This item is the archived peer-reviewed author-version of:

Directed real-world learned exploration

Reference:
Hutsebaut-Buysse Matthias, Guinjoan Ferran Gebelli, Rademakers Erwin, Latré Steven, Temsamani Abdellatif Bey, Mets Kevin, Mannens Erik, De Schepper Tom.- Directed real-world learned exploration
Proceedings of the International Conference on Intelligent Robots and Systems - ISSN 2153-0858 - IEEE, 2023, p. 5227-5234
Full text (Publisher's DOI): https://doi.org/10.1109/IROS55552.2023.10341504
To cite this reference: https://hdl.handle.net/10067/2015990151162165141

uantwerpen.be
Institutional repository IRUA

Directed Real-World Learned Exploration

Matthias Hutsebaut-Buysse[1], Ferran Gebelli Guinjoan[2], Erwin Rademakers[2], Steven Latré[1], Abdellatif Bey Temsamani[2], Kevin Mets[1], Erik Mannens[1] and Tom De Schepper[1]

***Abstract*— Automated Guided Vehicles (AGV) are omnipresent, and are able to carry out various kind of pre-programmed tasks. Unfortunately, a lot of manual configuration is still required in order to make these systems operational, and configuration needs to be re-done when the environment or task is changed. As an alternative to current inflexible methods, we employ a learning based method in order to perform directed exploration of a previously unseen environment. Instead of relying on handcrafted heuristic representations, the agent learns its own environmental representation through its embodiment. Our method offers loose coupling between the Reinforcement Learning (RL) agent, which is trained in simulation, and a separate, on real-world images trained task module. The uncertainty of the task module is used to direct the exploration behavior. As an example, we use a warehouse inventory task, and we show how directed exploration can improve the task performance through active data collection. We also propose a novel environment representation to efficiently tackle the *sim2real* gap in both sensing and actuation. We empirically evaluate the approach both in simulated environments and a real-world warehouse.**

I. Introduction

Automated Guided Vehicles (AGV) have started to emerge in various industry settings. These vehicles are often utilized to transport various goods from one place to another. In order to perform these tasks, they often rely on navigation systems which are based on markings on the floor, or rely on way points outlined in static prior maps [1], [2]. These systems, however, often rely on highly accurate sensors and dynamics models, require intensive prior configuration, are not able to handle dynamic environments well, and lack robustness [3].

Besides these limitations, it is also often yet unclear how to move beyond pure transportation tasks. For example, if one wants an AGV to find a certain object in the environment, heuristic modules which are prone to error propagation, are often required in order to go from object class input to a specific set of navigation world coordinates.

Reinforcement Learning (RL) has been utilized as an end-to-end learning based alternative approach in which noisy (high-dimensional) inputs can be directly utilized in order to output low-level actuation control actions. RL utilizes trial-and-error learning to allow the agent to come up with a policy and task oriented state representation solely from a reward signal, provided during training. In this manner, RL is capable of implicitly learning the agent affordances and world dynamics, without requiring explicit access to them.

[1]University of Antwerp - imec IDLab - Department of Computer Science, matthias.hutsebaut-buysse@uantwerpen.be

[2]Flanders Make, ferran.gebelli@flandersmake.be

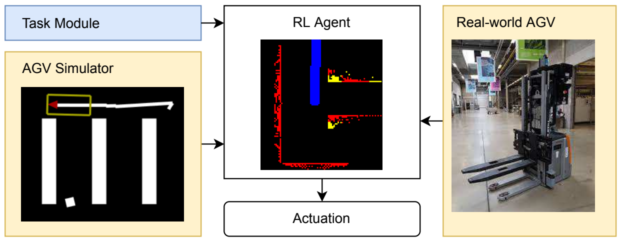

Fig. 1. A RL agent is trained in simulation. As the introduced state representation (middle) can be either obtained from the simulator, or real-world sensors, the trained policy can be directly used on a real-world AGV platform. In order to direct the exploration, a separate task module is utilized.

RL has been successfully applied in various domains including controlling stratospheric balloons [4] and nuclear reactions [5]. Unfortunately RL approaches are also plagued with a sample efficiency issue. The amount of required interactions with the environment is often too large, or too unsafe, in order to currently offer a practical solution [6]. This problem is aggravated in the typical RL setup, because a new policy is often trained from scratch for each new encountered task.

In this paper, a novel RL approach is presented which goes beyond pure navigation, allowing an AGV to perform various directed exploration [7] tasks. Such tasks require the agent to actively explore a previously unseen environment in an intelligent (directed) manner. Examples of such tasks might consist of patrolling, searching a specific object, or counting warehouse inventory.

In order to make it feasible to utilize an RL based directed exploration approach in a real-world environment, we propose a novel method for training a directed exploration policy in simulation, which can then be utilized for various downstream real-world exploration tasks, without expensive retraining. Training in a simulated environment allows running multiple environment instances in parallel, allows executing actions safely and at a much higher frequency than would be possible in the real world.

However, when simulating sensors, inconsistencies between simulated sensors and their real-world counterparts are often unavoidable. This gap is called the sim2real gap. In order to minimize this gap, a Light Detection And Ranging (LiDAR)-based state representation is introduced in this paper. We demonstrate that this approach is robust to sensor noise, and thus qualified to bridge the sim2real gap. While LiDAR-based simulations are often very compute intensive [8], the presented approach is less compute intensive, supporting above real-time simulation.

The introduced approach of directed exploration additionally offers a loose coupling between the exploration navigation policy and a task module. This loose coupling allows the re-usage of a trained navigation policy in order to perform multiple tasks, without re-training the navigation policy. We demonstrate how a separately supervised trained task module steers the RL policy in order to reduce the uncertainty of its predictions by actively navigating towards better observations.

The presented framework can be applied to a wide range of applications. For the evaluation of the framework, the task of automated warehouse inventory will serve as a use case. In this use case the uncertainty of an inventory box counter will be utilized in order to direct the exploration.

The contributions of the work are the following:

- A novel approach to make learned RL exploration directed through integration with a separately trained task specific module.
- An embodied training approach capable of (1) learning AGV affordances end-to-end, and (2) balancing directed and more generic exploration.
- A novel representation based on LiDAR point clouds that is able to robustly bridge the sim2real gap between training in simulation and real-world usage.
- A warehouse simulator with procedurally generated warehouse layouts, that can be utilized to further build upon the presented work.

II. Preliminaries

A. Reinforcement Learning

The problem studied in this paper can be modeled as a Markov Decision Process (MDP), represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. On each time step t, the agent samples an action $a_t \in \mathcal{A}$ from its policy $\pi(a_t|s_t)$, and the environment produces in turn a state $s_t \in \mathcal{S}$ according to an to the agent unknown transition function $\mathcal{P}(s_{t+1}|s_t, a_t)$. The agent receives feedback from an to the agent unknown reward function $r_t(s_t, a_t)$. The goal of RL consists of finding a policy π which maximizes the sum of rewards R_t, discounted by a factor $\gamma \in [0, 1]$, through interaction with the environment.

B. Proximal Policy Optimization (PPO)

PPO [9] is the on-policy RL algorithm used for training in this study. In this algorithm the agent alternates between a policy evaluation phase, in which the current policy is utilized to collect new samples, and a policy improvement phase, in which the agent updates the policy based on the collected samples from the last evaluation phase. Key to PPO is that during the improvement phase the new policy is constrained in order to not deviate too much from the previous policy. This approach emulates monotonic policy improvement [10] without getting stuck in a local optimum.

Different RL algorithms explore in different ways. PPO handles exploration by sampling actions from the current policy stochastically. The amount of randomness is typically reduced over time during training, as the agent gains more confidence which action will result in the highest return. The proposed method is however not limited to PPO, other RL algorithms can also be used for training.

III. Related Work

A. Navigation and Exploration

Classic navigation pipelines typically construct a map, and localize the agent within the map. These so called Simultaneous Localization and Mapping (SLAM) approaches (surveyed in [3]) then allow a planner [11], [12], [13] to construct a plan given a navigation destination. However, this still requires a different module to set destinations. This is often done in a greedy fashion by iteratively navigating to the frontiers between what is known and the unknown space [14], [15]. The results of these methods, however, heavily depend on the quality of the sensor observations [16]. Furthermore, the major limitation of these methods, and the delta with our approach, is that prior methods often ignore any semantic cues that can be found in the environment, and consider the problem to be of a purely geometric nature.

B. Learned Exploration

Exploration is a key component of RL, and is utilized to test out actions in order to find a suitable policy, capable of solving the task at hand. However, if the task consists of exploring the environment efficiently the task is called learned exploration.

In order to perform this task, [16] generates a map from depth camera observations and utilizes the improvement in coverage of the constructed map as its reward signal. [17] learns to construct a map in a supervised fashion, and utilizes this map as input to a policy trained on maximizing coverage of this map.

Curiosity based exploration aims to reward the agent for visiting states which the agent could not accurately predict. Using only a curiosity reward, [18] demonstrated an agent capable of solving a sparsely rewarded navigation task.

Most similar to our work is [19], which rewards the agent for removing uncertainty of a separately trained perception model. In [19] all experiments are limited to a single task.

C. Sim2real

A straightforward way to minimize the sim2real gap is to make the simulator as close to the real world as possible. However, this is a complex task as RL agents tend to overfit on low-level details in the simulated environment [20]. In order to combat this, [21] demonstrated that a lower fidelity simulator actually performs better when transferring policies to a real-world context. Furthermore, a lower fidelity simulator tends to be less compute intensive, and thus allows for better scaling. Similarly to the setup introduced in this paper, [22] trains both on a geometrically accurate simulator and a collection of localized real-world images.

Using a LiDAR as an input to an RL setup has been studied before [23] in a pure low-level motion planner setting. The presented approach however goes further and also handles high-level autonomous goal selection (e.g., uncertain areas to explore).

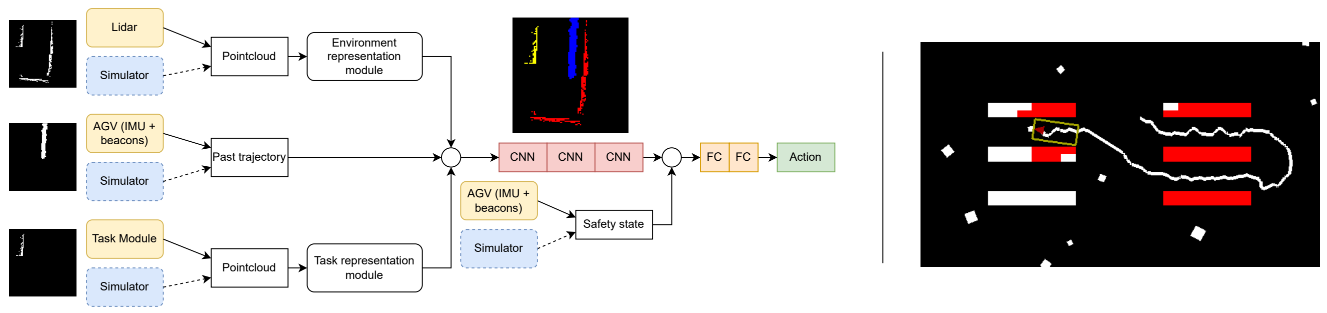

Fig. 2. An overview of the architecture of the navigation module. The different inputs are either coming from a real AGV or the simulator. These inputs are processed into an egocentric 3 channel image. Together with the safety state this can be utilized to directly output actions. On the right a top-down view of the simulator is plotted. Racks are plotted in white or red squares depending on whether the agent (yellow rectangle) has explored near them. Through the environment obstacle square boxes of various sizes are added randomly. The agent has no access to this ground truth map.

D. Automated Inventory

Industrial warehouses are dynamic environments, where different assets are changing continuously over time. Automating warehouse management, such as inventory analysis, enables faster operations and fewer errors. There are two main challenges: having a good inventory detector, and avoiding counting the same inventory item multiple times.

There are several commercial solutions for automated inventory which are based on having unique identifiers per detected object. In other cases, there is a predefined route where objects are never re-visited relying on a 2D Multi-Object Tracking (MOT) [24], [25] that gives unique identifiers between frames. The fusion of LiDAR and camera data for 3D detection [26], [27] and tracking [28] normally relies on annotations for both image and LiDAR data. In contrast, the proposed approach only requires image annotation, which makes it also feasible to incorporate existing datasets.

IV. Proposed Method

A. Environment Representation

In order to efficiently explore a previously unseen environment, the agent needs a representation of this environment in order to sample an appropriate action. In prior work, depth camera's and RGB camera's have been utilized [16], [17], [18], [19] to carry out exploration tasks. However, these sensors are plagued by a large sim2real gap, and are often noisy and dependent on the environmental conditions (e.g., lighting). A LiDAR sensor is a commonly used environment invariant and less noisy sensor in navigation tasks. The output of a LiDAR is typically represented utilizing a 3D point cloud consisting of coordinates on which the laser encountered an obstacle.

Our method however, does not require a 3D representation as an egocentric 2D map-like representation is sufficient in order to perform 2D navigation. In order to obtain this 2D representation, the 3D point cloud is projected onto a 2D plane by flattening the height dimension. Points which are too close, or too far in distance are also ignored in order to mimic the minimum and maximum range of a real-world sensor. While this resembles a laser scan representation, by flattening a 3D representation possible voids in the laser scan due to local holes are avoided.

A second input channel containing the local past trajectory of the agent is used in order for the agent to avoid visiting the same place multiple times.

B. Directed Exploration

In order to direct the exploration behavior of the agent, the proposed architecture allows an additional flattened point cloud map as input. This point cloud should mark regions which have been classified by a separate task module as interesting, and require further exploration. An object detector could for example communicate points of which it has only a low certainty about its classification accuracy. Navigating the AGV close to these points could allow the object detector to improve its classification results through actively obtaining better viewpoints.

When the task module becomes certain of its prediction (e.g., the classification probability becomes higher than a predefined threshold) the points are removed from the directed exploration point cloud.

C. Policy Architecture

In the presented approach the agent has a visual pipeline which consists of an egocentric local map with obstacles, an egocentric local directed exploration map and an egocentric local map with the past trajectory of the agent. This visual pipeline is processed by three CNN layers.

In turn, this output is concatenated with a boolean variable which indicates if the safety scanner of the AGV is active in the current state. This safety scanner is an independent system which prevents physical damage due to collisions. This input can be utilized by the policy to maneuver away from obstacles invisible to the visual pipeline. The safety scanner is simulated by checking if the result position of an action would cause a collision before actually moving the simulated AGV.

This concatenated output is then utilized to output a distribution over the discrete set of actions. This distribution of actions can be utilized to stochastically sample actions, leaving room for exploration of novel strategies. The full architecture is displayed in Figure 2.

D. Action Specification

While an RL policy could directly output a continuous value for the steering angle θ and forward velocity V, this is generally regarded as a more demanding setting [29]. Action discretization has been proposed as a viable less challenging alternative [30]. We have chosen in our approach to discretize the possible actions into 15 discrete actions (small step forward $V = 0.3m/s$, reverse straight $V = 0.3m/s$, large step forward $V = 0.5m/s$, small $\delta = 0.17rad/s$, medium $\delta = 0.4rad/s$, large $\delta = 0.7rad/s$ turn left/right and small/medium/large reverse turn). These steering angles and velocities have been obtained from human demonstrations collected utilizing a real world AGV. Actions are executed at 0.5Hz in both the real world and the simulator.

E. Training

In order to train the agent, the PPO algorithm [9] is utilized. As the reward function the agent is provided with a small slack penalty of -0.01, which prevents the agent from *slacking off*. In order to study the problem of exploration, three different options for the second term of the reward function are studied:

- **Directed**: receive 0.5 reward when positioning the AGV near an area marked for exploration in the directed exploration point cloud input.
- **Floor coverage**: the environment is divided in a virtual grid with tiles of each $1m^2$. When visiting a new tile the agent receives a positive reward of 0.1.
- **Combined**: in this setting the agent receives both 0.1 for visiting a new tile in the virtual grid, and 0.5 for positioning near areas marked for exploration.

A collision penalty of 0.05 is deducted upon a collision with an obstacle. The different values utilized in the proposed reward functions are obtained through hyperparameter searches.

The agent should be able to function in an environment it did not see before. In order to achieve this behavior the simulated environment is procedurally generated during training. In order to provide the agent with enough instances of seen/unseen trajectories the environment is reset after a fixed amount of 500 steps. In each episode a variable amount of box obstacles are added in different sizes in order to extend the dynamic navigation capabilities of the agent. Through the introduction of these random obstacles, the agent will be able to handle cluttered real-world environments. The agent is spawned in a random starting position in each episode. A top-down view of an episode can be seen in Figure 2. This view is not available to the agent.

V. Warehouse Simulator

The simulator was build on top of the *MiniWorld* environment [31]. It was modified in order to support our customized egocentric point cloud based representation.

A. Layout

The goal of our simulator is to allow the agent to learn how to navigate in typical warehouse settings. We consider a typical warehouse setting in which multiple rows of racks are placed.

In each episode the environment is procedurally generated using horizontal racks, vertical racks or empty spaces. An example containing two sets of horizontal racks is displayed on the right of Figure 2.

B. Lidar Simulation

When using the real LiDAR, the local point cloud can be utilized directly. In the simulator, a ray casting approach is utilized in order to obtain the same presentation (as plotted in Figure 1).

The proposed environment representation is not only an efficient way of representing the environment in navigation tasks, it is also a representation which is computationally inexpensive to simulate. This is currently an essential property when choosing an RL-based approach. As RL-based navigation approaches often require large amounts of interactions with the environment [32].

Through utilizing 8 environments in parallel, an average of 250 actions can be executed and evaluated each second on a modest GPU-enabled system (Intel Core i7-9700, Nvidia GTX1060). The real-world AGV in contrast only operates at 0.5 actions per second (on purpose).

C. Simulated Vehicle Dynamics

While prior work in sim2real and RL has focused on differential drive, the introduced simulator supports Ackerman steering, which is more complex to simulate, but allows the method to be deployed on a wider range of platforms. In order to simulate the movement of the AGV in the simulator, a kinematic bicycle model is utilized, which allows to calculate the position of the AGV according to the following set of equations:

$$\begin{aligned} \dot{x}(t) &= V(t)\cos\theta(t) \\ \dot{y}(t) &= V(t)\sin\theta(t) \\ \dot{\theta}(t) &= \frac{V(t)\tan\delta(t)}{l - a\tan\delta(t)} \end{aligned} \tag{1}$$

In this equation V is the forward velocity, θ represents the yaw angle and δ the steering angle. The vehicle specific l is the distance between front and rear wheels, and a is the lateral distance of the front wheels in respect to the longitudinal center-line. Prior research [33] has demonstrated that this type of model can be successfully utilized to produce consistent and feasible trajectories given a limited lateral acceleration. Through embodied trial and error the agent is capable of learning navigational affordances (e.g. can I steer into this narrow corridor from this position).

VI. Case Study: Warehouse Inventory Task Module

In the setting of autonomous warehouse inventory counting [34] there are two main challenges: (1) having a capable

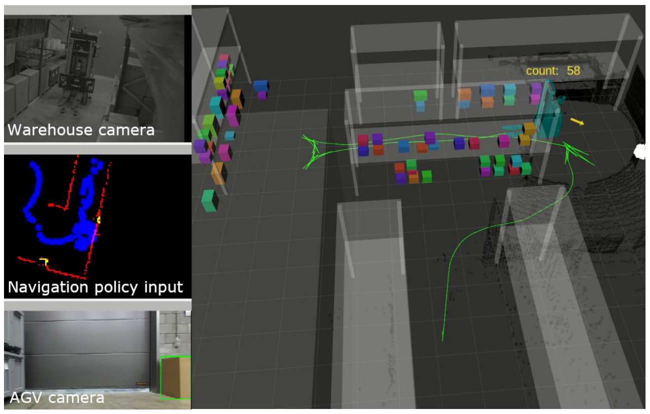

Fig. 3. A visualization of the real-world setup. The 3D map of the inventory is displayed on the right. In the current state the task module is uncertain about the object displayed in front (green bounding box). The navigation policy is instructed to explore the uncertain object by navigating towards it (yellow arrow). The agent does not have access to the warehouse camera and the 3D environment representation, those are included for visualization purposes.

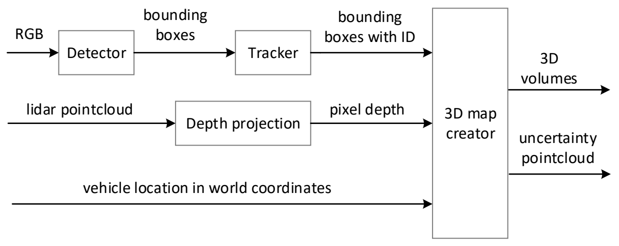

Fig. 4. Main inputs, outputs and building blocks of the perception module utilized in the warehouse inventory use case.

inventory detector, and (2) avoid counting inventory items multiple times.

Directed exploration helps to solve the first challenge, by providing the inventory detector with the capability of actively obtaining better observations through navigation, and thus reducing its uncertainty. The second challenge is addressed by constructing a 3D inventory map in world co-ordinates through combining past detections. The navigation policy will also autonomously steer the AGV to previously unvisited regions in the warehouse in order to allow the 3D inventory map to be fully constructed, and thus allow all boxes to become accounted for.

In Figure 3 the entire setup is displayed while executing the inventory task. The principal block is the 3D inventory map creator, which takes the vehicle position, bounding box detections and per pixel depth information (aligned with bounding boxes) and provides the detected objects 3D location and size, as well as a point cloud with the uncertain detection areas. The architecture of the perception module is plotted in Figure 4.

A. *Inventory Detection*

The detector uses only RGB camera data for classification purposes. LiDAR information is utilized to get and merge the 3D locations of inventory items. A fused approach where LiDAR is also used for detection would require 3D annotations as well. Because these kinds of industrial datasets are not publicly available, and labeling point clouds is an effort which makes the application infeasible in real cases, only RGB annotations were utilized for training the inventory detector. The tracker hands out consistent identifiers to the detections of the same objects in consecutive frames.

In order to run at real-time, the *YoloV7* object detector [35] was selected. Starting from a pre-trained version on the COCO dataset [36], 4 videos recorded in the test warehouse have also been annotated in order to fine-tune the object detector. Around 1500 frames were annotated. The detector is trained to learn only the ”box” class. When evaluated upon short recorded trajectories a precision of 89%, and a recall of 85% could be observed. This was however without any directed exploration, which should further improve the observed recall and precision of the model.

B. *Object Tracker*

BYTETrack [24] is utilized, which is a multi-object tracker based on spatial information. It implements a Kalman filter with a constant speed model for the bounding box detections position and size, and provides two loops where old tracks are matched with new detections, one for high confidence detections and a second for low confidence ones. Tracking provides unique identifiers across frames, but does not solve the problem of tracking objects when they re-enter the camera field of view. This is addressed in the 3D map creator.

C. *Object Positions*

Although a RGBD camera would already provide per pixel depth information, after some testing with several depth cameras (Intel Realsense D435, Stereolabs ZED Mini) it was found that the depth accuracy was not high enough, so a 3D LiDAR mounted next to the camera was used instead. The point cloud from the LiDAR is projected on the camera plane, with some radial inflation for closer points to have a richer depth image.

D. *3D Map Creator*

This module iterates over the bounding boxes and merges them with the previous map, taking into account the vehicle location. The map contains for each detected box the identifier, confidence, point cloud, centroid and cuboid size. It also differentiates between certain and uncertain detections. For each detected bounding box, there are two reasons to consider it as uncertain:

- Uncertainty in the detector output: If the confidence score provided by the detector is below a certain threshold, then the corresponding object is marked as uncertain.
- Uncertainty in the object location: Using the domain knowledge that boxes have flat surface, a sample consensus algorithm to fit a plane is computed. In case there are not enough inliers to the plane model, the plane is too far away, or the plane is not seen frontally, then the corresponding object is marked as uncertain.

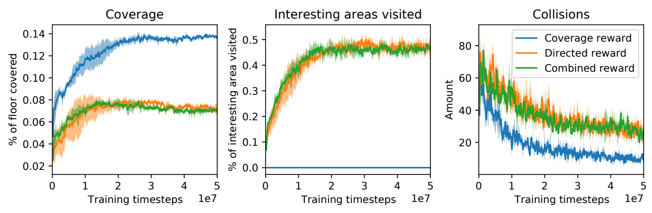

Fig. 5. Training performance in simulation. Utilizing a coverage-based reward does allow the agent to maximize its coverage, however it fails to navigate the agent to areas of interest. Utilizing a guiding reward signal allows navigating to the specific regions of interest. A combination of both reward signals has a slightly worse performance for both metrics.

In case of certain detections, they need to be merged to the map. There are two possibilities:

- The identifier of the current detection being merged into the map is already the map. In that case, an overlapping comparison is done with all the other detections already in the map, and if there is sufficient overlapping, they are merged. Otherwise, it is merged with the map object with the shared identifier.
- The current detection is not in the map. The same overlapping test is done as in the case above. If there is not enough overlapping it is a new detection and a new object is initialized in the map. Otherwise, if the overlapping is above the minimum threshold, the new detection is merged into the matched object in the map.

VII. Empirical Evaluation in Simulation

A. Training in Simulation

When solely optimizing for coverage, the agent chooses to stay far away from obstacles, and avoids going into narrow corridors. Going into narrow corridors has a higher chance of getting stuck, or colliding. The agent has learned to avoid these situations, because they lead to a negative reward.

In contrast, when specifically training utilizing a reward signal encouraging directed exploration, the overall floor coverage of the agent is reduced, but it manages to reach a significant amount of the specified areas of interest within the allocated 500 timesteps. The combination of both reward signals performs best in terms of directional exploration (as measured by the visited marked areas). Figure 5 shows the performance during training utilizing the different proposed reward signals.

B. Evaluation in Simulation

In order to evaluate the performance of the presented approach in simulation, results are collected on a set of 100 fixed warehouse configurations (floor plan, obstacles, starting position) in order to make sure that all runs are equally complex. Each episode is allowed to run for a maximum of 2000 steps, which should be enough for an optimal agent to fully complete the task.

When looking at the amount of obstacles (Table I) it is clear that the coverage (C) is heavily influenced by the navigational complexity of the environment. The percentage of visited areas (V) marked as interesting does however not remarkably decreases when making the environment more challenging.

In simulation the combined reward signal achieves better results overall in terms of directed exploration results. With no obstacles the agent is able to visit on average 91% of the areas marked as interesting. While the agent trained using only the directed exploration reward signal is able to visit 78% of the marked areas. The coverage part of the reward signal allows the agent to efficiently navigate in parts of the environment with no areas marked as interesting.

TABLE I
INFLUENCE OF OBSTACLES

# Obstacles	**Coverage**	**Directed**	**Combined**

0	C=0.46
	V=0
		C=0.14
		V=0.78
			C=0.18
			V=0.91

5	C=0.34
	V=0
		C=0.11
		V=0.63
			C=0.15
			V=0.78

10	C=0.27
	V=0
		C=0.09
		V=0.51
			C=0.14
			V=0.68

C. Robustness to Noise

During training, the navigation policies are trained with perfect sensors. As real-world sensors are often far from perfect, and are often subject to various kinds of noise, it is important that the navigation policy is able to withstand some amounts of noise added to the sensor observations.

In Table II, the results are plotted when only keeping a certain percentage of the points observed through the LiDAR. This essentially simulates how well the approach can work with a less expensive lower resolution LiDAR. Keeping 0% of the points essentially makes the agent blind.

TABLE II
POINTS SAMPLING (WITH 3 OBSTACLES)

	Coverage	**Directed**	**Combined**

100%	C=0.37
	V=0.0
		C=0.13
		V=0.71
			C=0.13
			V=0.72

50%	C=0.38
	V=0.0
		C=0.12
		V=0.68
			C=0.13
			V=0.73

10%	C=0.27
	V=0.0
		C=0.08
		V=0.50
			C=0.09
			V=0.51

1%	C=0.06
	V=0.0
		C=0.05
		V=0.2
			C=0.17
			V=0.17

0%	C=0.03
	V=0.0
		C=0.01
		V=0.02
			C=0.04
			V=0.05

While this blind agent is not able to cover or explore the environment, the trained agents are able to take a significant hit in terms of the amount of points coming from the LiDAR input.

Limiting the resolution is however only one source of a potential mismatch between the simulator and the real world. It is also possible that due to a non-perfect accuracy obstacles are detected which in fact are no obstacles, or it could be that obstacles are not detected. In order to simulate these possibilities, and to test the robustness of the trained agents, various amounts of *salt&pepper* noise are added to the input maps (Figure 7).

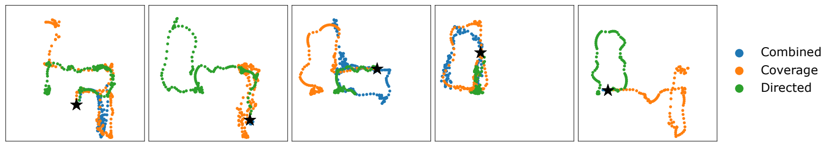

Fig. 6. Trajectories taken by the different policies during real-world evaluation. The points marked with stars are the fixed starting positions.

TABLE III
SALT&PEPPER NOISE (WITH 3 OBSTACLES)

	Coverage	Directed	Combined
p=0	C=0.37 V=0.00	C=0.13 V=0.71	C=0.13 V=0.72
p=0.001	C=0.34 V=0.00	C=0.16 V=0.76	C=0.13 V=0.73
p=0.005	C=0.37 vis=0.00	C=0.16 V=0.76	C=0.14 V=0.77
p=0.01	C=0.35 V=0.00	C=0.17 V=0.81	C=0.14 V=0.71
p=0.1	C=0.21 V=0.00	C=0.16 V=0.56	C=0.15 V=0.61
p=0.5	C=0.04 V=0.00	C=0.07 V=0.16	C=0.10 V=0.19

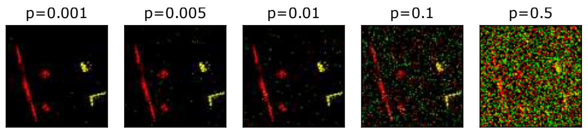

Fig. 7. Examples of the various levels of Salt&Pepper noise.

From Table III it can be concluded that while degrading the accuracy of the observations has a significant impact on the performance, the agent is still able to carry out its task with noisy sensors.

D. *Ablation study: Trajectory Information*

The requirement of adding the past trajectory as input to the agent is an expensive one in the real-world setup, as accurate indoor localization is a challenging problem, often requiring expensive hardware setups.

TABLE IV
INFLUENCE OF TRAJECTORY INFORMATION (WITH 3 OBSTACLES)

	Coverage	Directed	Combined
Enabled	C=0.37 V=0.00	C=0.13 V=0.71	C=0.13 V=0.72
Disabled	C=0.22 V=0.00	C=0.12 V=0.69	C=0.10 V=0.63

As seen in Table IV, removing the past trajectory does have a big impact on the performance of all three studied agents both in terms of coverage, and in terms of interesting areas visited.

VIII. REAL-WORLD EVALUATION

A. *Setup*

An open experimental platform has been built on top of a standard Still forklift (Figure 1). Localization is provided by a commercial system with reflector landmarks with known positions across the warehouse. An Ouster OS1 LiDAR with 64 vertical layers has been used. It has a vertical field of view of 45° and a maximum range of 120m. A Zed mini camera is used for the inventory detection. The layout and contents of the warehouse has been kept consistent throughout evaluation. The AGV was started consistently in different positions throughout the evaluation.

B. *Real-world Performance*

In order to evaluate the performance of the approach in a real-world warehouse, 5 positions were selected to run the different trained policies for as long as they made noticeable progress. The average results of these 15 runs are presented in Table V. The different trajectories taken by the AGV in each scenario are plotted in Figure 6.

TABLE V
REAL WORLD EVALUATION

	Coverage	Directed	Combined
Average steps taken	160.4	142.6	106.6
Average coverage	$44m^2$	$33m^2$	$21m^2$
Average reduced uncertainty	26.67	19.67	18.33
Rel. reduced uncertainty	0.61	0.60	0.87

In simulation the directed exploration capabilities were evaluated by recording if the AGV would position itself close to the objects marked as interesting. In the real-world setting the directed exploration capabilities are evaluated through recording the reduction in uncertainty of the task module. Concretely, in each run we track how many objects were marked as certain after they were first marked as uncertain (due to the prediction probability being lower then the specified threshold).

While this is only a small scale evaluation done in a highly realistic but complex environment filled with obstacles and small passages, some conclusions can be drawn. For example, unsurprisingly, the coverage agent is on average able to cover the largest area of the warehouse. However, by covering a larger area of the warehouse, this agent is also able to achieve the highest absolute uncertainty reduction scores.

When looking at agents which also take the uncertainty of the task module into account (directed, combined) we see that they take radically different trajectories (Figure 6). We noticed that these agents often propose more difficult paths, which led to lower floor coverage scores. If we however take efficiency into account the combined agent is able to reduce the most uncertainty per square meter coverage.

IX. Conclusion

In this paper, we demonstrated how deep RL can be utilized to perform real-world directed exploration of an unseen environment, relying on egocentric observations coming from a LiDAR sensor, and through training in simulation.

The proposed framework can be utilized in a wide range of different tasks without having to retrain the exploration policy. This was made possible due to the abstracted interface introduced between a task specific module and the navigation policy.

As an example the task of inventory management is studied. Within this study a supervised trained model was utilized to detect pieces of the inventory. The uncertainty of this model was utilized to steer the exploration behavior of the agent. The novel method is empirically evaluated both in simulation and through utilizing a real-world AGV.

In future work exploration of how to achieve similar results from cheaper, and more broadly available sensors such as depth cameras would further increase the applicability of the proposed approach.

Acknowledgment

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificile Intelligentie (AI) Vlaanderen" programme and EuROBIN (European ROBotics and AI Network).

References

[1] F. Gul, W. Rahiman, and S. S. N. Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, no. 1, p. 1632046, 2019.

[2] A. J. Moshayedi, G. Xu, L. Liao, and A. Kolahdooz, "Gentle survey on mir industrial service robots: Review & design," *J. Mod. Process. Manuf. Prod*, vol. 10, no. 1, pp. 31–50, 2021.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[5] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[6] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," in *RL4RealLife workshop*, 2019.

[7] S. Thrun, "Efficient exploration in reinforcement learning.," Tech. Rep. CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA, 1992.

[8] T. Hanke, A. Schaermann, M. Geiger, K. Weiler, N. Hirsenkorn, A. Rauch, S.-A. Schneider, and E. Biebl, "Generation and validation of virtual point cloud data for automated driving systems," in *2017 IEEE ITSC*, pp. 1–6, 2017.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[10] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," in *ICML15*, 2015.

[11] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[12] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 3310–3317 vol.4, 1994.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.*, pp. 146–151, 1997.

[15] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 351–356, 2011.

[16] T. Chen, S. Gupta, and A. Gupta, "Learning Exploration Policies for Navigation," in *ICLR19*, 2019.

[17] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to Explore using Active Neural SLAM," in *ICLR20*, 2020.

[18] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-Scale Study of Curiosity-Driven Learning," in *ICLR18*.

[19] D. S. Chaplot, M. Dalal, S. Gupta, J. Malik, and R. Salakhutdinov, "SEAL: Self-supervised Embodied Active Learning using Exploration and 3D Consistency," in *NeurIPS21*, 2021.

[20] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Are We Making Real Progress in Simulated Environments? Measuring the Sim2Real Gap in Embodied Visual Navigation," *IEEE RA-L*, vol. 5, 2020.

[21] J. Truong, M. Rudolph, N. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation," 2022.

[22] M. Rosano, A. Furnari, L. Gulino, and G. M. Farinella, "On Embodied Visual Navigation in Real Environments Through Habitat," in *ICPR*, 2020.

[23] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IROS17*, pp. 31–36, 2017.

[24] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," *arXiv preprint arXiv:2110.06864*, 2021.

[25] J. Cao, X. Weng, R. Khirodkar, J. Pang, and K. Kitani, "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking," *arXiv e-prints*, p. arXiv:2203.14360, 2022.

[26] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 244–253, 2018.

[27] A. Mahmoud and S. L. Waslander, "Sequential fusion via bounding box and motion pointpainting for 3d objection detection," in *2021 18th Conference on Robots and Vision (CRV)*, pp. 9–16, 2021.

[28] H. Zhong, H. Wang, Z. Wu, C. Zhang, Y. Zheng, and T. Tang, "A survey of lidar and camera fusion enhancement," *Procedia Computer Science*, vol. 183, pp. 579–588, 2021. Proceedings of the 10th International Conference of Information and Communication Technology.

[29] H. V. Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement learning*, pp. 207–251, Springer, 2012.

[30] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10688–10694, 2020.

[31] M. Chevalier-Boisvert, "Miniworld: Minimalistic 3d environment for rl & robotics research." `https://github.com/maximecb/gym-miniworld`, 2018.

[32] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *ICRL20*, 2020.

[33] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *IEEE IV 2017*, pp. 812–818, 2017.

[34] F. G. Guinjoan, E. Rademakers, A. B. Temsamani, G. Radevski, T. Tuytelaars, M. Hutsebaut-Buysse, K. Mets, T. De Schepper, E. Mannens, S. Latré, and H. Van Hamme, "A multi-modal ai approach for agvs: A case study on warehouse automated inventory," *ICAS 2023*, vol. 15, p. 34, 2023.

[35] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV14*, pp. 740–755, Springer, 2014.