Learning Efficient Policies for Vision-based Navigation

Armin Hornung Hauke Strasdat Maren Bennewitz Wolfram Burgard

Abstract— **Cameras are popular sensors for robot navigation tasks such as localization as they are inexpensive, lightweight, and provide rich data. However, fast movements of a mobile robot typically reduce the performance of vision-based localization systems due to motion blur. In this paper, we present a reinforcement learning approach to choose appropriate velocity profiles for vision-based navigation. The learned policy minimizes the time to reach the destination and implicitly takes the impact of motion blur on observations into account. To reduce the size of the resulting policies, which is desirable in the context of memory-constrained systems, we compress the learned policy via a clustering approach. Extensive simulated and real-world experiments demonstrate that our learned policy significantly outperforms any policy that uses a constant velocity. We furthermore show, that our policy is applicable to different environments. Additional experiments demonstrate that our compressed policies do not result in a performance loss compared to the originally learned policy.**

I. Introduction

Autonomous navigation is one of the most essential tasks for a mobile robot. Hereby, the robot needs to find its way to the destination or to a certain waypoint on its path. For this, it is crucial that the robot knows its position. Cameras are frequently used sensors for localization, especially in the case of small robots such as humanoids or unmanned aerial vehicles (UAVs) because they are compact and lightweight. However, the movements of the vehicle 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. Typical images of the floor recorded with a downward-looking camera on a wheeled mobile robot moving at different speeds are depicted in Fig. 1. As can be seen, with an increasing velocity the features on the ground become more and more difficult to recognize. While there are methods to reduce the influence of motion blur [1], the problem in general remains since the degradation obtained by motion blur cannot be completely removed by filtering techniques.

In this paper, we present a new approach to vision-based navigation that implicitly takes motion blur into account. Our method applies reinforcement learning (RL) to learn how fast the robot should move to reach its destination as fast as possible and with appropriate accuracy. We use an unscented Kalman filter (UKF) to track the pose of the robot and include the uncertainty of the filter, measured by the entropy, as one component of the state representation of the Markov decision process (MDP) modeling the navigation task, thus leading to an augmented MDP [2]. Additionally, the state representation contains the distance and relative angle to the goal location. The action space of the augmented MDP consists of setting the velocity of the robot. We apply Sarsa(λ) RL [3] to determine the optimal action for each state and provide experiments to evaluate the learned policy in both simulated and real-world experiments. The experiments demonstrate that the learned policy significantly outperforms any constant velocity policy and additionally can be transferred to different environments.

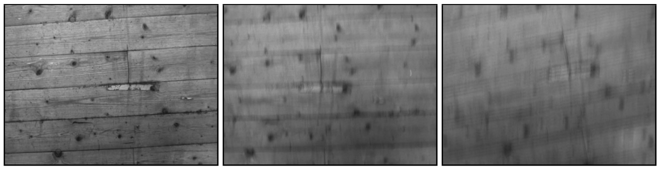

Fig. 1. The same floor patch observed at different velocities (0.05 m/s, 0.4 m/s, 1.0 m/s), with motion blur of different magnitude.

Furthermore, we developed a method of compressing the policy using a clustering approach, which reduces the size of the policy representation by one order of magnitude. By employing X-means clustering [4] on the visited state space when following the learned policy, the problem can be regarded as a classification problem with the actions as the individual classes. The cluster means can then be used for a nearest-neighbor classification on which action to take. As the experiments indicate, the compressed policy does not lead to a loss of performance.

The contribution of our work is a system that allows for implicitly learning about the effects of motion blur on the observations during fast movements. In contrast to previous works that aim at minimizing the uncertainty in the belief distribution (e.g., [5], [6], [7]), our approach enables the robot to choose a navigation strategy that minimizes the time to reach the destination. Thereby, the robot avoids delays caused by localization errors. 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 and at the same time reliably.

The remainder of this paper is structured as follows. We first discuss related work in the next section, followed by the preliminaries on state estimation and reinforcement learning in Sec. III. We present the navigation task in Sec. IV and our learning approach in Sec. V. Policy compression is discussed in Sec. VI. Finally, in Sec. VII, we present and discuss the experimental setup and the results.

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

A. Hornung, M. Bennewitz, and W. Burgard are with the Department of Computer Science, University of Freiburg, Germany. H. Strasdat is with the Department of Computing, Imperial College London, UK.

II. Related Work

In the past, various frameworks were presented which employ active methods in the context of localization and navigation. Kollar and Roy [6] use reinforcement learning to optimize a robot trajectory for exploration. Similar to our approach, the authors learn the behavior of the controller to navigate to a destination. While we consider the problem of motion blur in vision-based localization, they learn the translation and rotation behavior which minimizes the uncertainty in laser-based SLAM (simultaneous localization and mapping). A different method of minimizing the uncertainty of localization is to plan a path for the robot which takes the information gain into account. Roy *et al.* [7] presented an approach for this called *coastal navigation*. Recently, He *et al.* [5] applied this technique to a quadrotor helicopter for indoor navigation with a short-range laser range finder. Bryson and Sukkarieh [8] suggested a framework for UAV localization and exploration in unknown environments. They also use an intelligent path planning scheme to maximize the quality of the resulting SLAM estimate.

Strasdat *et al.* [9] developed a reinforcement learning approach for the problem of landmark selection in SLAM. They learn a policy on which new landmark to integrate into the environment representation to improve the navigation capabilities of the robot. The motivation behind their work is the application on memory-constrained systems. Kwok and Fox [10] apply reinforcement learning to increase the performance of soccer-playing robots by active sensing. A camera is used for the localization of relevant objects on the soccer field. As in our approach, the authors model the uncertainty in the belief distribution explicitly as entropy in the augmented state of the underlying MDP.

Bennewitz *et al.* [11] developed a localization method based on visual features and presented experiments with a humanoid robot. The authors mention the impact of motion blur on feature extraction, but do not address the problem specifically. Instead, the robot stops moving at fixed time intervals to observe the environment. To overcome the problem of motion blur in the context of humanoid robots, Ido *et al.* [12] explicitly consider the shaking movements of the head and acquire images only during stable phases. Since humanoid robots suffer a lot from blurred images, it would be interesting to apply our approach of learning how fast to move in which situation to these types of robots.

Pretto *et al.* [1] propose an additional image processing step prior to feature extraction, in particular for humanoid robots. 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.

To the best of our knowledge, we present the first approach which considers the effect of motion blur on observations and generates a policy to reach the destination reliably and as fast as possible.

III. Preliminaries

A. Unscented Kalman Filter

The unscented Kalman filter (UKF) is a recursive Bayes filter to estimate the state $\mathbf{x}_t$ of a dynamic system [13]. This state is represented as a multivariate Gaussian distribution $N(\mu, \Sigma)$. The estimate is updated using controls u_t and measurements 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 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.

B. Reinforcement Learning

In reinforcement learning, an agent seeks to maximize its reward by interacting with the environment. 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 gets a reward $r_{t+1} \in \mathcal{R}$. The overall goal of the agent is to maximize its return

$$R_t = \sum_{i=t+1}^{T} r_i \ , \qquad (1)$$

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 *action-value function*, also called *Q-function*, for the policy π is defined as

$$Q^\pi(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\} \ , \qquad (2)$$

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

Sarsa is a temporal difference (TD) learning algorithm, combining the advantages of bootstrapping and sampling methods [14]. Sarsa is model-free, which means that transition probabilities between states do not need to be defined. The estimate of the Q-function is continuously updated. $Q^\pi(s,a)$ is learned on-policy, meaning the current behavior policy π is learned and updated. The current estimate is updated based on its old values, the new reward, and the new value:

$$\begin{aligned} Q(s_t,a_t) &\leftarrow Q(s_t,a_t) \\ &\quad + \alpha\big(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\big) \end{aligned} \qquad (3)$$

The parameters α and γ 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 an extension of Sarsa that averages over a number of future rewards and which is called Sarsa(λ) [3]. The parameter $\lambda \in [0, 1]$ determines the decay of the impact of future rewards.

C. Augmented Markov Decision Process

In an MDP, it is assumed that the agent is able to infer its state deterministically. If the belief about the state is presented by a probability distribution, the system is ideally modeled by a partially observable MDP (POMDP, [15]). Since POMDPs require to model the probability distribution explicitly, they are computationally hard to solve and impractical for many real-world tasks. In cases where the underlying distribution can be modeled by a unimodal distribution, the so-called *augmented MDP* [2] can be used as an efficient approximation. The belief of the state is represented by its most-likely estimate and the task is modeled as MDP. As measure of the uncertainty of the belief distribution, the entropy is included in the state.

IV. Navigation Task

We consider the scenario of a robot navigating from its current pose to a (intermediate) goal location. We assume that a longer and more complicated trajectory can be planned with a path planner such as A^*, yielding waypoints the robots has to reach on the way to its target location. The task is finished as soon as the distance between the robot's true pose and the goal location is below a certain threshold.

We apply a straightforward controller to steer the robot to the destination, based on the current pose estimate and the desired target value v_{target} for the velocity.

The velocity influences the visual perception of the robot because the viewed scene is affected by motion blur. The faster the robot moves, the more its visual perception is influenced, with a direct impact on feature extraction and matching. By selecting a low value for v_{target}, the negative impact of motion blur can be avoided, but the robot needs more time to finish the navigation task.

V. Learning Navigation Policies

We formulate this problem as a reinforcement learning task. As learning algorithm, we employ Sarsa(λ). Throughout the learning phase, we use an ε-greedy action selection (ε = 0.1). This selection method chooses all non-greedy actions with equal probability.

We represent the learning task as an augmented MDP, whereas the probability distribution over the robot 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.

A. State Space

The complete state of the robot consists of the global pose estimate $\mathbf{x}_t$ modeled as Gaussian distribution (μ, Σ), the current velocity, and a characterization of the environment. However, this complete state representation would be impractical to consider for reinforcement learning. Thus, we define a set of features, which characterizes the complete state sufficiently detailed and general as needed for learning. Based on the current, most-likely pose estimate $\mathbf{x}_t = (x_t, y_t, \theta_t)$, we define the following features:

- The Euclidean distance to the goal $(g_x, g_y)^T$
$$d = \sqrt{(g_x - x_t)^2 + (g_y - y_t)^2}\ . \tag{4}$$
- The angle relative to the goal
$$\varphi = \operatorname{atan2}(g_y - y_t, g_x - x_t) - \theta_t\ . \tag{5}$$
In combination with d, this completely characterizes the goal location.
- The uncertainty of the localization, computed as entropy over the pose covariance Σ of the UKF:
$$h = \frac{1}{2} \ln \left((2\pi e)^3 \cdot |\det(\Sigma)| \right)\ . \tag{6}$$
This measures how well the robot is localized: A higher entropy corresponds to a higher pose uncertainty.

We experimentally found these 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.

We represent the state-action space by a radial basis function (RBF) network, which is a linear function approximator [16]. 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. We experimentally determined a sufficient, small representation with 120 discretization steps (d: 10, φ: 3, h: 4).

B. Action Set

The behavior of the robot can be influenced by restricting the overall velocity of the controller to a target velocity v_{target} (in m/s). This is the action the agent learns via reinforcement learning. We define the possible actions as

$$\mathcal{A} = \{0.1, 0.2, 0.3, 0.4, 1.0\}\ . \tag{7}$$

This discretization was determined according to the effect of motion blur on the landmark observation probability, which we observed in experiments. Velocities higher than 0.4 m/s blur the image almost beyond recognition.

C. Reward

We define the immediate reward at time t as

$$r_t = \begin{cases} 100 & \text{if } t = T \\ -\Delta_t & \text{otherwise}\ , \end{cases} \tag{8}$$

where T is the final time step and Δ_t is the time interval in 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 agent 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 short range 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.

VI. Policy Compression

A policy learned using the framework described above is represented by a huge table that contains for each given state tuple (d, φ, h) of distance to destination, angle, and entropy, the optimal target velocity v_{target}. Especially in the context of systems with memory constraints such as embedded systems it is often desirable to find a more compact representation of the learned policy.

While the learned table of the policy contains entries for all values of (d, φ, h) within the discretized ranges, not all of these values are relevant. Some combinations of them were never experienced in the learning phase because they never appear in the task. Thus, we do not use the actual learned table for policy compression, but the visited state space while the robot is following this policy.

A. Formulation as Classification Problem

The robot follows the learned policy for 100 episodes in the learning environment. For each state (d, φ, h), the chosen velocity v_{target} is regarded as classification of the state. Thus, we can treat the problem of finding the best policy as a classification problem, where we have labeled samples available from the executed episodes. This labeled data naturally contains only the entries of the state that are relevant for the task, while more common states appear more frequently in the data.

B. X-means Clustering

We use X-means clustering [4] to find a number of clusters which approximate this data. X-means is an extension of K-means clustering, which finds the best number of clusters according to the Bayesian Information Criterion. Like in K-means, the data is clustered based on the Euclidean distance to the cluster mean.

After the number of clusters and the location of their means is found, each state (d, φ, h) 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, all values are normalized within their discretization range. After all state entries are assigned to the closest cluster, the velocity classification of that cluster is computed as the average velocity of the samples assigned to the cluster. The cluster means – each labeled with a velocity value – can now be used as a compact represention of the learned policy.

VII. Experiments

The practical experiments were conducted on a wheeled Pioneer 2-DX8 robot with a top-mounted ImagingSource DFK 31AF03 camera (shutter speed set to 1/25 s) to observe the floor in front of the robot (Fig. 2). Speeded-Up Robust Features (SURF, [17]) are extracted from the images as visual landmarks. These observations are integrated with odometry in an UKF for localization. Additionally, a SICK laser range finder on-board the robot is used for obstacle avoidance and Monte Carlo localization, providing a ground truth for evaluation. The experimental environment is a hallway with wood parquet floor and a distance of approx. 8 meters between the starting pose of the robot and the destination. A map of this environment with global landmark positions was built prior to the experiments. To make the experiments more challenging, we introduced a systematic error on the odometry by slightly deflating one tire of the robot.

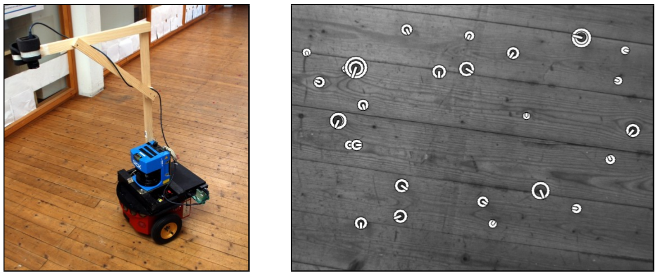

Fig. 2. Pioneer 2-DX8 robot in the experimental environment (left) and an observed floor patch with SURF as visual landmarks (right).

The policy is learned in simulations. This allows us to evaluate different parameter settings for the learning algorithms and to run a large number of learning and testing episodes. Additionally, we have an accurate true pose available as reference. The simulated robot and its environment is modeled as close to reality as possible. Instead of modeling SURF detection and matching explicitly, we use a map of artificial landmarks. The landmark positions are randomly distributed with an average density matching the real map (40 landmarks/m^2). Because we want to avoid an adaption of the robot's behavior to a specific environment, landmark positions are randomized in each new learning and evaluation episode. In order to obtain a policy which takes motion blur into account, we model motion blur as an effect on the probability of an observation z given the current velocity v. This dependency $p(z \mid v)$ was estimated on the real robot in our experimental setting with

$$\lim_{v \to 0} p(z \mid v) = 1 \quad \text{and} \quad \lim_{v \to 1} p(z \mid v) = 0 \,. \tag{9}$$

For the learning parameters, we used $\alpha = 0.2$, $\gamma = 0.95$, and $\lambda = 0.85$. 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 start to destination and a 95% confidence interval. All statements concerning significance are with respect to a t-test with 95% confidence.

A. Comparison to Constant Velocity

We now compare our learned policy with the standard approach of setting a constant target velocity v_{target}. Under no influence of motion blur, the best choice minimizing the time to destination would be the highest possible velocity. In our scenario, however, there is not an immediate benefit of

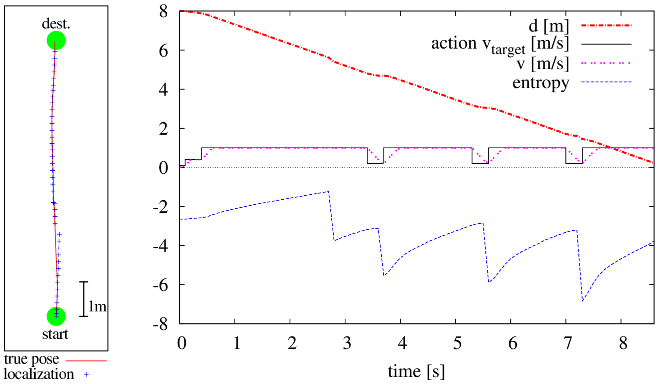

Fig. 3. A typical example of the trajectory (left) and state space over time (right) of a learned policy. For clarity, the angle is not plotted though it is included in the state. The robot maximizes its velocity until its uncertainty gets too high. To re-localize, it slows down. As soon as the entropy decreases as an effect of localization, it accelerates again.

a high velocity because the robot quickly gets lost, leading to a higher time to destination.

1) Qualitative Analysis: Exemplary, a typical trajectory and the corresponding state space over time of the learned policy is displayed in Fig. 3. The robot optimizes its time to 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 to observe landmarks. The trajectory is close to the optimal path of the straight-line connection between start and destination.

2) Quantitative Analysis: Figure 4 displays an extensive evaluation of following a constant velocity from $v_{\text{target}} = 0.2$ m/s to 1 m/s, compared to a 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 and regularly gets lost on its path. Despite this, there is still a small improvement in the average time to destination. The robot would then accept 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), the robot is able to finish the task with our learned policy in 10.04 s $\pm\, 0.18$ s, which corresponds to a reduction of 26%.

B. 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 evaluate policies learned in an environment with an average landmark density of 40 landmarks/m^2 in significantly sparser (10 landmarks/m^2) and denser (70 landmarks/m^2) environments. The performance is compared to learning in an environment matching the respective test environment, which we expect to yield the best results.

In order to obtain general results, we compare 50 independently learned policies here, each learned in 500 learning episodes and evaluated over 100 test episodes. The resulting times are displayed in Tab. I. There is no significant difference between using a constant density during learning and the optimal case. This shows that our learned policy generalizes over different environments, without the need of explicitly accounting for this in the learning scenario.

learning environment	landmark density in test environment		
	10	40	70
const. avg. density	11.19 ± 0.44	10.42 ± 0.24	10.32 ± 0.25
density of test env.	10.78 ± 0.16		10.56 ± 0.18

TABLE I

INFLUENCE OF LEARNING ENVIRONMENT ON PERFORMANCE IN VARIOUS TEST ENVIRONMENTS.

C. Verification on a Real System

We now transfer the results from simulations into the real world. Therefore, we apply the policy learned in simulation on a real robot. Each policy is evaluated in 10 test runs of navigating from start to destination. The resulting times are shown in Fig. 5.

Similar to the results from simulations, our learned approach outperforms any policy of constant velocity by more than 35% and is significantly better. When looking at the trajectories generated by the policies qualitatively, the results are also similar to the simulated ones (Fig. 6). 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 however, it 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 it is in risk of getting lost, the robot immediately slows down to re-localize. As a result, it reaches the destination reliably and fast.

To summarize, the policy learned in simulations could be successfully applied on a real robot and performs significantly better than the naive approach.

D. Policy Compression

The policy we learned so far consisted of a table with 120 entries. The resulting clustering found via X-means is shown in Figure 7 as projection into the two dimensions of the state features distance and entropy. In total, four clusters are found (0.26 m/s, 0.39 m/s, 2×1 m/s). These four cluster means lead to a representation which is an order of magnitude compacter than the initial table.

To decide on which velocity to set, the robot now uses the previously determined cluster means instead of the table-based action selection. According to the current state (d, φ, h), the velocity assigned to the closest cluster mean is selected. Using 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 10.29 s $\pm$ 1.42 s with no significant difference between using the two representations. This shows that we were successfully able to compress the learned policy to a significantly smaller representation with no loss of performance in the task.

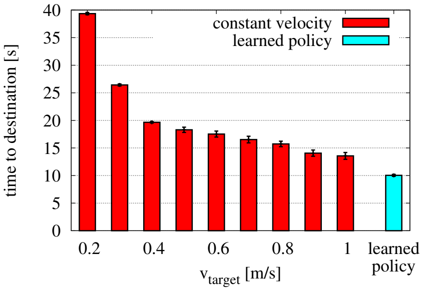

Fig. 4. Comparison of a learned policy to following a constant velocity policy in simulation. The learned policy is significantly better than all policies with a constant velocity.

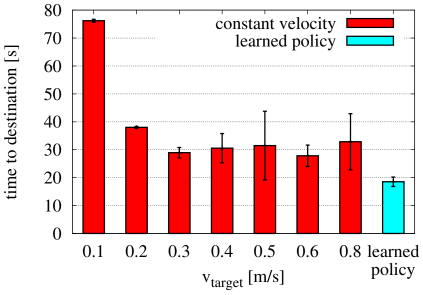

Fig. 5. Comparison of constant velocity policies to a learned policy on the real robot, each averaged over 10 runs. The learned policy is significantly better than all policies with a constant velocity.

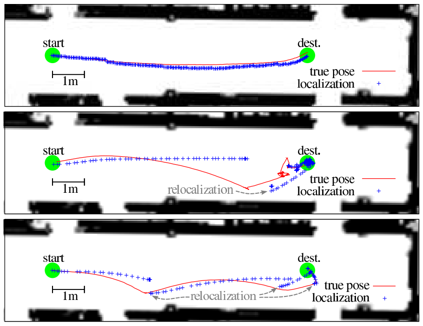

Fig. 6. Comparison of trajectories in reality at constant velocity (top: 0.2 m/s, middle: 0.8 m/s) and variable velocity, i.e., following the learned policy (bottom).

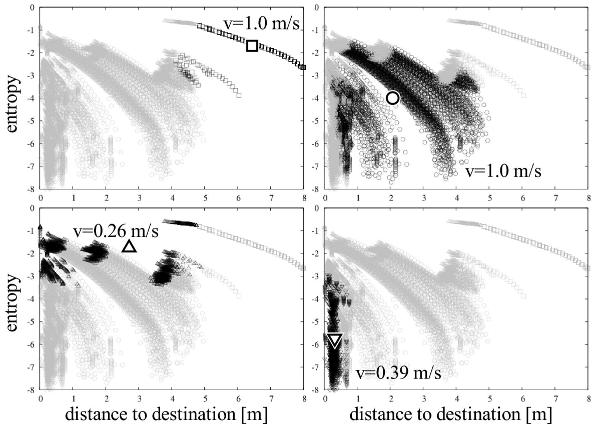

Fig. 7. X-means clustering of the learned policy. Observed values of distance, entropy, and angle are used for clustering, while only the projection on distance and entropy is shown. Four clusters with three different velocity classifications are found, highlighted in the single plots.

VIII. Conclusion

We presented a new approach to learning an efficient navigation policy using visual features for localization. By considering this as a reinforcement learning task, the robot learns a policy for choosing the optimal velocity such that it reaches its target location as fast as possible and with minimum error. During learning, the inherent impact of motion blur on the feature detections coming from the movements of the robot is implicitly taken into account. In simulated and real-world experiments, we demonstrate that our learned navigation policy significantly outperforms any strategy that applies a constant velocity in terms of time to reach the destination. Furthermore, we were able to show the general applicability of the learned policy to different environments in terms of feature density, which simplifies learning.

Additionally, we introduced a technique for compressing the learned policy by employing X-means clustering on the visited state space when following the learned policy. In our experiments, the compressed policy yields a similar performance as the original one. This is especially valuable for memory-constrained systems such as lightweight UAVs and humanoid robots, to which we plan to apply our approach in the future.

References

[1] 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, to appear.

[2] N. Roy and S. Thrun, “Coastal navigation with mobile robots,” in *Advances in Neural Processing Systems 12 (NIPS)*, vol. 12, 1999.

[3] M. Wiering and J. Schmidhuber, “Fast online Q(λ),” *Machine Learning*, vol. 33, no. 1, pp. 105–115, October 1998.

[4] D. Pelleg and A. Moore, “X-means: Extending K-means with efficient estimation of the number of clusters,” in *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[5] 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 International Conference on Robotics and Automation (ICRA)*, 2008.

[6] T. Kollar and N. Roy, “Using reinforcement learning to improve exploration trajectories for error minimization,” in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[7] N. Roy, W. Burgard, D. Fox, and S. Thrun, “Coastal navigation–mobile robot navigation with uncertainty in dynamic environments,” in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1999.

[8] M. Bryson and S. Sukkarieh, “Active airborne localisation and exploration in unknown environments using inertial SLAM,” *IEEE Aerospace Conference*, 2006.

[9] H. Strasdat, C. Stachniss, and W. Burgard, “Which landmark is useful? Learning selection policies for navigation in unknown environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, to appear.

[10] C. Kwok and D. Fox, “Reinforcement learning for sensing strategies,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, vol. 4, 28 Sept.–2 Oct. 2004, pp. 3158–3163.

[11] 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*, ser. STAR Springer tracts in advanced robotics, H. Christiensen, Ed., vol. 22, 2006.

[12] 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*, vol. 28, no. 2, pp. 315–325, 2009.

[13] 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*, 1997, pp. 182–193.

[14] G. A. Rummery and M. Niranjan, “On-line Q-learning using connectionist systems,” Cambridge University, Cambridge, UK, Tech. Rep. CUED/F-INFENG/TR 166, September 1994.

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

[16] K. Doya, “Reinforcement learning in continuous time and space,” *Neural Computation*, vol. 12, no. 1, pp. 219–245, 2000.

[17] H. Bay, T. Tuytelaars, and L. V. Gool, “SURF: Speeded-up robust features,” *Proceedings of the ninth European Conference on Computer Vision*, vol. 110, no. 3, pp. 346–359, 2006.