faster, it also increases the amount of motion blur, thus seriously reducing the average number of successfully matched features while moving. To evaluate the impact of motion blur, we learned policies for a different set of estimated observation probabilities in the simulator, i.e., we decreased the probability of a successful feature match by 80% during walking and turning.

Again, we compared the learned policy to a hand-optimized dual-mode controller that stops the humanoid whenever the entropy exceeds a fixed threshold. For each of the different observation probabilities, we selected the hand-optimized controller leading to the smallest average time to destination while still achieving a success rate of 100 %. The results show that the learned policy is significantly faster (9% gain) than the hand-optimized policy.

Figure 12 shows the state space of two typical runs with the hand-optimized controller (left image) and the learned policy (right image). The hand-optimized controller stops the robot in regular intervals to obtain good observations. In contrast to that, the learned policy accepts higher uncertainties as long as the distance to the destination is high, whereas it increases the robot's stand frequency when approaching the destination.

5.2 Landmark Selection Policy for Navigation in Unknown Environments

We evaluate the performance of our learned landmark selection policies in the single-goal and round trip SLAM scenarios on our wheeled *Pioneer* platform (Fig. 3), first in simulations and then on the real robot.

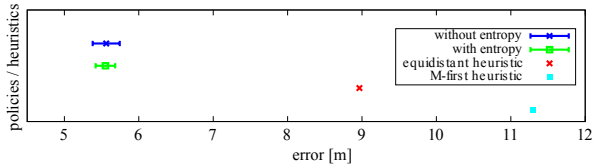


Figure 13. Average performance of the learned policies and heuristics w.r.t. 1,000 test episodes in the single-goal SLAM task. For the learned policies, the mean over ten training runs as well as the corresponding 95%-confidence interval is shown.

5.2.1 *Single-goal Task in Simulation*

For the single-goal task in simulation, we choose an environment where N landmarks are randomly distributed in a 30 m by 60 m area. The distance between the start position A and the goal B is set to 44 m. We train our policy for 1,000 episodes. In each episode, landmarks are randomly re-distributed. We compare the trained policies with two heuristics. The first one is the *M-first heuristic* which simply integrates the M first landmarks that are observed. An apparently better policy is the *equidistant heuristic*. With this heuristic, the robot only integrates a new landmark after it has driven a certain distance so that the landmarks are approximately uniformly distributed over the whole trajectory (similar to [6]).

At first, we consider an UKF with a landmark capacity of $M = 10$ and an environment with $N = 50$ landmarks. For each learning approach, ten training runs are performed. Each trained policy and heuristic is evaluated in 1,000 different environments (see Fig. 13). The one-sample t-test at 95% confidence shows that all three learning approaches are significantly better than the equidistant heuristic.

A notable fact is that in this setting, we were not able to show that there is any benefit from including the feature of the entropy h of the robot's pose in the state space. Even at 75% confidence, the t-test did not reveal a difference between the learning approach using the entropy compared to the setting where it is ignored. One reason why the current entropy of the robot's pose is not a good indicator of

whether to integrate a landmark or not in the SLAM task is that landmarks are integrated with an uncertainty over the robot's pose. That means that the robot is not able to reduce its uncertainty immediately after integrating a landmark. The relative position of the landmark, for example, is a better indicator on how the robot will perform in reaching the goal.

In order to evaluate how good the trained policies generalize, we trained and tested a policy in environments with $N = 50$ as well as $N = 100$ landmarks. In addition, we use UKFs with a capacity M of five, ten, and 15 landmarks. Fig. 14 (a) illustrates the high degree of generalization of our learning approach. For instance, if we perform a training in a setting with $N = 50$ and $M = 5$, we see that the trained policy leads to significantly better results than the equidistant heuristic in all six test scenarios. This indicates that our approach generalized over different landmark densities which is similar to environments of different scale and sensor range.

5.2.2 *Single-goal Task Performed in a Real World Experiment*

Furthermore, we evaluated our landmark selection learning approach in the real experimental environment. Similarly to Sec. 5.1, we use a pioneer robot equipped with a camera and laser range finder in a hallway environment. Learning the policy in this real-world environment would be impractical because this would not only require us to perform hundreds of training episodes but also to install different landmark distributions for each training episode. Thus, we trained the policy in simulation and tested it in the real-world setting. We also compared the trained policy to the equidistant heuristic. Both the trained policy as well as the equidistant heuristic were tested ten times. The trained policy results in an error of 0.50 ± 0.08 m whereas the equidistant heuristic leads to an error of 0.66 ± 0.07 m. Hence, the trained policy is significantly better than the equidistant heuristic (w.r.t. a t-test at 95% confidence).

5.2.3 *Round-trip Task*

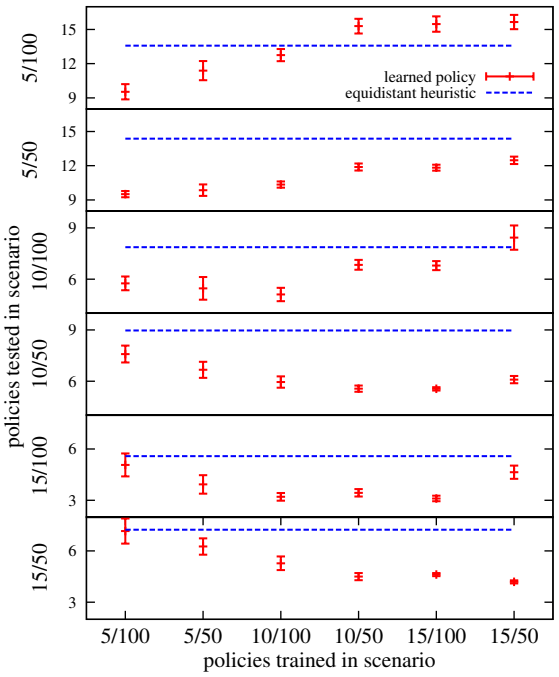
The performance of our learning procedure for the round-trip task is evaluated in a simulated environment with a wheeled robot, similar to the single-goal task. The error evaluation, however, differs since the average localization error over the whole trajectory was considered here to provide a better performance when approaching also the intermediate goal. Again, we compare our learning with the equidistant heuristic. Fig. 14 (b) shows that the learned policy is significantly better than the heuristic. Furthermore, it is shown that we were able to generalize over the UKF capacity M as well as the number of landmarks N.

6 CONCLUSION

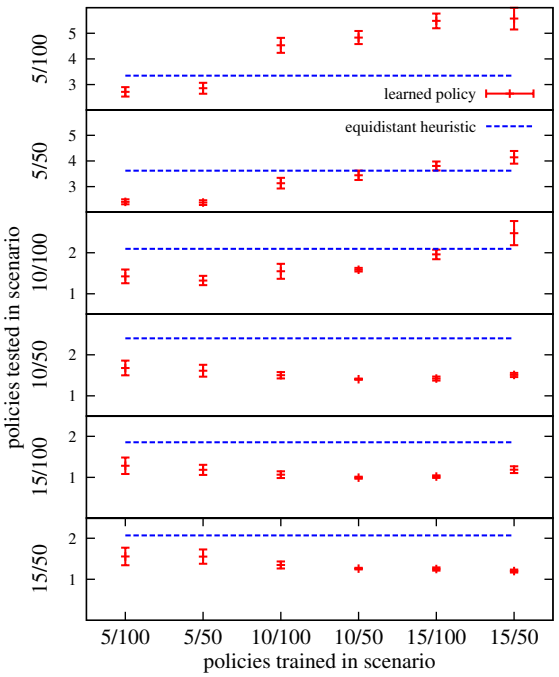
In this paper, we presented a novel approach to learning efficient navigation policies for mobile robots that are constrained in their sensing capabilities as well as in their computational resources. We considered navigations tasks in known as well as unknown environments. By considering these navigation problems as reinforcement learning tasks, the robot can learn policies for choosing appropriate actions.

In case of navigating in known environments, the robot is able to select the optimal velocity so that it reaches its target location as fast as possible and with minimum error.

For navigation in unknown environments, the map has to be estimated as well to navigate efficiently. This task, however, requires significant computational resources. We presented an approach to learn



(a) Single-goal task



(b) Round-trip task

Figure 14. High degree of generalization in the single-goal task (a) and the round trip task (b) in unknown environments. The mean error over ten training runs and the corresponding standard derivation is shown. All policies below the dashed horizontal line are significantly better than the equidistant heuristic ($\alpha = 0.05$).

an efficient landmark selection policy. The ability of a mobile robot to decide which landmark to incorporate into its belief given the navigation task at hand allows for navigation under computational constraints. The presented method is able to determine which landmark is valuable for the robot to efficiently solve its current navigation task.

In a series of real-world and simulated experiments with wheeled robots and a humanoid, we demonstrated that our learned navigation policies significantly outperform strategies using advanced and manually optimized heuristics.

ACKNOWLEDGEMENTS

This work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and within the Research Training Group 1103.

REFERENCES

[1] Aldebaran Robotics. The Nao humanoid robot. http://www.aldebaran-robotics.com/en/. Retrieved June 2010.

[2] H. Bay, T. Tuytelaars, and L. V. Gool, 'SURF: Speeded-up robust features', *Proc. of the ninth European Conf. on Computer Vision*, (2006).

[3] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke, 'Metric localization with scale-invariant visual features using a single perspective camera', in *European Robotics Symposium 2006*, ed., H. Christiensen, volume 22 of *STAR Springer tracts in advanced robotics*, (2006).

[4] M. Bryson and S. Sukkarieh, 'Active airborne localisation and exploration in unknown environments using inertial SLAM', in *Proc. of the IEEE Aerospace Conference*, (2006).

[5] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, 'Acting under uncertainty: Discrete bayesian models for mobile-robot navigation', in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, (1996).

[6] G. Dissanayake, H. Durrant-Whyte, and T. Bailey, 'A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem', in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, pp. 1009–1014, (2000).

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

[8] R. He, S. Prentice, and N. Roy, 'Planning in information space for a quadrotor helicopter in a GPS-denied environments', in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (2008).

[9] A. Hornung, M. Bennewitz, and H. Strasdat, 'Efficient vision-based navigation – Learning about the influence of motion blur', *Journal of Autonomous Robots*, **29**, 137–149, (August 2010).

[10] A. Hornung, H. Strasdat, M. Bennewitz, and W. Burgard, 'Learning efficient policies for vision-based navigation', in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, (2009).

[11] A. Hornung, K. M. Wurm, and M. Bennewitz, 'Humanoid robot localization in complex indoor environments', in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, (2010). Accepted for publication.

[12] V. A. Huynh and N. Roy, 'icLQG: Combining local and global optimization for control in information space', in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, (2009).

[13] J. Ido, Y. Shimizu, Y. Matsumoto, and T. Ogasawara, 'Indoor Navigation for a Humanoid Robot Using a View Sequence', *The International Journal of Robotics Research*, **28**(2), 315–325, (2009).

[14] S. J. Julier and J. K. Uhlmann, 'A new extension of the Kalman filter to nonlinear systems', in *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, pp. 182–193, (1997).

[15] T. Kollar and N. Roy, 'Using reinforcement learning to improve exploration trajectories for error minimization', in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (2006).

[16] C. Kwok and D. Fox, 'Reinforcement learning for sensing strategies', in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, (2004).

[17] R. Lerner, E. Rivlin, and I. Shimshoni, 'Landmark selection for task-oriented navigation', *IEEE Transaction on Robotics*, **23**(3), (2007).

[18] J. Michels, A. Saxena, and A. Y. Ng, 'High speed obstacle avoidance using monocular vision and reinforcement learning', in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pp. 593–600, New York, NY, USA, (2005). ACM.

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

[20] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, 'FastSLAM: A factored solution to the simulaneous localization and mapping problem', in *Proc. of the National Conf. on Artificial Intelligence (AAAI'02)*, pp. 593 – 598, (2002).

[21] S. Oßwald, A. Hornung, and M. Bennewitz, 'Learning reliable and efficient navigation with a humanoid', in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (2010).

[22] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello, 'A visual odometry framework robust to motion blur', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, (2009).

[23] N. Roy, W. Burgard, D. Fox, and S. Thrun, 'Coastal navigation–mobile robot navigation with uncertainty in dynamic environments', in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (1999).

[24] N. Roy and S. Thrun, 'Coastal navigation with mobile robots', in *Advances in Neural Processing Systems 12 (NIPS)*, volume 12, (1999).

[25] P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson, 'Landmark selection for vision-based navigation', *IEEE Transaction on Robotics*, **22**(2), (2006).

[26] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press, Cambridge, MA, USA, 2006.

[27] E. J. Sondik, *The optimal control of partially observable Markov decision processes*, Ph.D. dissertation, Stanford University, Stanford, USA, 1971.

[28] H. Strasdat, C. Stachniss, and W. Burgard, 'Which landmark is useful? Learning selection policies for navigation in unknown environments', in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (2009).

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning, The MIT Press, March 1998.

[30] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, 'Simultaneous localization and mapping with sparse extended information filters', *Int. Journal of Robotics Research*, **23**, (2004).

[31] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, September 2005.

[32] S. Zhang, L. Xie, and M. Adams, 'Entropy based feature selection scheme for real time simultaneous localization and map building', in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, (2005).