Localisation-Safe Reinforcement Learning for Mapless Navigation

Feiqiang Lin[1], Ze Ji[1], Changyun Wei[2] and Raphael Grech[3]

***Abstract*— Most reinforcement learning (RL)-based works for mapless point goal navigation tasks assume the availability of the robot ground-truth poses, which is unrealistic for real world applications. In this work, we remove such an assumption and deploy observation-based localisation algorithms, such as Lidar-based or visual odometry, for robot self-pose estimation. These algorithms, despite having widely achieved promising performance and being robust to various harsh environments, may fail to track robot locations under many scenarios, where observations perceived along robot trajectories are insufficient or ambiguous. Hence, using such localisation algorithms will introduce new unstudied problems for mapless navigation tasks. This work will propose a new RL-based algorithm, with which robots learn to navigate in a way that prevents localisation failures or getting trapped in local minimum regions. This ability can be learned by deploying two techniques suggested in this work: a reward metric to decide punishment on behaviours resulting in localisation failures; and a reconfigured state representation that consists of current observation and history trajectory information to transfer the problem from a partially observable Markov decision process (POMDP) to a Markov Decision Process (MDP) model to avoid local minimum.**

I. Introduction

Mapless point goal navigation has been a popular research topic, where researchers wish to teach robots navigating to a target goal position in environments without maps available beforehand, such as under search and rescue scenarios. There are many conventional path planning algorithms designed for a broad range of scenarios or environments. These algorithms, however, have certain well known limitations [1]. For instance, they may require hand-crafted or constraint functions customised to specific application conditions, which make those algorithms lack of the generalisation capability for other different environments [2].

In order to improve its generalisation ability, learning-based navigation, such as reinforcement learning (RL), has gained more and more attention. Different from other learning based algorithms like supervised learning, RL requires little human resources or intervention during the training stage. Robots are trained through receiving continuous rewards or punishments through direct interaction with the environments. In the context of mapless navigation, RL-based algorithms have proven to be effective for mobile robots to gain promising performance of collision-free navigation in unknown environment [1].

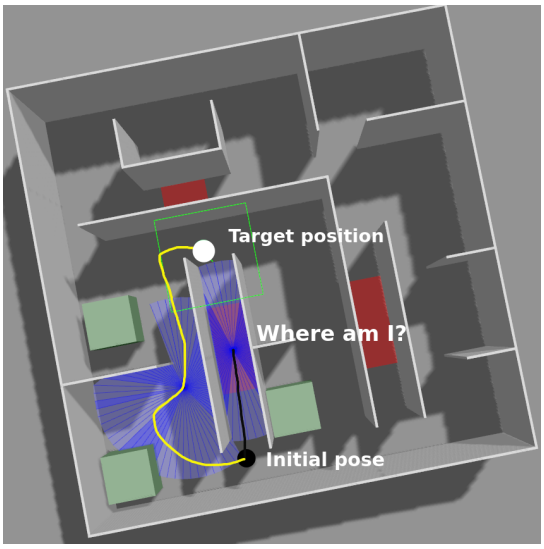

Fig. 1. Navigation examples (red regions: regions where localisation fails; black trajectory: unsuccessful trajectory into the symmetric corridor; yellow trajectory: successful trajectory avoiding the symmetric corridor.)

There are two different goal position representation formats in mapless navigation tasks: images or coordinates of goal positions. For image goals, agents may not need to know its current poses, but just remember configurations of the environments in order to find goal positions. Hence this kind of agents will limit its generalisation ability to unseen environments. In this work, we focus on goals represented by coordinates, where agents need to estimate its own position at each time step.

Among those RL algorithms for mapless navigation tasks discussed above, almost all of them make an unrealistic assumption that robots can obtain ground truth poses throughout the missions. This is unrealistic for real world applications. Usually, for robot navigation tasks, observation-based localisation algorithms, like Lidar-based or visual odometry, are needed to assist robot self-localisation in the real world. Thus, this work will take a more realistic position to address the mapless navigation task by the introduction of observation-based localisation algorithms for robot self-pose estimation.

However, observation-based localisation algorithms could fail where no sufficient or ambiguous observations are perceived along the trajectory. Taking Lidar-based odometry as an example, this kind of odometry will be problematic and not well constrained in long symmetric corridor envi-

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

[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]Spirent Communications, Paignton, UK raphael.grech@spirent.com

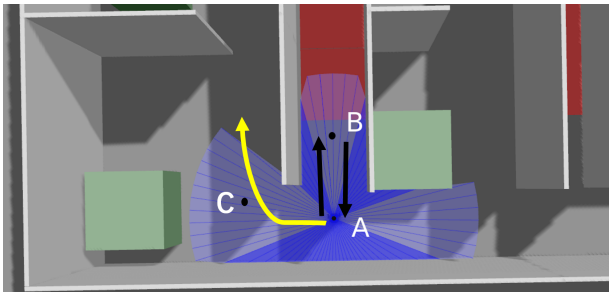

Fig. 2. The robot gets stuck in local minimum (the goal region is at the other end of the corridor)

ronments due to ambiguous laser scans received and will output inaccurate pose estimation [3]. For example, the red regions in Fig. 1 illustrate those corridor areas that present just two parallel and symmetric walls that are less distinct geometric features for Lidar to estimate motion changes. For a mapless navigation task, robots trained with ground truth poses will tend to take the shortest path strategy. As shown in Fig. 1, the robot would choose the black trajectory which however will lose its track due to the symmetric corridor. Such navigation tasks will fail without accurate localisation of robots or accurate relative pose to the corresponding goal positions.

In our work, we hypothesise that robots that are trained with observation-based self-localisation for mapless navigation will be able to be aware of regions with deteriorated localisation and may travel in other trajectories to avoid symmetric corridor regions or similar. For example, the yellow trajectory in Fig. 1 illustrates a less optimal trajectory in terms of length, but more reliable in achieving its goal due to the improved localisation performance.

However, one problem will arise when taking localisation performance into consideration. When robots are discouraged to enter localisation-unsafe regions, the robots are more likely to get stuck in local minimum, if this navigation problem is treated as an Markov Decision Process (MDP), as with many previous research works [4]. As shown in Fig. 2, the robot starts from point A and arrives at point B where it finds the corridor ahead is not traversable. It will return to the initial position A, where the same decision i.e. navigating to point B will be taken because history information is not used for decision making in the MDP setting. The robot will be trapped in local minimum (the black trajectories in Fig. 2). Hence, an MDP setting may sometimes fail the task.

To alleviate the problem above, we consider that the use of history information will help robots get out of local minimum regions, as illustrated by the yellow trajectory in Fig. 2.

The contributions of our work are summarised as follows:

- We propose an RL-based mapless navigation algorithm for localisation-safe navigation that localisation failures during navigation are avoided.
- Localisation-safe behaviours are learned by introducing a localisation performance-related reward, so that robots get punished, when localisation quality starts to deteriorate.
- We introduce a method based on the determinant of pose estimation covariance as the localisation performance metric.
- The Long Short-Term Memory (LSTM) network is introduced to feed robots with encoded history information, as part of state information of the RL algorithm, such that the original POMDP problem is converted to an MDP problem to address the local minimum problem.

The remainder of this paper is organised as follows. Section II introduces related works. Our method of this work is described in section III, followed by experiments and results in section IV. The conclusions are presented in section V.

II. Related Work

With the great advancement of Neural Networks (NNs), data-driven approaches have attracted consistently rising attention for the problem of mobile robot navigation. Supervised learning has been used to train neural networks, like Convolutional Neural Networks (CNNs) in an end to end style, as controllers for mobile robots to output motion commands based on observations like RGB images, depth images or Lidar scans [5], [6]. Because those methods usually require great amounts of labelled data, which would be costly to collect, they are often trained and tested in simulated worlds. For that reason, more recently efforts have been paid to techniques on how to transfer NNs trained in virtual environments to the real world [7], [8], [9].

Reinforcement learning (RL), in contrast to supervised learning, can be trained without human intervention by allowing agents autonomously exploring in environments to gain experiences. This self-learning nature has attracted significant attention from researchers in recent years for the mapless navigation problem. One prominent work was introduced by Tai et al in [10], where robots are trained with RL by a two-step method with depth images as inputs. Another noticeable work was introduced in 2017 [1] that demonstrated the effectiveness of training a mapless navigation agent in a 2D environment using a 2D planar Lidar. Many works have inherited the above approach and derived various variations for further improvement [11]. Deep Q-networks have been enhanced with state-of-the-art technologies like the duel architectures and double networks to improve navigation performance with discretized action space [12]. For navigation in continuous action space, policy-based RL, like the asynchronous deep deterministic policy gradient (DDPG) algorithm, was proposed in [1]. RL for continuous action space needs more experience data than discretized ones. Thus work has been focused on how to improve data efficiency in [13] by first using imitation learning to pre-train the agent network and then fine-tune the pre-trained agent network with the constraint policy optimisation (CPO) reinforcement learning. An action scheduling mechanism during training is introduced to perform more efficient exploration and exploitation [14]. There are also other improvements regarding to sampling efficiency [15] or

optimising hyper parameters [16]. As training could be difficult when visual images are taken as inputs, reinforcement learning with auxiliary tasks has been proposed to achieve better state representations for navigation tasks from image inputs [17], [18], [19].

Although works introduced above have obtained relatively promising results, to the authors' best knowledge, none of these learning-based algorithms have considered the effect of localisation performance on the final navigation results, as illustrated in Section I, except the preliminary work [20] by the authors. They all assume the availability of ground truth robot poses during navigation and result in policies alike shortest path strategies without considering localisation quality along navigation trajectories. In other words, robot perception and navigation strategy are decoupled. A similar learning-related work taking localisation performance into consideration is [21]. However, the learning part is only used for perception to score different environment regions depending on whether the region is favourable for localisation or not with image inputs. The planning part is still using conventional planing algorithm. Thus, it is different from what our work tries to achieve: utilising RL to train an agent to learn to 1) navigate in a mapless environment, 2) avoid collisions, and 3) traverse localisation failure-free trajectories.

III. Method

This section first introduces the basic knowledge of RL, followed by two ideas proposed in our proposed RL algorithm of localisation failure-free mapless navigation. The algorithm implementation details will be also introduced.

Markov Decision Process (MDP) is a mathematical framework to model problems that can be solved using RL. To describe a MDP, we need a state space S, an action space A, a state transition model $p(s_{t+1}|s_t, a_t)$, and a reward function $r(s_t, a_t)$: $S \times A \rightarrow R$, which describes the feedback from the environment after agents taking actions a_t at state s_t. The objective is to obtain maximum overall discounted rewards $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ from environments. RL algorithms usually involve estimating an state-action pair value function $Q_\pi(s_t, a_t) = E[R_t|s_t, a_t]$ or state value function $V_\pi(s_t) = E[R_t|s_t]$, where π: $a \sim \pi(\cdot|s, \theta)$ described by parameters θ is the policy that agents try to learn.

There are two remarks from above description. First, RL requires problems satisfying the Markov property $p(s_{t+1}|s_t, a_t)$: the next state s_{t+1} depends purely upon the present state s_t. The past does not have effects on the future. This property does not be satisfied for a POMDP problem, where agents are unable to receive states s_t information but only observations o_t from a state conditioned distribution $p(o_t|s_t)$. The Markov property is broken as $p(o_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, ..., o_0) \neq p(o_{t+1}|o_t, a_t)$. In the mapless navigation settings, the robot can only have observations of its surrounding environments, which could be formed as a POMDP problem. Also in this work, the robot does not have ground truth poses, but, instead, has to make pose estimation from observations with localisation algorithms using Lidar or visual odometry. This would formulate our task as a POMDP problem. Second, the learned policy will depend on the rewards feedback from environment. As this work aims to train agents learning to navigate with localisation-safe policies, a metric to decide a reward related to localisation performance will need to be proposed.

A. POMDP to MDP

As discussed above, mapless navigation that uses estimated localisation could be viewed as a POMDP problem from the following two perspectives.

Firstly, from the localisation perspective, the robot pose estimation at current time step will require historical poses estimated as shown in Eq. (1).

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

where $x_{1:t}$, $z_{1:t}$, $u_{1:t-1}$ are robot poses to be estimated, observations, and control inputs at different time steps; m represents stored features set. From above equation, we can learn that the poses estimation built on past wrongly estimated poses caused by insufficient or ambiguous observations will be also wrong. As estimated robot poses and observations are part of agent state input data, continuing training on those problematic data will misguide the robot learning. Hence, we suggest to end training the episodes, where localisation starts to diverge from ground truth. One may suggest using pose estimation errors as a localisation performance criterion. However, it takes time to accumulate position errors to a noticeable scale. The experiences between this time can still do harm to the training procedure. We use the determinant of the estimation covariance calculated from hessian matrix as the metric which will be discussed later.

From the perspective of mapless settings, a robot does not have access to global maps. Robots may get stuck in local minimum regions, where robots hesitate to enter certain localisation-unsafe regions, as shown in Fig. 2 in Section I. In order to solve this and turn this POMDP problem into an MDP problem for RL algorithms, we propose to combine history information $h_{t-1} = (o_{t-1}, o_{t-2}..., o_0)$ with the current observation o_t as an approximation of the current state s_t. The entire complete history trajectories could be resource consuming to store and is not efficient to be used as the neural network inputs. In this work, we propose to employ the LSTM architecture, which is designed for processing sequences of data, to encode history information represented by hidden values h_t at each time step t that are reserved as the input for calculation in the subsequent time steps as shown in Fig. 3.

B. Reward Metric

As introduced above, a reward related to localisation performance is of significant importance for the robot to learn a localisation-safe strategy. Different criteria exist for measuring localisation performance. In this work, we propose to use the determinant of the covariance matrix, $\det C$, as the measure of localisation quality. To compute the covariance, we use the inverse Hessian matrix to compute

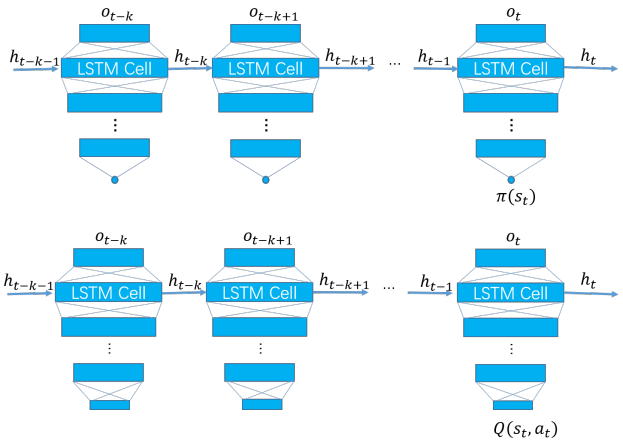

Fig. 3. LSTM for encoding history information

the covariance matrix of the poses estimated by localisation algorithms [22]. This principle is not limited to any particular sensor modality that could include Lidar-based algorithms, such as Hector [23] or visual odometry algorithms, such as Semi-direct Visual Odometry (SVO) [24]. We hypothesise that the estimated covariance could provide a measure of localisation performance and be used to produce localisation related reward.

The following explains how to derive the covariance matrix using the inverse Hessian matrix. To formulate the localisation problem, an observation model is needed to establish the error function which needs to be optimised for estimating robot poses. For simplicity of illustrating , we first assume the problem can be modelled as a linear model, so that linear regression can be performed, and the covariance matrix can be derived analytically. In the case of a linear model, the model can be described as below:

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{w} \tag{2}$$

where $\mathbf{y}$ is the measurement, $\mathbf{M}$ is the observation matrix, $\mathbf{x}$ represents model parameters, which is the to-be-estimated pose $[x, y, \theta]^T$ and $\mathbf{w}$ represents Gaussian noises $\mathbf{w} \sim N(0, \sigma^2 I)$. The to-be-optimised error function becomes:

$$\mathbf{E}(\hat{\mathbf{x}}) = (\mathbf{y} - \mathbf{M}\hat{\mathbf{x}})^T(\mathbf{y} - \mathbf{M}\hat{\mathbf{x}}) \tag{3}$$

where $\hat{\mathbf{x}}$ is the estimated state, as formulated in the estimated model $\hat{\mathbf{y}} = \mathbf{M}\hat{\mathbf{x}}$. In this linear case, this error function can be solved analytically, as below:

$$\hat{\mathbf{x}} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y} \tag{4}$$

The covariance matrix can be then represented as:

$$\mathbf{C}(\hat{\mathbf{x}}) = (\mathbf{M}^T\mathbf{M})^{-1}\sigma^2 \tag{5}$$

The Hessian matrix for Eq. (3) can be represented as:

$$\mathbf{H} = \frac{\mathrm{d}\mathbf{E}^2(\hat{\mathbf{x}})}{\mathrm{d}\hat{\mathbf{x}}^2} = 2\mathbf{M}^T\mathbf{M} \Rightarrow \mathbf{M}^T\mathbf{M} = \frac{1}{2}\mathbf{H} \tag{6}$$

Hence, from Eqs. (5) and (6), the covariance matrix can be derived:

$$\mathbf{C}(\hat{\mathbf{x}}) = (\frac{1}{2}\mathbf{H})^{-1}\sigma^2 \tag{7}$$

In most cases, the measurement model function usually is not linear. However, iterative optimisation algorithms like Gaussian-Newton is usually used where the error function is linearized in a local region and minimized step by step. Once the minimum is found, the Hessian matrix can be directly calculated at that minimum point and thus the covariance matrix can be estimated according to Eq. (7).

Finally, as discussed above, the determinant of covariance $\det \mathbf{C}$ will be used as the measure of localisation uncertainty or failure and is monitored against a threshold. When the determinant $\det \mathbf{C}$ is above the threshold, the training episodes will be terminated. A negative reward will be applied to penalise the behaviours that result such failures.

C. Implementation of Localisation-safe Navigation RL

In this section, we explain the details of RL implementation that incorporates the two techniques above: LSTM mechanism and the localisation covariance-based reward metric. We aim to demonstrate the effectiveness of considering localisation quality as the reward metric during navigation to avoid localisation-unsafe regions and solving the local minimum problem introduced. In principle, the proposed work would not be limited to any specific RL algorithm. Therefore, this paper will use the Deep Q-Network (DQN), because of its wide adoption, simplicity, and easy implementation. The detailed configurations of the proposed RL algorithm for localisation-safe navigation are discussed in the following paragraphs.

In DQN, the state-action value $Q_\pi(s, a) = E[R_t|s_t = s, a_t]$ can be represented by a Deep Neural Network (DNN), which is trained through interaction with the environment. As discussed above, the DNN should need an LSTM network component stacking with other types of networks, such as fully connected networks, to enable the agent to estimate its states with encoded history information as shown in Fig. 3. The DQN agent input includes the sensor measurement o_t and the relative goal position g_t, which consists of relative distance to the goal d_g and relative goal heading β with respective to the robot. As we use DQN, the agent action space is discretized. At each time step, the agent decides its linear velocity v_{linear} from a set of given values $[v_{l_1}, v_{l_2}...v_{l_i}]$ and an angular velocity within a set of values $[w_1, w_2...w_j]$.

To design the reward function, as this DQN agent relies on observation-based localisation algorithms for pose estimation, the agent should not only be optimised towards its obstacle avoidance capability and optimal path length, but also needs to avoid selecting actions that lead to localisation failures. Thus, the reward function is designed as follows:

$$r = \begin{cases} r_{lost} & \text{if } \det \mathbf{C} >= th \\ r_{collision} & \text{if collision happens} \\ r_{goal} & \text{if } d_g < d_{gmin} \\ f \times (d_{t-1} - d_t) & \text{otherwise} \end{cases} \tag{8}$$

where r_{lost} is a negative value, when the localisation failure metric is satisfied, i.e. the determinant of the estimated covariance $\det \mathbf{C}$ exceeding a threshold th; $r_{collision}$ is negative to punish the agent when colliding with an obstacle; the agent is rewarded with a positive value r_{goal}, when it reaches the point goal position within an acceptable range d_{gmin}; the last term $f \times (d_{t-1} - d_t)$ is designed in such a way, so that the agent will be encouraged to select actions that tend to reduce the distance between itself and the goal; d_t is the distance from the robot to the goal at time step t; and f is the distance rate factor.

In brief, most previous research works do not consider the penalty r_{lost} in the reward space. However, this is an essential component to prevent agent from selecting actions that move the robot into localisation-unsafe regions, where localisation algorithm tends to fail. This work proposed a localisation failure criteria $\det C$ to decide when to apply this penalty, as described in section III-B.

IV. Evaluation

A. Experiments Setup

We test our algorithm by deploying a Turtlebot-3 mobile robot equipped with a 2D-Lidar sensor in Gazebo simulation environments. Our method however is not limited to Lidar sensor module as discussed in reward metric part (section III-B).

The observation state o_t for the agent consists of laser scan data $(l_1 \cdots l_{72})$ and the relative pose to the goal, represented in the polar coordinate (relative distance and angle to the goal (d_g, β)). This would then form a 74-dimension state vector as the input for the agent network.

Considering that DQN is used in our work, the action space contains a set of angular velocities: $(-1.5, -0.75, 0.0, 0.75, 1.5)$ rad/s. To simplify the problem, the robot linear velocity is set to be a constant value $v_l = 0.1$ m/s. The network configuration is shown in Fig. 4, which consists of one LSTM cell, and 3 Dense layers.

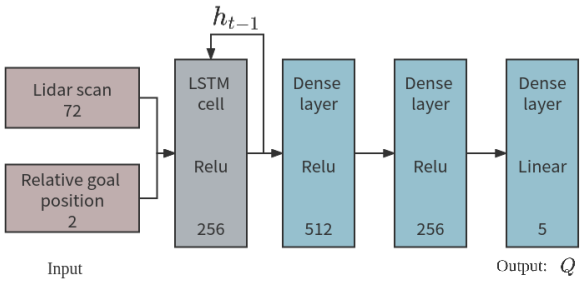

Fig. 4. Agent network architecture

In terms of pose estimation, we use the popular Lidar-based localisation algorithm, the Hector SLAM [23]. However, the maps built by Hector are not used by the agent, hence mapless navigation.

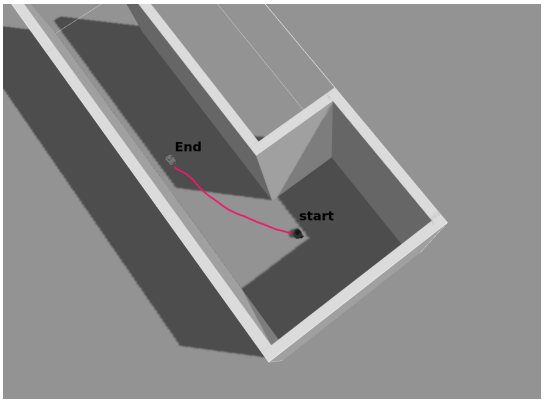

Fig. 5. Robot travelling from room into corridor

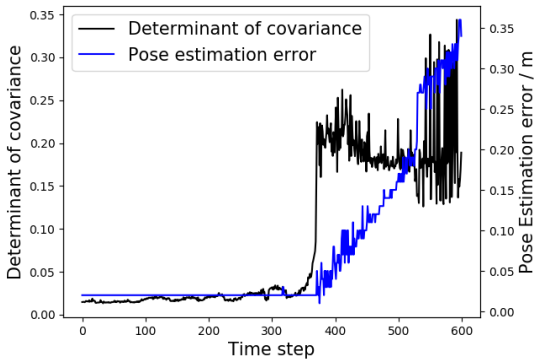

Fig. 6. Determinant values of covariance and pose estimation errors

B. Inverse of Hessian Matrix as Reward Metric

In this subsection, we first demonstrate the effectiveness of the value of determinant $\det \mathbf{C}$, on how it may affect the robot's behaviours, e.g., if the robot could avoid areas with deteriorated localisation. For illustration purpose, Fig. 5 shows one scenario that the robot located in a room, which has distinct geometric features, moves into a corridor, which is challenging for Lidar-based localisation due to ambiguous symmetric features in the corridor.

As discussed before, one obvious metric could be the error of the estimated pose. We therefore here compare the pose errors between the estimated robot poses and the ground truth poses and the determinants $\det \mathbf{C}$ along the trajectory. As shown in Fig. 6, we can see that, after the robot enters the corridor at the time step of about 380, the pose estimation error starts to rise gradually, while, in contrast, the determinant sharply moves upwards and remains high afterwards with a certain level of fluctuations.The pose estimation errors grow steadily and it takes quite a number of time steps to reach a noticeable level. This is not desirable to be used as a reward metric in our context, as it is unable to provide immediate response to localisation failures. On the contrary, as shown in the figure, the proposed determinant of the covariance can more promptly indicate when the robot starts to lose its track, and, hence, penalise the behaviours

of the RL agent immediately. In this work, we have tested extensively on similar scenarios, and have observed similar phenomena of the determinant value changes.

C. DQN Training Results

The following introduces how we train the DQN agent. In each episode, the robot is re-spawned at a random position. The goal position is also randomly decided. The episode ends when any of the following cases happens: 1) the robot collides with any obstacle or wall; 2) the determinant value of the covariance is higher than a pre-defined threshold (indicating deteriorated localisation, e.g. robot entering a symmetric corridor area in this environment); 3) the robot reaches the goal region; 4) or the maximum time step is met. The training of the policy network follows the standard DQN algorithm.

Fig. 7 shows the result of agent's navigation policy. It can be seen that the success rate reaches about 85% after training of about 3300 episodes. This is considered promising for this DQN-based learning implementation. Fig. 8 shows some trajectories of the agent trained with our method, where the red circles represent the goal regions. It is clear that the performance is similar to or comparable with the original mapless navigation which assumes the availability of ground-truth poses.

Since it is unfair to directly compare the original mapless navigation, as it assumes the availability of ground-truth poses. For comparison purpose, in our work, we periodically calculate the failure rates for only those cases that failed because of the problem of unsuccessful localisation, i.e., when $\det \mathbf{C}$ is above the threshold. As shown in Fig. 7, the localisation-related failure rate drops from 14% to 6% (green line), that clearly contributed to the overall success rate of navigation. This proves our hypothesis that penalising behaviours of entering regions with deteriorated localisation is effective in improving its navigation performance.

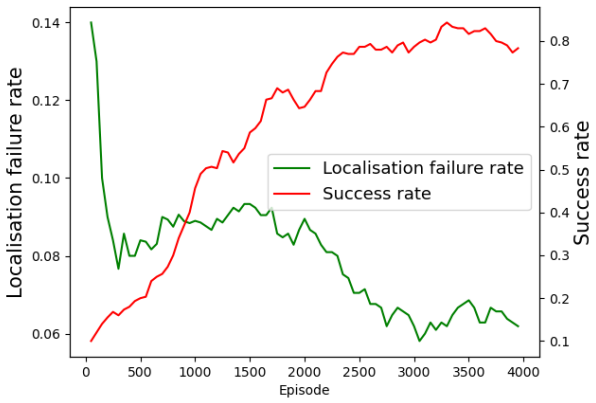

Fig. 7. Success rate and localisation failure rate

To further demonstrate the effectiveness of the proposed RL policy, we created a few specific scenarios, for illustration purpose, that the robot will be initially spawned at the

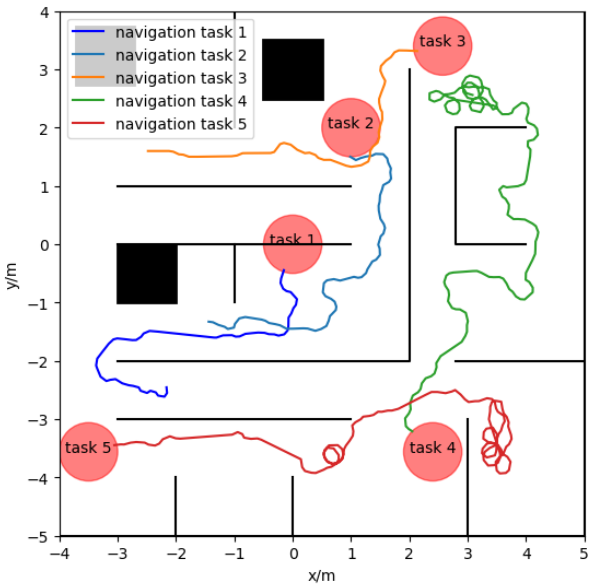

Fig. 8. Example trajectories of the agent trained with our method with randomly located start and end positions. (red circles: goal region)

entrances of some symmetric corridors and the corresponding goals are located at the other end of the corridor. With most path planning algorithms, the robot will choose to traverse the corridor directly. As shown in Fig. 9, the agent trained without localisation-related reward r_{lost} will behave in a similar way to the shortest path strategy. However, traveling in corridors will lead the localisation to failures and pose estimation will be wrong as discussed previously (Fig. 6). Consequently, it will miss the targets although it may actually already pass through the goal regions. The agent that does not know its true state will end up with a wrong position due to the inaccurate poses estimation.

With our proposed method, as expected, the capability of robot avoiding entering localisation failure regions has been greatly enhanced. Fig. 10 shows the trajectories generated using the new algorithm. We can see that the robot will avoid corridor areas that are unsuitable for Lidar-based localisation.

For each task, the agent will first hesitate slightly around the corridor entrance and gain some path information ahead of it. As we feed the agent with encoded historical information, the agent will be well informed that there is a symmetric corridor (dangerous regions for Lidar-based localisation) ahead of it and will decide to navigate along a more distant path. The learned capability to avoid entering deteriorated localisation regions is due to the introduction of our proposed punishment reward r_{lost} produced during training. Also the use of LSTM cell enables observations from the past to be encoded and passed to later time steps as hidden state values of the network as shown in Fig. 3, which, in this localisation case, gives the robot the ability to remember where it has been and what it has seen in the past time steps. Consequently, after the robot has thoroughly explored the local regions, it will acquire enough environment information around it and will make decisions based on information

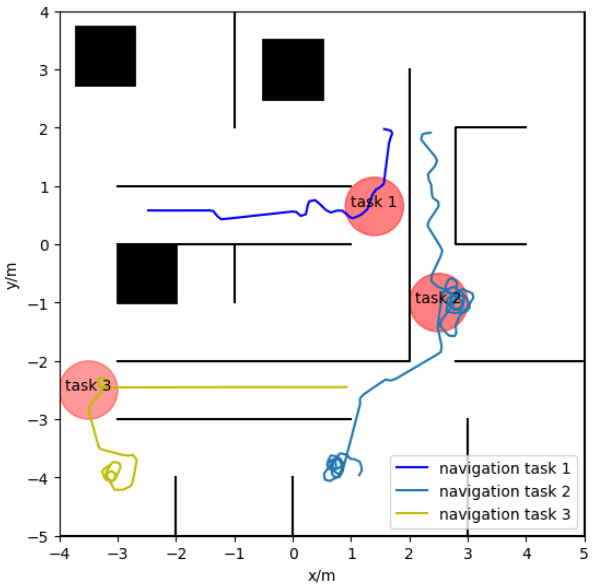

Fig. 9. Trajectories of the agent without localisation related reward r_{lost} (red circles: goal region)

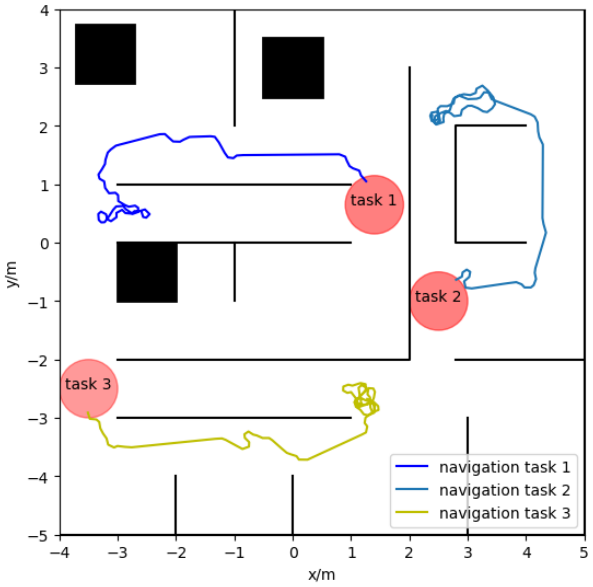

Fig. 10. Trajectories of the agent trained with our method (red circles: goal region)

collected. Finally, the robot will escape the local minimum region.

In addition, we tested the agent in unseen environments of different layouts, corridor entrance conditions and sizes (Fig. 11, Fig. 12, Fig. 13). As shown in Fig. 11, the robot can still navigate to the goal position without traveling in the region dangerous for localisation (the red region). However, interestingly, the agent behaved more conservatively than needed in this case. At the beginning, the robot hesitated at the two entrances of the two corridors, including one corridor that is alike a symmetric corridor (the blue region in the figure), which is actually safe for localisation. The robot

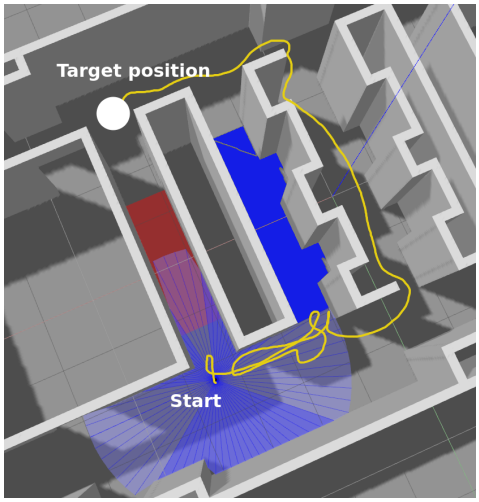

Fig. 11. A navigation example in the unseen environment-1

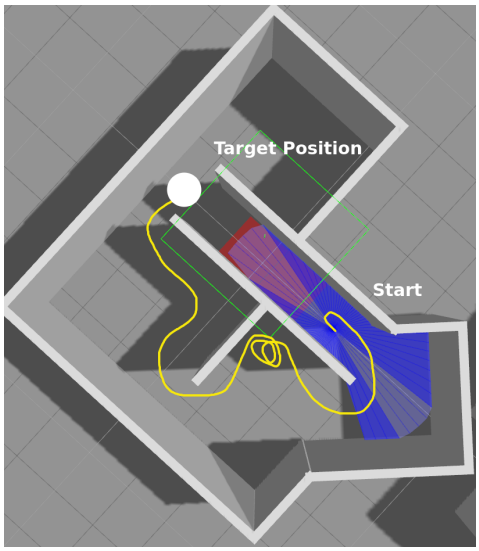

Fig. 12. A navigation example in the unseen environment-2

finally abandoned entering this passage and chose another one which presents more distinct features for localisation at the entrance of the corridor.

V. Conclusion

In this work, we proposed a novel RL-based mapless navigation method that does not rely on the availability of ground truth robot poses as assumed in other works. Instead, our work deploys on-board localisation algorithms for robot pose estimation. In order to train a robot to avoid navigating in regions where localisation algorithms would fail, we designed a penalty r_{lost} to regulate robot behaviours. We proposed using the determinant of the pose estimation covariance as the measure of localisation failures to decide when to punish the robots. Because of the use of pose estimation algorithm, the agent may encounter local minima more easily. For this problem, we proposed to incorporate history information embedded for decision making of the

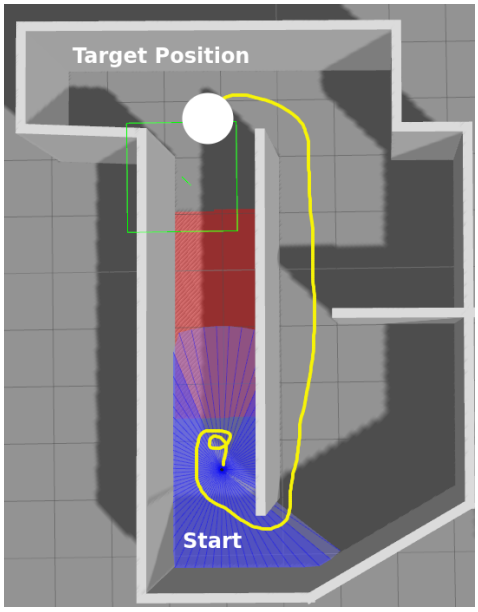

Fig. 13. A navigation example in the unseen environment-3

robot by stacking an LSTM cell in the robot policy network. We tested our proposed method with the DQN architecture. It can be clearly seen that the agent has learned the ability of mapless navigation, while avoiding localisation-unsafe regions or getting trapped in local minimum regions, thanks to the localisation related penalty and the involvement of history information in state representation in this work.

REFERENCES

[1] L. Tai, G. Paolo, and M. Liu, "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)*. IEEE, 2017, pp. 31–36.

[2] C. Wang, J. Wang, and X. Zhang, "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)*. IEEE, 2018, pp. 1228–1232.

[3] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8729–8736.

[4] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," *arXiv preprint arXiv:1804.00456*, 2018.

[5] A. Kanezaki, J. Nitta, and Y. Sasaki, "Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 696–703, 2017.

[6] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 2759–2764.

[7] S. Choi, K. Lee, S. Lim, and S. Oh, "Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6915–6922.

[8] K. Lee, H. Kim, and C. Suh, "Crash to not crash: Playing video games to predict vehicle collisions," in *ICML 2017*, 2017.

[9] L. Tai, P. Yun, Y. Chen, C. Liu, H. Ye, and M. Liu, "Visual-based autonomous driving deployment from a stochastic and uncertainty-aware perspective," *arXiv preprint arXiv:1903.00821*, 2019.

[10] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in *2016 ieee international conference on real-time computing and robotics (rcar)*. IEEE, 2016, pp. 57–62.

[11] H. Niu, Z. Ji, F. Arvin, B. Lennox, H. Yin, and J. Carrasco, "Accelerated sim-to-real deep reinforcement learning: Learning collision avoidance from human player," in *2021 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2021, pp. 144–149.

[12] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *2019 Chinese control and decision conference (CCDC)*. IEEE, 2019, pp. 6174–6178.

[13] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[14] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Transactions on Games*, vol. 10, no. 4, pp. 400–412, 2018.

[15] B. Moridian, B. R. Page, and N. Mahmoudian, "Sample efficient reinforcement learning for navigation in complex environments," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2019, pp. 15–21.

[16] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[17] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[18] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[19] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectgoal navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 117–16 126.

[20] F. Lin, Z. Ji, C. Wei, and H. Niu, "Reinforcement learning-based mapless navigation with fail-safe localisation," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2021, pp. 100–111.

[21] L. Bartolomei, L. Teixeira, and M. Chli, "Semantic-aware active perception for uavs using deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3101–3108.

[22] O. Bengtsson and A.-J. Baerveldt, "Robot localization based on scan-matching—estimating the covariance matrix for the idc algorithm," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 29–40, 2003.

[23] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*. IEEE, 2011, pp. 155–160.

[24] Z. Zhang and D. Scaramuzza, "Perception-aware receding horizon navigation for mavs," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2534–2541.