Reinforcement Learning-based Mapless Navigation with Fail-safe Localisation*

Feiqiang Lin[1][0000−0001−5528−0001], Ze Ji[1][0000−0002−8968−9902], Changyun Wei[2][0000−0002−5788−6573], and Hanlin Niu[3][0000−0003−0457−0871]

[1] School of Engineering, Cardiff University, Cardiff, UK
{linf6, jiz1}@cardiff.ac.uk
[2] College of Mechanical and Electrical Engineering, Hohai University, China
weichangyun@hotmail.com
[3] Department of Electrical & Electronic Engineering, University of Manchester, UK
hanlin.niu@manchester.ac.uk

Abstract. Mapless navigation is the capability of a robot to navigate without knowing the map. Previous works assume the availability of accurate self-localisation, which is, however, usually unrealistic. In our work, we deploy simultaneous localisation and mapping (SLAM)-based self-localisation for mapless navigation. SLAM performance is prone to the quality of perceived features of the surroundings. This work presents a Reinforcement Learning (RL)-based mapless navigation algorithm, aiming to improve the robustness of robot localisation by encouraging the robot to learn to be aware of the quality of its surrounding features and avoid feature-poor environment, where localisation is less reliable. Particle filter (PF) is deployed for pose estimation in our work, although, in principle, any localisation algorithm should work with this framework. The aim of the work is two-fold: to train a robot to learn 1) to avoid collisions and also 2) to identify paths that optimise PF-based localisation, such that the robot will be unlikely to fail to localise itself, hence fail-safe SLAM. A simulation environment is tested in this work with different maps and randomised training conditions. The trained policy has demonstrated superior performance compared with standard mapless navigation without this optimised policy.

Keywords: Fail-safe Localisation Navigation · Mapless Navigation · Reinforcement Learning

1 Introduction

For robots to navigate in unknown environments without knowing the maps, such as in search and rescue scenarios, reliable decision making for the robot of immediate responses to collisions or efficient path planning towards the goal is critical. We categorise such problems as mapless navigation. Conventional path

* The authors thank the China Scholarship Council (CSC) for financially supporting Feiqiang Lin in his PhD programme (201906020170). (*Corresponding author: Ze Ji*)

planning methods have been dominantly applied in most occasions. However, there are known limitations with these algorithms [13]. Usually, hand-crafted heuristic or constraint functions are needed and customised for different conditions. However, too much hand-engineered path planning could limit the generalization capability of mobile robots to be employed in different environments [14].

To address the limitations above, with recent advances in deep learning and reinforcement learning (RL), learning based navigation approaches have continuously attracted increasing attention. Supervised learning that learns from expert demonstrations is one popular approach, which, however, would require a large amount of labelled data for training. An alternative approach is RL that deploys an agent in the environment and lets the agent explore by itself through direct interaction with the environment. By gaining corresponding rewards from environment during exploration, the agent will learn how to navigate gradually. One promising recent work is the RL-based mapless navigation [13] that aims to train an agent, a mobile robot, to navigate in an unknown environment with the capabilities of collision avoidance. This could reduce a considerable amount of time to tailor hand-crafted rules or heuristics for navigation and decision making.

Despite the promising performance from previous works, they all assume that the robots can access their actual poses. However, this assumption is unrealistic, especially for GPS-denied environment. Also, even with GPS localisation, localisation quality along the navigation path should be taken into consideration. SLAM-base localisation will be required in such cases. However, its performance is prone to poor observation of environment features, e.g. navigation in areas with no distinct features, e.g. an open area. Most localisation algorithms, such as PF or Kalman filter, will be negatively impacted by environment ambiguities and, hence, more weights will be given to interoceptive sensors, such as odometers, leading to unreliable localisation. The decoupled nature of robot perception and path planning could lead to catastrophic failures of self-localisation, due to the unpredictable observable features from the surroundings to perform SLAM-based localisation. The unreliable localisation will then in turn result in failures of reaching its goal location. Fig. 1 illustrates a real-world scenario, where the grey path is more preferred than the less reliable path in white for drone navigation (assuming GPS localisation is unavailable). In this case, the grey path would allow the drone to observe more local features, hence improving localisation robustness.

This leads to the motivation of this work: how to train a policy for an agent to learn to navigate while also prevent localisation failures during navigation. The aim of our work is, therefore, two-fold: to train a robot that is able to 1) avoid collisions, while also 2) plan its paths that can provide robust localisation. This is different from other mapless navigation agents in previous works, which only consider obstacle avoidance without considering localisation performance.

The remainder of this paper is organised as follows. Section 2 introduces related work. Our method in this work is introduced in section 3, followed by

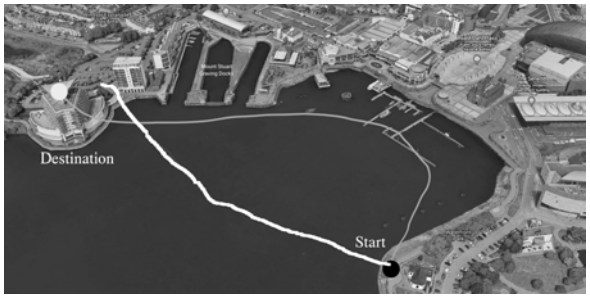


Fig. 1: A robot navigates from the start location to the destination on the left-hand side. The white trajectory is traversing in a feature-poor area, which is not suitable for SLAM-based localisation. The grey trajectory is a more preferred path, which maximises feature observations for robust SLAM-based localisation.

experiments and results in section 4. The conclusion and future research are presented in section 5.

2 Related Work

With the great advances of neural networks, deep learning has been widely utilised to teach mobile robots driving by expert demonstrations by various means. For example, supervised learning techniques, such as Convolutional Neural Networks (CNN), have been deployed to train robots to autonomously make decisions or act directly based on depth images or Lidar data, to learn to navigate [5,9,11]. However, as it would be costly to collect labelled data in the real world, those methods are often trained and evaluated in virtual environments. In recent years, efforts have been paid to focus on transfer the trained networks to work in real world too [13,2].

RL, on the other hand, is more favourable, as it allows an agent to perform autonomous exploration and learning without human intervention. For mapless navigation, one prominent work is introduced in [12], where robots are trained with RL by a two-step method with depth images as inputs. Since then, several variants of related works have been introduced, inheriting the above method to improve the performance of mapless navigation in different aspects [8]. For discretized action space, state-of-the-art Deep Q-Networks (DQN), such as double networks and duel architectures, are integrated together to enhance robot navigation abilities [10].

Further, RL is also used for navigation in continuous action space by deploying the Asynchronous Deep Deterministic Policy Gradient (DDPG) algorithm [13]. RL in continuous action space requires more data than discretized space. To improve its sample efficiency, imitation learning and RL can be combined for improving efficiency [9], where the policy network is first pre-trained with imitation learning, and, then further tuned with the constraint policy optimisation, named as the (CPO)RL algorithm. In [15], a modular architecture

is introduced to train robots on the modular basis by dividing a task into local obstacle avoidance and global navigation modules. An action scheduling mechanism is proposed to perform efficient exploration and exploitation. Other improvements have also been made in sampling efficiency [7] and algorithm hyper parameters selection [1]. When visual inputs are used for navigation, a technique called reinforcement learning with auxiliary tasks is applied in order to obtain effective representations from images for navigation tasks [6,4].

Although the works discussed above have achieved relatively promising performance, to the authors' best knowledge, none of these learning-based works have discussed the effect of localisation quality on its final navigation performance. In other words, path planning and robot perception should be considered as tightly coupled problems for decision making. Considerations should be given not only to localisation and mapping, but also optimal path planning or policy to optimise performance of localisation. The agent policy makes decisions to ensure paths are also beneficial to the localisation and mapping performance, such that uncertainty of its localisation and map construction are minimised. This is related to our work.

3 Methodology

3.1 System Description

Fig. 2 shows the system overview of this work. First, measurement data of the robot are fed to the localisation algorithm for calculating the current estimated robot pose. The estimated robot pose and the goal position are then used to compute the relative goal pose, represented by the relative distance and relative heading with respect to the robot. Finally, the relative goal pose together with measurement data are provided to the fail-safe localisation reinforcement learning agent to make decision on the next action. This procedure iterates until the robot reaches the designated goal position.

As mentioned, most mapless navigation algorithms assume the availability of ground truth poses of the robot and this assumption is highly impractical for real-world applications. On the other hand, robot pose estimation purely based on odometry is unacceptable too, due to unpredictable odometry drifts over time. In our work, it relies on sensors such as Lidar or cameras with a localisation algorithm to estimate robot poses.

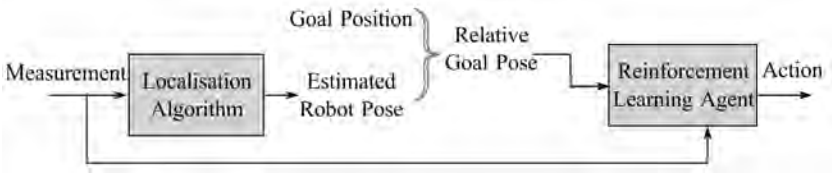


Fig. 2: System overview

3.2 Localisation Algorithm

In this paper, we consider a feature-based PF localisation algorithm, specifically, the Rao-Blackwellized Particle Filter (RBPF) [3], which is probably the most deployed method for robot state estimation. In principle, our focus should not be limited to any particular localisation algorithm. Briefly, according to the RBPF framework, the joint probability of the map m and the robot poses x can be factorised through Rao-Blackwellization, formulated as follows:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t})p(x_{1:t}|z_{1:t}, u_{1:t-1}) \tag{1}$$

where z and u represent the measurement and the control input respectively. The particle filter maintains a batch of particles, where each particle produces their own pose estimation from control inputs and measurements and then builds a map of their own according to Equation 1. An importance weight factor is assigned to each particle to evaluate the pose estimation quality of this particle, which is defined by the following equation:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t|m_{t-1}^{(i)}, x_t^i) \tag{2}$$

where the weight factor w is updated recursively and i represents the particle identification. The particle filter will do re-sampling based on the importance weight factors. It will recursively select some particles to replace some others. The larger the importance weight factor is, the higher possibility it is of to be selected to replace other particles. After a few iterations, the particles will then converge towards the true pose gradually.

3.3 Reinforcement Learning Agent

An RL agent gains experience from interaction with the environment. At each time step t, it selects an action a from a θ parameterised policy $\pi(a|s;\theta)$ based on its current state s and executes the selected action in the environment. After execution, the state will be updated and the agent will receive a reward r. This process will iterate continuously until a termination condition is met, such as goal state achieved or exceeding the maximum time. The aim of training is to generate a policy, which maximises the accumulated discounted reward, formulated as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where γ is the discount factor.

As the objective of this work is to illustrate the necessity of considering localisation quality for planning, thus the specific RL algorithm should not affect the final conclusion. Considering DQN is relatively easy to implement and widely deployed, we use DQN in our work for this pilot study.

In DQN, a deep neural network is trained to estimate the action value $Q_\pi(s,a) = E[R_t|s_t = s, a_t]$, which is the expected return for selecting action a at state s following the policy π. The details of the DQN configuration in this work are as follows. The state space of the DQN agent consists of sensor observation measurement o_t and relative goal position g_t, which includes the relative distance d_g and heading β with respective to the robot. As the classic

DQN is designed for handling discrete action space, the action space needs to be discretized. During each time step, the agent selects a linear velocity v_{linear} among a set of values $[v_{l_1}, v_{l_2}...v_{l_i}]$ and an angular velocity among a set of values $[w_1, w_2...w_j]$. i and j can be decided according to different requirements.

In the task, the agent needs to navigate to a designated goal position, while, meanwhile, also avoids obstacles and minimises its localisation uncertainty. Therefore, in this work, the reward function is defined as follows:

$$r = \begin{cases} r_{lost} & \text{if no enough features are observed} \\ r_{collision} & \text{if collision happens} \\ r_{goal} & \text{if } d_g < d_{gmin} \\ f \times (d_{t-1} - d_t) & \text{otherwise} \end{cases} \tag{3}$$

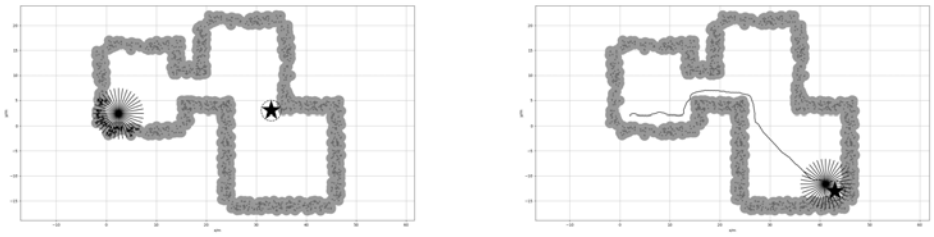
where r_{lost} is negative when the agent observation o_t does not contain enough environmental features for robust localisation; $r_{collision}$ is a negative value to punish the agent when it collides with obstacles; r_{goal} is a positive value and is set when the robot arrives at the goal position within a minimum acceptable distance, defined by d_{gmin}; the term $d_{t-1} - d_t$ is to encourage the agent to make decisions that reduce the relative goal distance; and f is the distance rate factor that can be adjusted.

Previous research works seldom consider the penalty of r_{lost} to regulate agent behaviours. However, this reward is critical to prevent the robot from moving into open space, where no or very sparse features can be observed. According to the description in section 3.2, it is clear that when the robot moves into open space, where has no enough observed features, the second term in equation 2 will not be calculated. Hence, the weight factors of the particles will not be updated. Consequently, the PF will not be able to evaluate the quality of the particles and will not perform re-sampling to correctly estimate the robot state using these weight factors. The localisation algorithm will thus fail and depend solely on odometry, which is not accurate.

4 Experiments and Results

4.1 Experiment Setup

We test our work in a 2-dimensional simulation environment using a mobile robot of a 3-dimensional kinematic motion model. As illustrated in Fig. 3a, the grey dots serve as landmarks that may be observed by the robot for localisation. Each landmark also represents an obstacle, in the circular shape with the radius of 1 m (illustrated by the light grey regions in Fig. 3a). The observation of the robot contains relative distances and angles of those landmarks to the robot within the robot maximum observation range, which is 5.0 m with a full 2π coverage. The robot needs to travel to a goal position, denoted by a black star, as shown in Fig. 3a. Those black crosses are the estimated landmarks that are observed during navigation.



(a) (b)

Fig. 3: (a) Environment: robot (black dot with laser scan beams) and goal position (star), (b) Robot navigation trajectory with ground truth poses provided

During training, those landmarks and goal positions are generated randomly. We also use randomly generated maps of different shapes. The robot linear velocity is set to be a constant value $v_l = 1.0$ m/s. The angular velocity is a selection from the following set of values $(-2.0, -1.0, 0.0, 1.0, 2.0)$ rad/s. Both linear and angular velocities are added with Gaussian noises during the robot execution to simulate odometry errors. The reward elements r_{lost}, $r_{collision}$ and r_{goal} are -300, -300 and 600 respectively and the distance rate factor f is 10.

For the DQN-based RL framework, measurement data need to be converted into a discrete structure. The observed landmarks are first divided into 36 groups according to relative angles (10 degrees per group). The observation o_t consists of two value lists: $[lmin_1 \cdots lmin_{36}]$, where each element represents the value of the relative distance to the nearest landmark in that angle group and $[number_1 \cdots number_{36}]$, where each element represents the number of observed landmarks in that angle group. Using the restructured observation ($36 + 36$ dimensions) together with the relative goal position (2 dimensions), the agent state would be then represented by a 74-dimension vector. The input data is connected with 2 dense layers (512 nodes each) and the final layer uses a linear activation function, as shown in Fig. 4. Other DQN parameters are shown in Table. 1.

parameter	value
learning rate	0.00025
discount factor	0.99
epsilon decay rate	0.998
replay buffer	1000000
target network update rate	per 10 episodes

Table 1: DQN settings

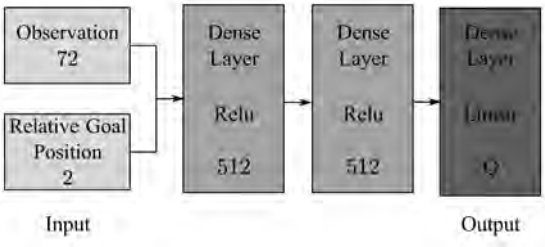


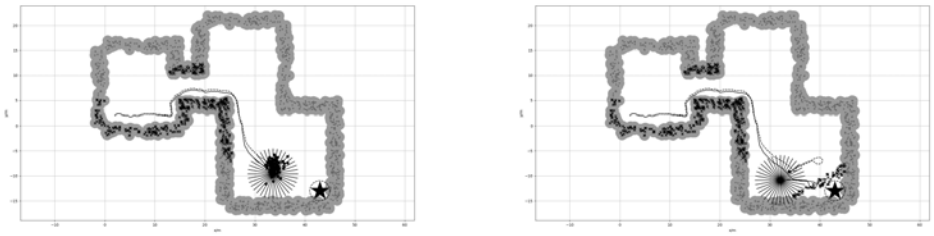
Fig. 4: The Q network structure

4.2 Results

As mentioned before, previous work always assume that the robot ground truth poses are accessible. Fig. 3b shows the trajectory, when a robot is provided with true poses and trained without the localisation failure penalty. The robot navigates through an open space diagonally to reach the goal position with a relatively short distance. In the real world, however, it is not always easy to obtain ground truth poses. When PF-based localisation is deployed in the same task, the robot will diverge from the true trajectory, as shown in Fig. 5, where the dash and solid lines are the estimated and ground truth trajectories respectively. During navigation, the divergence is caused by the poor observation of environmental features, as shown in Fig. 5a. In the case of navigation with diverged particles, when new features are observed, particles will be re-sampled to re-localise the robot to re-converge to a new pose estimated with respect to the new observed features. However, the estimated pose could potentially lose its original track and, hence, converge to wrong poses, as illustrated in Fig. 5b. The PF localisation will then fail catastrophically. In certain cases, due to the PF failures, the robot goal position might become unreachable for the robot due to the misaligned obstacles (black crosses at the bottom right corner in Fig. 5b). In this case, the robot will never reach the goal position.

The same experiments are performed with the additional localisation failure penalty r_{lost} introduced in our work. Fig. 7 shows the trajectories estimated using the same PF algorithm for localisation. As expected, it can be clearly seen that the estimated trajectories align closely to the true trajectories. It is also worth noting that the new trajectories tend to stay close to landmarks, to ensure high-quality landmark observation for robust localisation. Consequently, the robot can arrive at the goal successfully with only PF-based localisation. As mentioned, the performance improvement is mainly attributed to the new landmark-aware RL-based navigation policy, which encourages the robot to maintain a distance with good observation of features to ensure high localisation confidence.

The success rates evaluated at different training episodes are shown in Fig. 6. As can be seen, the success rate rises as the number of training episodes increases and stabilises at about 0.8 after training for 3000 episodes. The network is relatively simple and thus the training takes several hours on a workstation with an Intel i7-8700 CPU and an Nvidia RTX-2080 GPU.



(a) (b)

Fig. 5: Trajectories without localisation failure penalty (a) PF localisation diverges when no feature is observed, (b) PF localisation re-converges to wrong poses (solid line: ground truth trajectory; dashed line: estimated trajectory)

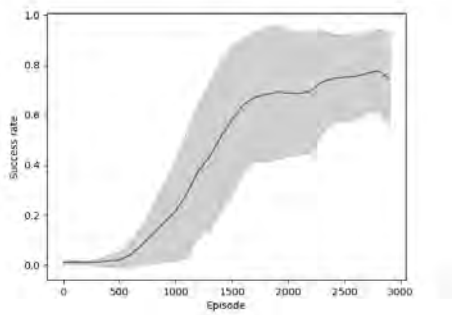


Fig. 6: Success rate

5 Conclusion

In this work, we introduced a novel DQN-based mapless navigation method that uses SLAM-based localisation for robot pose estimation, rather than relying on robot ground truth poses as used in previous works. A localisation failure penalty r_{lost} is introduced in the reward function to regulate agent behaviours to prevent robots from entering areas with no observable features, where SLAM-based localisation tend to fail. We performed different tests with and without using localisation failure penalty in different environments for training with randomised robot start/goal locations and maps. It can be clearly seen that our work considerably improves localisation performance attributed to the effectiveness of localisation failure penalty, which encourages a robot to follow paths with consistent observable landmarks while also free from collisions, hence fail-safe localisation.

References

1. Chiang, H.T.L., Faust, A., Fiser, M., Francis, A.: Learning navigation behaviors end-to-end with autorl. IEEE Robotics and Automation Letters **4**(2), 2007–2014 (2019)
2. Choi, S., Lee, K., Lim, S., Oh, S.: Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 6915–6922. IEEE (2018)
3. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE transactions on Robotics **23**(1), 34–46 (2007)
4. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397 (2016)
5. Kanezaki, A., Nitta, J., Sasaki, Y.: Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks. IEEE Robotics and Automation Letters **3**(2), 696–703 (2017)
6. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., et al.: Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673 (2016)
7. Moridian, B., Page, B.R., Mahmoudian, N.: Sample efficient reinforcement learning for navigation in complex environments. In: 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). pp. 15–21. IEEE (2019)
8. Niu, H., Ji, Z., Arvin, F., Lennox, B., Yin, H., Carrasco, J.: Accelerated sim-to-real deep reinforcement learning: Learning collision avoidance from human player. In: 2021 IEEE/SICE International Symposium on System Integration (SII). pp. 144–149. IEEE (2021)
9. Pfeiffer, M., Shukla, S., Turchetta, M., Cadena, C., Krause, A., Siegwart, R., Nieto, J.: Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. IEEE Robotics and Automation Letters **3**(4), 4423–4430 (2018)
10. Ruan, X., Ren, D., Zhu, X., Huang, J.: Mobile robot navigation based on deep reinforcement learning. In: 2019 Chinese control and decision conference (CCDC). pp. 6174–6178. IEEE (2019)
11. Tai, L., Li, S., Liu, M.: A deep-network solution towards model-less obstacle avoidance. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 2759–2764. IEEE (2016)
12. Tai, L., Liu, M.: A robot exploration strategy based on q-learning network. In: 2016 ieee international conference on real-time computing and robotics (rcar). pp. 57–62. IEEE (2016)
13. Tai, L., Paolo, G., Liu, M.: Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 31–36. IEEE (2017)
14. Wang, C., Wang, J., Zhang, X.: A deep reinforcement learning approach to flocking and navigation of uavs in large-scale complex environments. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). pp. 1228–1232. IEEE (2018)
15. Wang, Y., He, H., Sun, C.: Learning to navigate through complex dynamic environment with modular deep reinforcement learning. IEEE Transactions on Games **10**(4), 400–412 (2018)

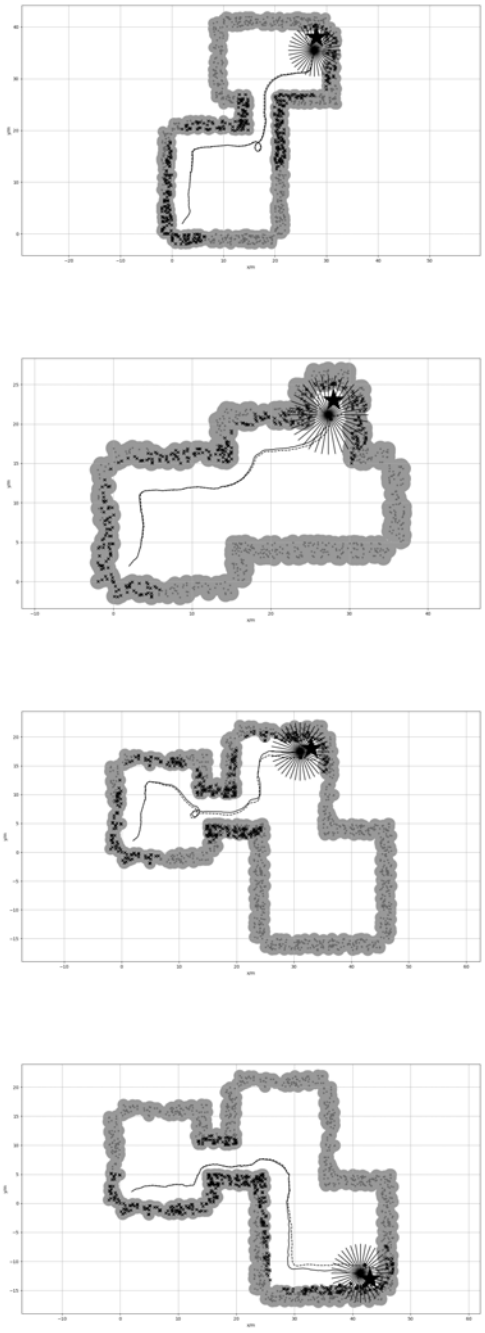


Fig. 7: Example trajectories generated with localisation failure penalty (solid black line: ground truth trajectory; dashed line: estimated trajectory)