Active Neural Localization in Noisy Environments

Cong Li[1] Geng Yan[1] Wentao Yuan[1]

Abstract— **Localization, the problem of estimating the location of the robot given a map and a sequence of observations, is a fundamental problems in mobile robotics. Most traditional localization methods are passive, i.e. the robot does not have the ability to adjust its action based on its observations. The recently proposed active neural localization algorighm combines deep neural networks with Bayes filter to perform efficient active localization. In this project, we seek to extend active neural localization in noisy environments, where Gaussian noises are added to both the position and the observation of the agent. A series of experiments show that our active localization method outperforms passive localization methods in both noiseless and noisy environments.**

I. Introduction

Localization, which is the problem of estimating the location of the robot from an observation and a map of environment, has received the greatest attention in the past decade. Traditional methodologies, such as Kalman filter [1], updates the agent's location and the map based on a passive stream of observations, which, Chaplot et al.[2] argue, is sub-optimal in the number of steps required to localize the robot and recover the map, as the actions taken by the agent are rather random and arbitrary. To solve this problem, Chaplot et al. proposed a fully differentiable neural network that grants the agent the ability to optimally localize itself given the environment map. Different from traditional localization methods, the model is based on the Bayesian filter containing the perceptual model and a policy model, has the ability to predict the action taken in the future. In this way, the robot can learn to navigate quickly to unambiguous locations. However, the problem that they solved, is based on one strong prior that the perception and motion models are noise-free, which is not suitable for real-life situations. In this project, we seek to extend their work to Active Neural Localization in noisy environments, with the agent capable of adjust the perception based on previous conditions.

The contribution of our project is to extend the scope of active neural localization, introducing noise into the perceptual model and applying technique such as probabilistic graphical model to address the noise and increase the robustness of the proposed model.

II. Related Work

Localization has been an active field of research since more than two decades. There are many efficient passive localization methods. Smith et al.[3] proposed the stochastic map which always contains the best estimates of relationships among objects in the map, and their uncertainties. The Kalman filter is used for localization. Nister et al. [4] used geometry-based vision methods for estimating the motion of the robot. Recently, some learning-based localization methods such as DeepVO [5] and VINet [6] are proposed showing the effectiveness of deep learning for localization problems. However, these localization methods are passive because they don't have the ability to predict the action of the robot taken in the future. The ability to predict the action could eliminate the ambiguity to localize more quickly and more accurately.

Recent work has also made progress by incorporating reinforcement learning based policy model with traditional localization methods. Zhang et al. [7] presented an approach to provide deep reinforcement learning agents with long-term memory to learn representations of a global map from sensor data. They formulated the exploration task as a Markov decision process in which the agent interacts with the environment through a sequence of observations, actions and rewards. They embed procedures including localization prediction, motion prediction, data association, measurement mapping and mapping, mimicking that of traditional SLAM into the soft-attention based addressing to bias the write/read operations towards SLAM-like behaviors. Chaplot et al.[2] proposed "Active Neural Localizer", which uses structured components for Bayes filter-like belief propagation and learns a policy based on the belief to localize accurately while minimizing the number of steps for localization. They use grid-based representation of belief, which is popular among localization methods. The proposed model has two main components: the perceptual model and the policy model. For the perceptual model, it firstly outputs the feature representation from the agent's observation and the states given in the map information. A trainable deep convolutional network is used to get the feature representation for 3D environment. Then, Cosine similarity is utilized to update the likelihood of each state in the map information. The policy model, which is trained using Asynchronous Advantage Actor-Critic (A3C), gives the probability of the next action given the current belief of the agent. The experimental results show the proposed model outperforms the baseline models while being order of magnitudes faster for various 2D and 3D environments including a realistic map in the unreal environment. However, their implementation is based on one strong prior that the perception and motion models are noise-free, which is not suitable for real-life situations. In this project, we extend their work for noisy environments.

[1]`{congl1, gyan, wyuan1}@andrew.cmu.edu`

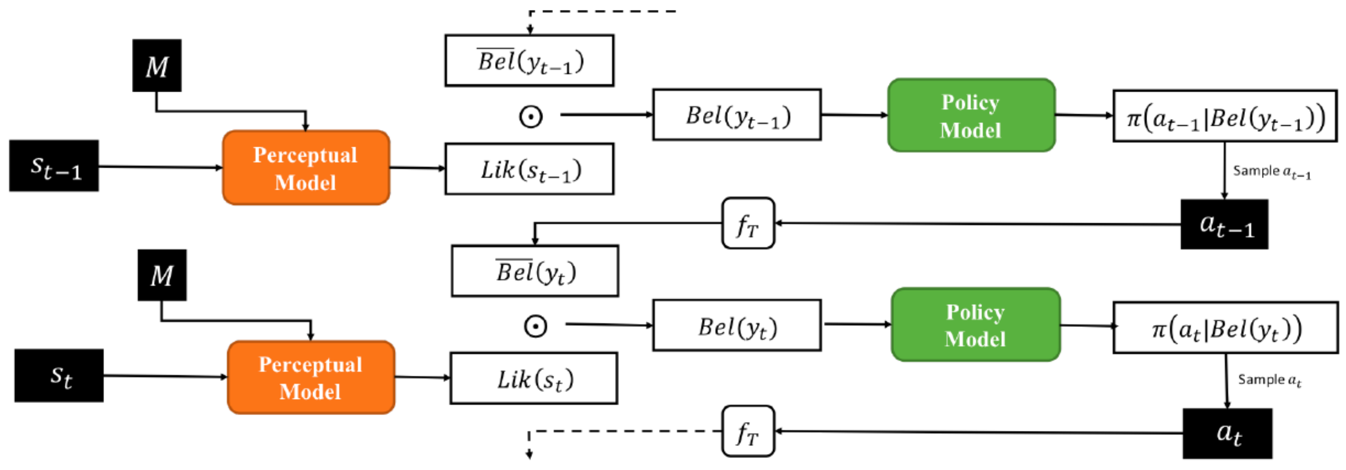


Fig. 1. The Active Neural Localization framework [2]

III. Methodolody

A. Active Neural Localization

We build our model on top of the model proposed by [2]. The model is based on Markov Localization [8], a probabilistic approach using Bayesian filtering whose belief can be recursively computed using following equations:

$$\overline{Bel}\left(y_t\right) = \sum_{y_{t-1}} p\left(y_t|y_{t-1}, a_{t-1}\right) Bel(y_{t-1}) \qquad (1)$$

$$Bel(y_t) = \frac{1}{Z} lik\left(s_t\right) \overline{Bel}\left(y_t\right) \qquad (2)$$

where y_t is random variable denoting location of the agent at time t, agent's observation at time t is represented as s_t, a_t denotes the action taken by the agent at time t, $\overline{Bel}\left(y_t\right)$ and $Bel(y_t)$ are the belief of the location of the robot at time t before and after current observation, respectively, $Lik(s_t)$ is the probability of observing s_t and Z is the normalization constant. The model has two novel components: the perceptual model and the policy model. The overall architecture is shown in Figure 1. The goals for active localization are both estimating $Bel(y_t)$ to obtain the probability distribution of $y_t$and learning the policy model.

For the 2D environment, the map size is denoted as $M \times N$, belief is expressed as the 3-dimensional matrix as an $O \times M \times N$ tensor denoting the probability of being at a particular position, O is the number of possible orientations of the robot which is defined as 4 (North, East, South, West) in this project. Furthermore, we also estimate the likelihood map and $Lik(s_t) = p(s_t|y_t)$ is the probability of observing s_t conditioned the location of the robot.

In perceptual model, the feature representation from the agent's observation and the states given in the map information are computed. In the 2D environment, we compute a one-hot vector of the same dimension as the observation to represent the depth which is used as the feature representation directly. In order to improve the computational efficiency, we pre-calculated the feature representation from the agent's observation for each location of the 2D map. The perceptual model takes the observation s_t, the map M and belief of the agent's location at time t before observing s_t as input and produces the likelihood of the observation $Lik(s_t)$, which is then used to update the agent's belief as following:

$$Bel(y_t) = \frac{1}{Z} Lik\left(s_t\right) \odot \overline{Bel}\left(y_t\right) \qquad (3)$$

The policy model is a neural network which takes the agent's current belief of its pose $Bel(y_t)$ and the map design M as input and produces a distribution of possible actions to take in the next step in order to localize as fast as possible. The model starts with two convolutional layers, each with 16 filters of size $3\times$ and stride of 1. Then, the feature vector is concatenated with the embeddings of the previous 5 actions and the current step and feed to two streams of fully-connected layers – one predicts the probability of each action and the other predicts the value of the current state. For more information about this two-stream architecture, please refer to [9]. The embeddings are computed by mapping the one-hot representation of the action/step into a 8-dimensional space via a learned linear transformation. We find this is crucial to avoid the situation where the agent is stuck in keeping turning around at the same location.

B. Bayes Filter for Noisy Environments

The Bayes filter model in [2] did not take any noise in the odometry and the observation into account. In order to deal with the noisy environments that we designed, we introduces a Bayes filter model that considers both the odometry and observation noise.

It is challenging to introduce noise into the model proposed by [2], because it is designed for a discrete environment, while most of existing filtering algorithms assume a continuous environment. For simplicity, we decide to use a discrete approximation of the Gaussian noise.

For the odometry noise, we assume a 1D Gaussian with mean and sigma 0.4 as the noise distribution. In other words, after receiving the command of moving forward 1 unit, the agent will move 1 unit (noise= 0) with possibility of 0.7884, move 2 units (noise= 1) with possibility of 0.1058, or stay

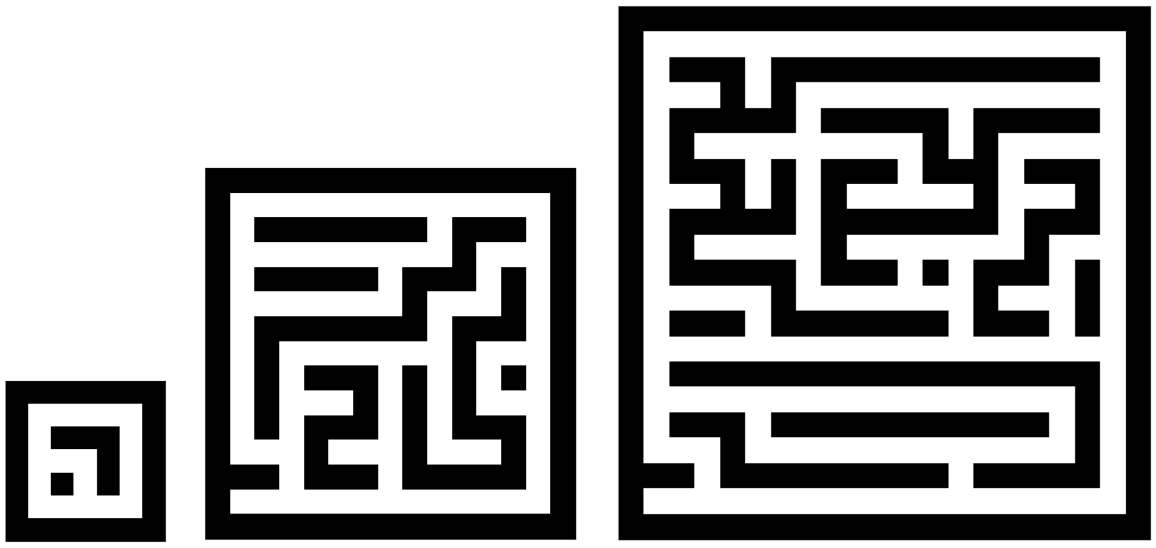

(a) 7×7 (b) 15×15 (c) 21×21

Fig. 2. 2D Maze Environment

still (noise$= -1$) with possibility of 0.1058. We modify the belief propagation to incorporate this noise. Previously, the propagation of belief from $Bel(y_{t-1})$ to $\overline{Bel}(y_t)$ is determinant. Now the agent can move to 3 different possible locations, so we calculate the updated belief map using all three possibilities and use the weighted average as the final belief map $\overline{Bel}(y_t)$ for the next step. Note that we only consider noise in forward motion. This is because incorporating rotational noise requires a totally different design for the belief maps. We decide to leave that for future work.

For the observation noise, we assume the same noise distribution of a 1D Gaussian with mean 0 and sigma 0.4. This means that if the actual observation is s_t at step t, the observation that the agent receives can be $\tilde{s}_t = s_t$ with probability of 0.7882, $\tilde{s}_t = s_t + 1$ with probability of 0.1058, or $\tilde{s}_t = s_t - 1$ with probability of 0.1058. To address this noise, we calculate the separate likelihood maps for all three possible observations based on the input observation s_t, then use the weighted average with possibilities as weights as the final likelihood map.

IV. Experiments

A. Environment

We trained and tested our models on a 2D simulation environment of random mazes, similar to the one in [2]. In this environment, the map is a 2D maze represented as a $M \times N$ occupancy grid, as shown in Figure 2. The maze design is generated by a random depth first search algorithm that creates n islands of walls of length p in an enclosed area while keeping the free space connected. The parameters n and p controls the density and complexity of the maze, respectively. For more details about maze generation algorithms, please refer to [10]. We generated random mazes of three different sizes to test our algorithm (see Figure 2 for example maze designs).

The localization problem is represented as a Markov Decision Process $\mathcal{M} = (S, A, R, T, \gamma)$, where S, A, R, T and γ are the state space, action space, reward function, transition function and discount factor. Here, the agent's state $s \in S$ consists of its location and orientation. The location is discretized into positions on the $M \times N$ grid and the orientation is discretized into 4 directions – North, East, South and West, but we will relax this assumption later. At the beginning of each episode, the agent is placed in a random free grid with random orientation. At each step, the agent takes an action a and receives a reward $r = R(s, a)$. Its state is then changed based on the transition function to $s' = T(s, a)$. The episode ends when a maximum number of steps is reached.

In the environment, three actions are available to the agent: moving forward, turning left 90 degrees and turning right 90 degrees. The transition function changes the agent's position and orientation accordingly. If the agent's action is moving forward but a wall is in front, it stays at the last location. In our case, the state is not directly visible to the agent (otherwise there will be no point doing localization). Instead, the agent receives some observation o based on its state s. In the 2D maze, o is simply the number of free space in front of the agent. We can think of the agent as having a single-beam range sensor.

We designed three versions of the environment which increasingly approximate the real world scenario. In the Maze-v1 environment, the transition function is deterministic and there is no noise in the observation. This means the agent has perfect motion and sensor model, which allows us to focus on testing the correctness of the policy model. In the Maze-v2 environment, the agent's motion and observation are corrupted by a noise σ whose distribution is a

Env		Maze-v1					
Map Size		7x7		15x15		21x21	
Episode Length		15	30	20	40	30	60
Passive Localization	Time	2.89	7.55	26.55	26.50	62.85	71.07
	Acc	0.239	0.357	0.401	0.609	0.490	0.717
Active Neural Localization	Time	14.98	26.24	41.79	59.73	88.40	118.78
	Acc	**0.699**	**0.817**	**0.949**	**0.980**	**0.729**	**0.987**

TABLE I

RESULTS ON DETERMINISTIC ENVIRONMENTS

Env		Maze-v2					
Map Size		7x7		15x15		21x21	
Episode Length		15	30	20	40	30	60
Passive Localization	Time	6.97	12.21	48.42	79.65	158.66	258.11
	Acc	0.212	0.326	0.376	0.549	0.443	0.641
Active Neural Localization	Time	29.50	54.83	131.72	143.69	240.21	438.43
	Acc	**0.512**	**0.624**	**0.695**	**0.842**	**0.546**	**0.847**

TABLE II

RESULTS ON DISCRETE STOCHASTIC MOVEMENTS

Env		Maze-v3					
Map Size		7x7		15x15		21x21	
Episode Length		15	30	20	40	30	60
Passive Localization	Time	6.35	11.09	50.92	78.19	137.99	221.89
	Acc	0.154	0.209	0.311	0.446	**0.409**	0.576
Active Neural Localization	Time	17.21	33.77	74.19	118.99	158.14	346.04
	Acc	**0.308**	**0.401**	**0.347**	**0.578**	0.396	**0.587**

TABLE III

RESULTS ON CONTINUOUS STOCHASTIC MOVEMENTS

discrete approximation to a Gaussian. Specifically, we have $p(\epsilon = -1) = 0.10558$, $p(\epsilon = 0) = 0.7884$ and $p(\epsilon = 1) = 0.10558$. In the Maze-v3 environment, the agent's location is no longer discrete but a tuple of real numbers (x, y). When the agent moves forward, we add a continuous Gaussian noise $\sigma \sim N(0, 0.4)$. When the observation is calculated, we discretize the agent's location again in order to perform ray casting. No rotational noise is added because that involves significant changes to the ray casting procedure and likelihood calculation.

For Maze-v2 and Maze-v3 environments, the belief propagation is changed according to the noise. Specifically, when the agent moves forward, instead of shifting the previous belief directly, we linearly interpolate the previous belief and the shifted belief so that the motion noise is taken into account. The main point here is to test the robustness of the policy model to motion noise, so we did not use a complicated filtering scheme. However, this sometimes leads to the situation where the agent's belief becomes all 0, especially in Maze-v3 where the drift accumulates. In that case, we reinitialize the belief to be uniform over the map so that the agent can relocalize based on new observations.

For all the environments, the agent receives a reward of 1 if it successfully localizes and 0 otherwise. A successful localization means that at the end of episode, the probability of the agent's actual pose in its belief matrix is greater than 0.5. In other words, the pose that the agent is most certain about needs to be its actual pose and its probability needs to be greater than 0.5. To speed up training, we also use the negative entropy of the agent's belief matrix as an intermediate reward at each step.

B. Results

We compared the performance of a passive localization algorithm with our implementation of active neural localization. In the passive localization algorithm, only the policy model is replaced by random action and everything else including the Bayes filter remains the same. The results are shown in Table I, II and III.

The policy model in the active neural localization algorithm is trained using advantage actor-critic (A2C) [11] for 200,000 episodes. During training, every 500 episodes we evaluate the model's localization accuracy on 100 mazes. The training curves are shown in Figure 5.

We ran both agent for 1000 episodes on mazes of three different sizes 7×7, 15×15 and 21×21. The random seed used to generate the test mazes are fixed so that the agent is always tested on the same set of mazes. The accuracy (Acc) entry in the tables measures the percentage of successful localization (see definition in the previous subsection) out of 1000 test episodes. Since each episode is of a fixed length, the accuracy is closely correlated with the agent's ability to efficiently localize itself within a limited number of steps. The time entry in the tables measures the time it takes to run the algorithm for 1000 episodes. Note that this does not reflect the efficiency of localization, because in the simulation environment the agent's motion is instant. This entry is there to illustrate the computation overhead that the policy model imposes. In practice, the localization time mainly depends on the number of steps the agent takes to correctly localize.

We note that our maze environment is very challenging for localization, since the agent's observation is limited to only one direction and there are lots of similar structures. Nevertheless, from the tables we can see that our implementation of the active neural localization is able to achieve fairly high localization accuracy within a small number of steps and outperform the passive localization in nearly all cases. This shows that the ability to take conscious actions significantly improves localization efficiency. An example that further illustrates this is shown in Figure 3 and 4. In this example, we can see that the policy that the active agent learned tells it to avoid behaviors that will not help the localization, such as bumping into walls, so it turns left at step 1 (Figure 3(a), 3(b)) while the passive agent moves straight into a wall (Figure 4(a), 4(b)). Moreover, the active agent learns behaviors that can help it localize faster, e.g. turning at intersections. At step 8 when it encounters an intersection, it turns left and immediately localizes it self to one possible pose (Figure 3(c), 3(d)). A passive agent, however, goes straight when it encounters the same intersection and is not able to reduce the ambiguity in its belief (Figure 4(c), 4(d)).

Another notable thing for the noisy environments (Maze-v2 and Maze-v3) is that the policy model is not retrained. Only the Bayes filtering is modified to incorporate noise in the motion and sensor model. This shows the policy model's robustness to noise and ability to generalize. It means that we can train the policy model in simulation environments which may have very different noise characteristics from real environments, but an agent can still use it to improve localization efficiency in real environments.

V. Conclusion and Discussion

In this project, we implemented the active neural localization framework. Moreover, we designed more realistic simulation environments with motion and observation noise and extended the original model to incorporate these noises. We first built a simple noisy environment by corrupting the agent's motion and observation by a discrete approximation to a Gaussian noise. Then, we built another environment using a continuous pose representation which allows us to add a continuous Gaussian noise for the motion model. A series of comparative experiments show that our our implementation of active neural localization is more efficient to accurately localize than a passive localization algorithm. This shows the advantages of combining localization and planning in a single end-to-end framework. In the future, this work can be extended to 3D simulation and real environment and even a full SLAM system by incorporating graph optimization and external memory.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[2] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," *arXiv preprint arXiv:1801.08214*, 2018.

[3] R. Smith, S. Matthew, and C. Peter, "Estimating uncertain spatial relationships in robotics," *Autonomous robot vehicles*, 1990.

[4] D. Nistr, N. Oleg, and B. James, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, 2006.

[5] S. Wang, C. Ronald, W. Hongkai, and N. Trigon, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017.

[6] R. Clark, W. Sen, W. Hongkai, M. Andrew, and T. Niki, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," *AAAI*, 2017.

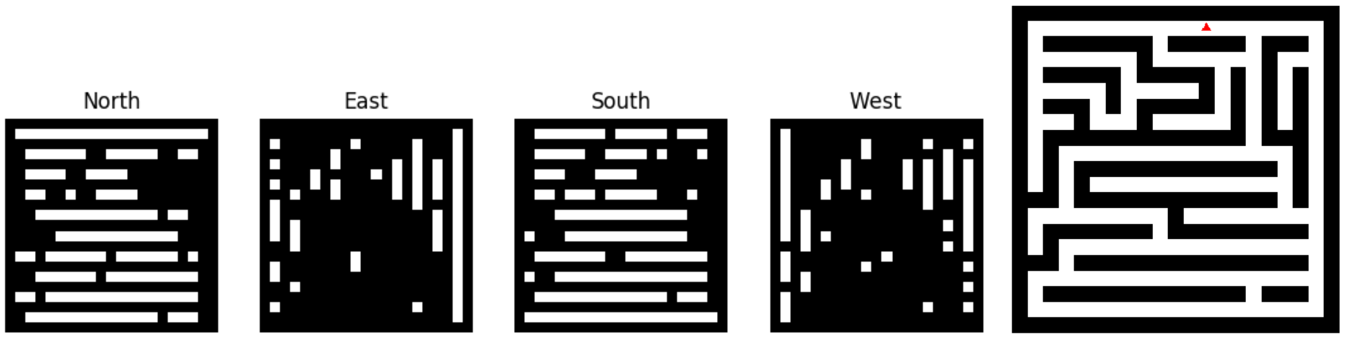
[7] Z. Jingwei, T. Lei, B. Joschka, B. Wolfram, and L. Ming, "Neural slam: Learning to explore with external memory," *arXiv preprint arXiv:1706.09520*, 2017.

[8] F. Dieter, "Markov localization-a probabilistic framework for mobile robot localization and navigation," *phD thesis, Universitat Bonn*, 1998.

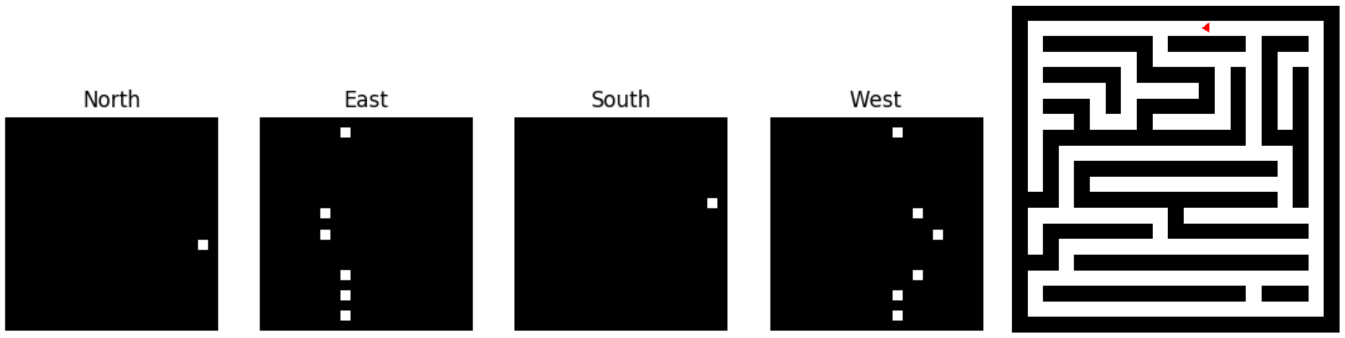
[9] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[10] Wikipedia contributors, "Maze generation algorithm — Wikipedia, the free encyclopedia," 2018. [Online; accessed 4-May-2018].

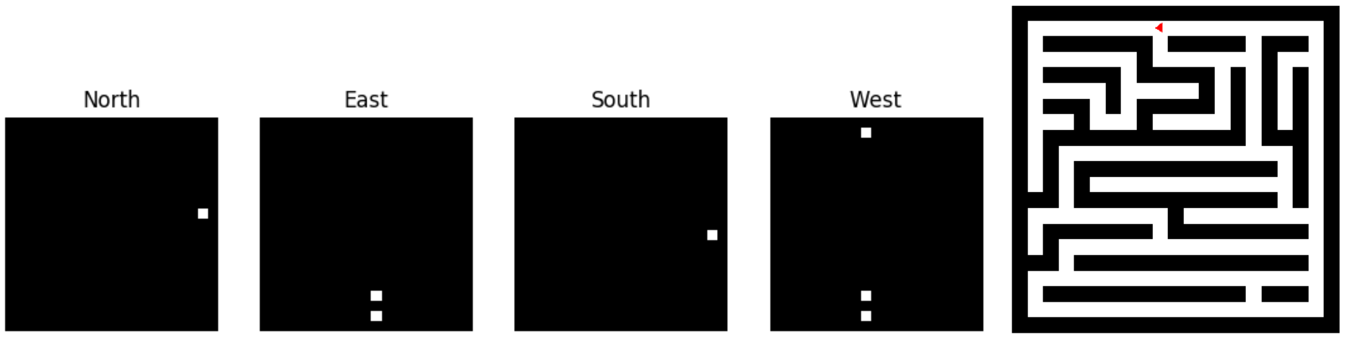
[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.



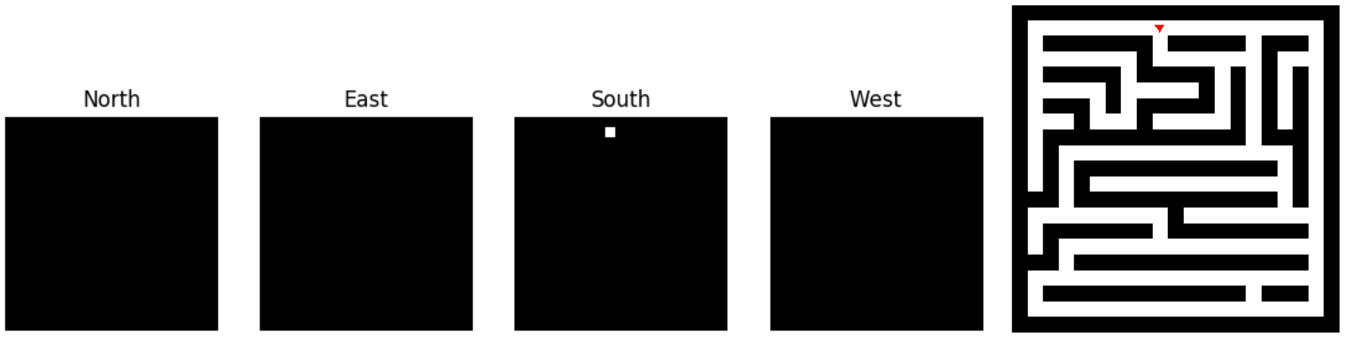
(a) Step 0



(b) Step 1

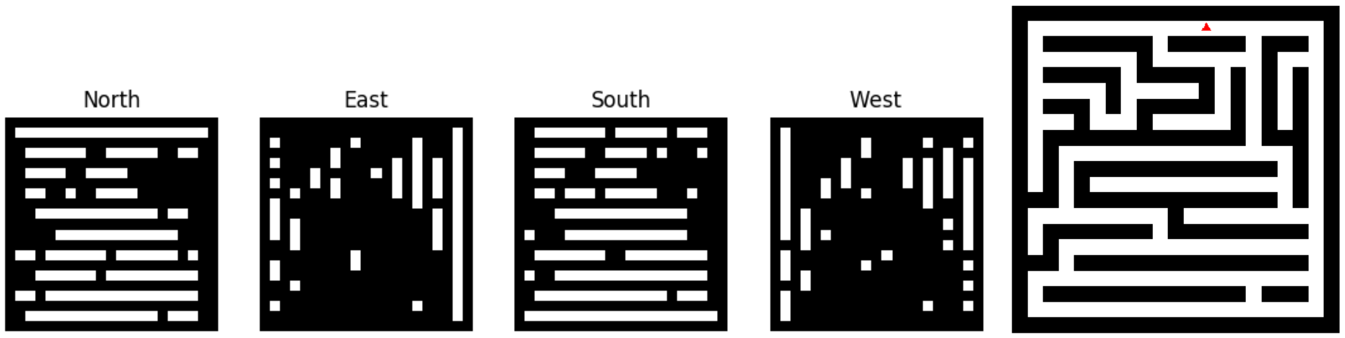


(c) Step 8

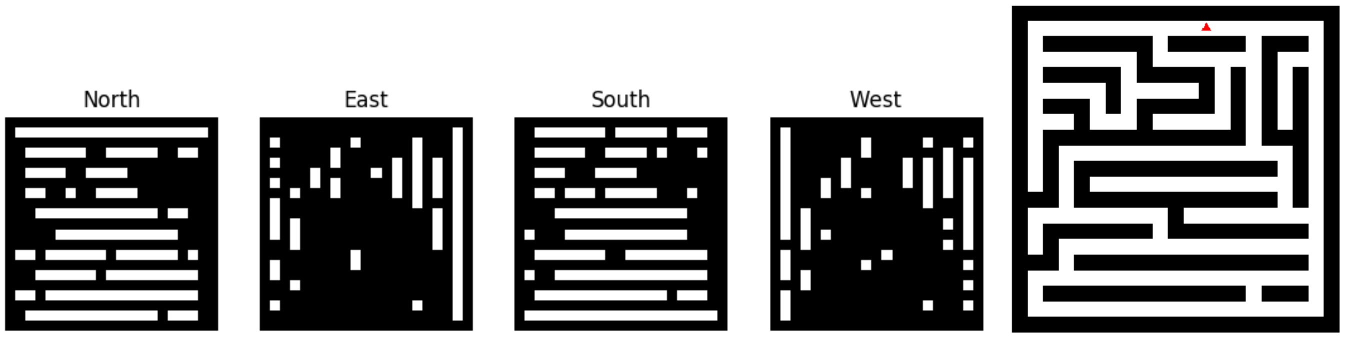


(d) Step 9

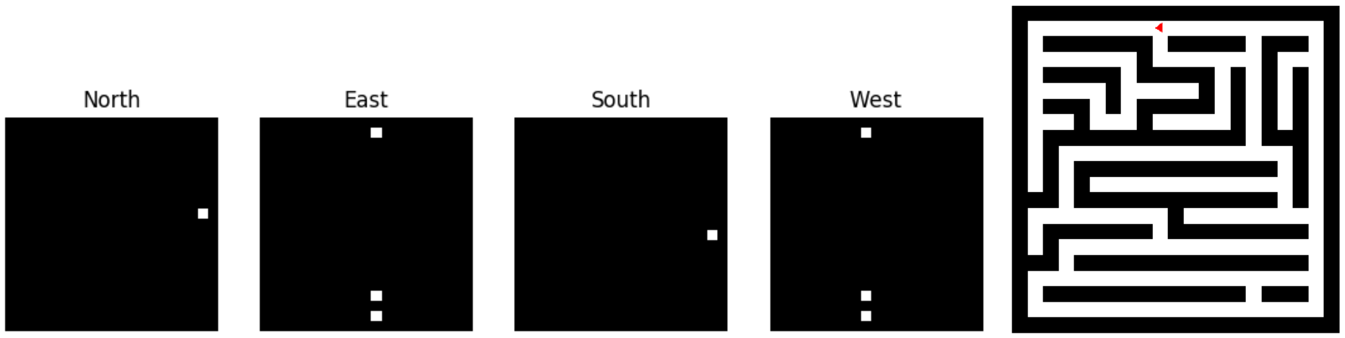
Fig. 3. An active agent localizing itself, with the agent's belief matrix on the left and the agent's ground truth pose on the right.



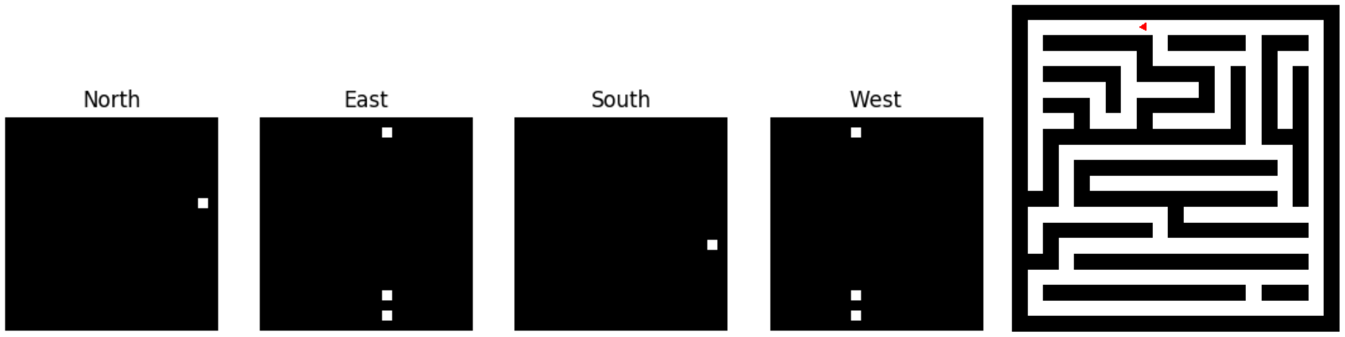
(a) Step 0



(b) Step 1

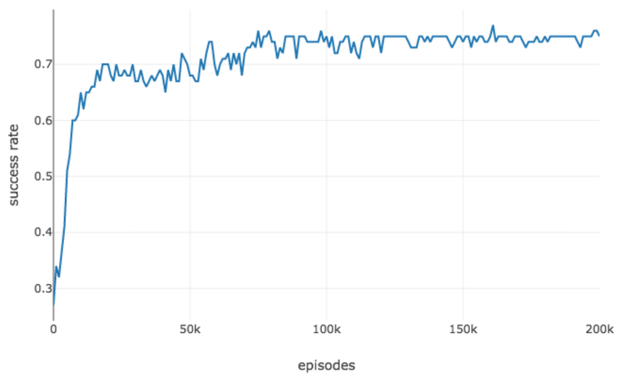


(c) Step 10

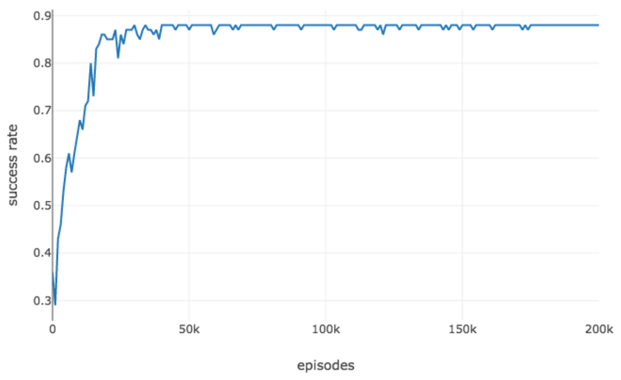


(d) Step 11

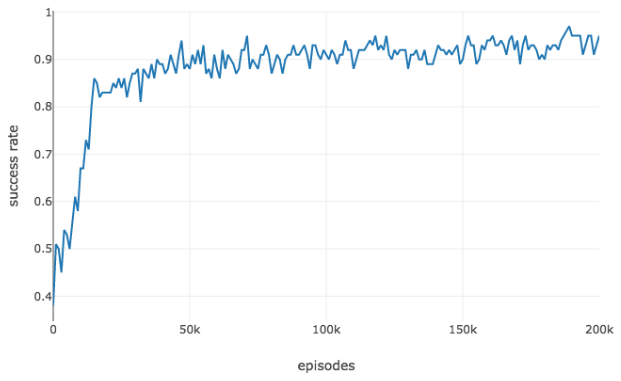
Fig. 4. A passive agent localizing itself, with the agent's belief matrix on the left and the agent's ground truth pose on the right.



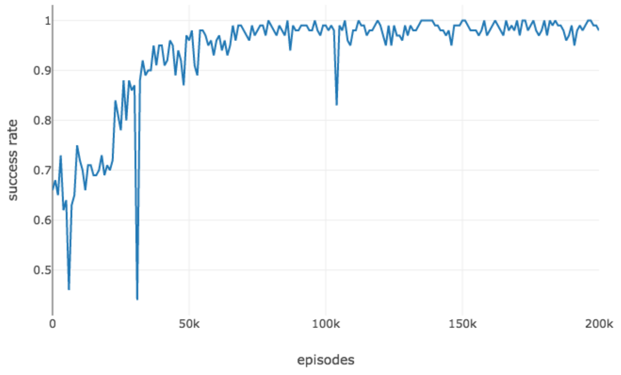
(a) Size 7, Max step 15



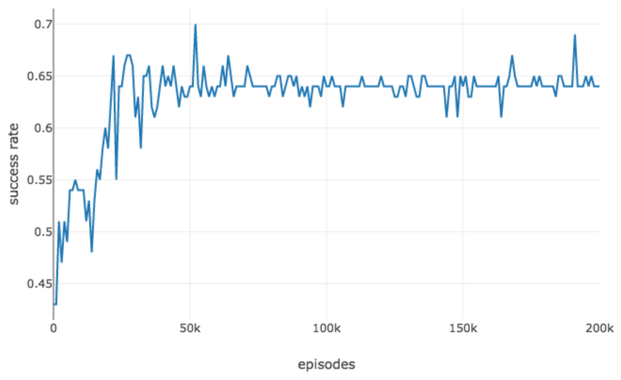
(b) Size 7, Max step 30



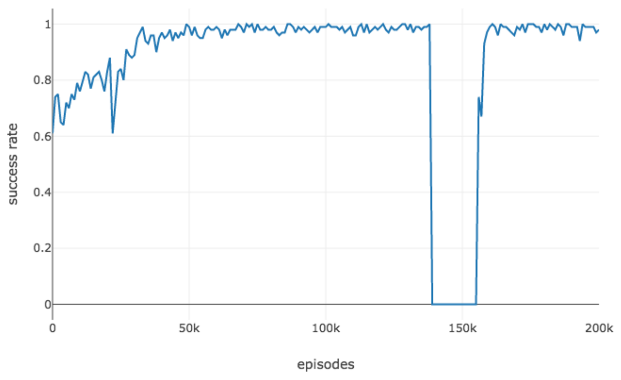
(c) Size 15, Max step 20



(d) Size 15, Max step 40



(e) Size 21, Max step 30



(f) Size 21, Max step 60

Fig. 5. Localization accuracy during training