A Multi-modal AI Approach For AGVs: A Case Study On Warehouse Automated Inventory

Ferran Gebellí Guinjoan[1], Erwin Rademakers, Abdellatif Bey Temsamani
Flanders Make
Lommel, Belgium
[1] email: ferran.gebelli@flandersmake.be
Gorjan Radevski[3], Tinne Tuytelaars
KU Leuven, ESAT
Leuven, Belgium
[3] email: gorjan.radevski@esat.kuleuven.be

Matthias Hutsebaut-Buysse[2], Kevin Mets, Tom De Schepper, Steven Latré, Erik Mannens
University Of Antwerp - imec
Antwerpen, Belgium
[2] email: matthias.hutsebaut-buysse@uantwerpen.be
Hugo Van hamme[4]
KU Leuven, ESAT
Leuven, Belgium
[4] email: hugo.vanhamme@esat.kuleuven.be

***Abstract*— We present a multi-modal AI approach for Automated Guided Vehicles (AGVs) to perform autonomous warehouse inventory monitoring. A vision module detects and tracks the goods and registers them to the inventory when the confidence score is high. Moreover, we use uncertain detections to direct the AGV to better viewpoints that could lead to new inventory counts. Navigation is done with a Reinforcement Learning (RL) agent trained to perform directed exploration in previously unseen warehouse settings. Because there is not a pre-defined route, we implement a robust way to merge detected items to avoid double counts. We also use speech as an easy way to provide instructions to the AGV.**

Keywords- AI based autonomous systems; Automated Guided Vehicles, multimodal AI; navigation; deep learning; neural networks; reinforcement learning; warehouse monitoring & management system; automated logistics & inventory

I. INTRODUCTION

Inventory is done manually by humans in many logistic and industrial warehouses, which is a slow and costly procedure. Automated inventory analysis is potentially faster, safer and more accurate [1]. To perform automated inventory monitoring, two main tasks need to be achieved: count the goods and move autonomously through the warehouse.

Regarding the counting of goods, supervised AI vision techniques have proven good performance for detection of many types of objects [2]. However, accuracy depends on the amount and quality of training data, and often it is hard to reach very high accuracy values. For the inventory application, a requirement is to have as few labeled examples as possible. Otherwise, the implementation in a real setting would be infeasible if a lot of manual labeling is necessary each time a new product is introduced.

To move autonomously through the warehouse, many industry settings use Automated Guided Vehicles (AGVs). Classic navigation pipelines typically need to construct a map by scanning the environment with sensors, such as lidars [3], while manually driving the AGV. Sometimes the usage of floor markings or fiducial landmarks (e.g., reflectors) are used as well. These approaches do not only require an updated map, but also require a different module to set destinations or missions with waypoints, meaning that a high set-up time for new or modified environments is needed. Because of the increasing variability in warehouses, it is common that rack configurations are modified after short periods of time. This exposes the need for an increased flexibility in the whole navigation approach.

In this work, we show how multi-modal AI is leveraged to overcome current limitations and perform an inventory in a pure exploration manner without previously knowing the rack configuration. We demonstrate a proof of concept in a real-time AGV forklift (Figure 1), which exploits spatial-temporal data to increase accuracy and robustness.

Figure 1. AGV used for integration and experiments

Regarding the detection of goods, by using the video stream, lidar data and the forklift position we avoid the need to detect all objects in all frames, and detecting at least once each object becomes sufficient. Moreover, it is important to reject false positive counts to avoid adding wrong detections to the inventory. We deal with this problem by setting a high confidence threshold, which increases the object count only when the model is certain about the existence of the object. However, the information in low confidence detections is still valuable, and we use it to direct the navigation module to uncertain areas. This allows to actively interact with the

 ISBN: 978-1-68558-053-7

environment and get better viewpoints. In many cases, this will lead to high confidence detections, and as a consequence to new counts in the inventory.

In terms of robustness, the challenge is to avoid counting twice the same object, so we add tracking between frames. Furthermore, as exploration can lead to revisiting some places after a while, we implement a robust merging strategy to avoid counting again objects seen in the far past. This is an issue that existing navigation solutions do not face because they navigate on a mission with manually selected waypoints on a map.

To explore an unseen environment, we use a Reinforcement Learning (RL) agent, which is trained through trial-and-error in simulation to explore a warehouse while being directed by an external module.

Operators need a fast and easy way to interact with AGVs. We introduce a module that allows operators to give instructions via speech commands. This way, the cooperation between humans and machines becomes smoother

In previous work [4], we focused on the multi-modality between speech and vision to detect objects given speech cues. The RL navigation modality was also present, but only available in simulation as a digital twin which was synchronized with the real-world vehicle. In this work, we have a higher focus on the vision modality and how it interacts with the RL navigation. Furthermore, we have implemented all the modules in a real platform, and we have switched from an autonomous off-highway vehicle into an AGV which performs warehouse inventory.

In Section II, the related work is reviewed. Section III specifies the multi-modal AI solution used to solve the automated warehouse inventory use case. Section IV describes the experimental platform and the results on the real setup. Finally, Section V gives conclusions and talks about future work.

II. Related Work

A. Object detection and tracking

There are two main approaches to perform Multi-Object Tracking (MOT), "tracking by detection" and "detection by tracking". "Tracking by detection" is a two-stage process: first an object detector [5] [6] [7] recognizes the objects of interest in a new frame, and then an object tracker uses the past detections to associate the new detections with the previous, creating tracks with consistent IDs across frames. CenterNet [8] is used in many trackers [9] [10] [11], as it is simple and efficient. The YOLO family of detectors [12] [13] are used in various trackers [14] [15] [16] due to the good trade-off between speed and accuracy. MOT tracking in 3D [17] [18] [19] follows the "tracking by detection" approach. One of the advantages of "tracking by detection" is that there are two modules, which can be easily combined, e.g., the same tracker can be fused with different detectors. In "detection by tracking", the tracking information is also used to improve the accuracy of the detector. In some cases [20] [21], a Kalman filter [22] is used to predict the tracks in the next frame. Transformer-based detectors can also be used in tracking [23] [24] to propagate boxes between frames.

Most methods keep only high confidence score detection boxes to perform data association, as detections with low confidences can be unrelated background objects. BYTETrack [16] implements a strategy with a second association round for low score detections, as an additional step, to filter potentially occluded objects. Generally, all methods use a similarity metric followed by a matching strategy for data association. SORT [25] uses location and motion cues to compute the Intersection over Union (IoU). Some methods [24] [26] [8] are robustified w.r.t. the camera movement by having motion specific parameters. Appearance similarity of Re-ID features is used in a standalone way in DeepSORT [27] and in a joint way with detections in other methods [15] [28].

Few works focus on re-tracking objects once they are lost and allow for re-tracking [29] only in a short future horizon.

B. Navigation

Currently, navigation with an AGV throughout a warehouse mostly relies on SLAM-based approaches in which the agent constructs a map, and simultaneously tries to localize itself within this map. Alternatively, beacons, floor markings or guiding rails might be placed in the environment in order to allow the AGV to navigate efficiently [30].

These approaches, however, have two main drawbacks. The first is that they are often not very flexible and require a lot of manual tuning and setup in order to operate. The second drawback is that navigation tasks are currently mostly limited to positioning the AGV to a specific point in space, through providing the coordinates of the location. Semantic tasks such as directly searching for a specific item, without knowing its location would require visiting the entire environment in a greedy fashion.

Reinforcement Learning (RL) is an alternative approach which is able to learn control policies in sequential decision-making processes through trial-and-error learning. In an RL setting, the agent executes actions in order to figure out the effect they have on the environment. An RL agent learns about the quality of its performed actions through feedback from a reward function.

RL has been applied successfully in navigation settings, e.g., in operating stratospheric balloons [31], or controlling robotic platforms [32]. However, RL is still facing an issue of sample efficiency. An RL-based approach generally requires a large amount of interactions with its environment in order to learn a successful policy. While there is a lot of research being conducted on making RL more sample efficient in navigation settings [33] it is still very impractical and unsafe to train an RL agent in the real-world [34] [35]. For this reason, RL is usually trained in simulation, where multiple environment instances can be parallelly run at a faster rate.

While training in simulation can generally alleviate the problem of sample inefficiency, a new issue arises due to the fact that the observations coming from a simulated sensor, and those coming from a real-world sensor are often very different. Real-world sensors are often noisier and are utilized in a larger variety of setups (such as weather conditions or lighting). A few different methods have been proposed before in order to solve this sim2real gap. Attempts have been made

 ISBN: 978-1-68558-053-7

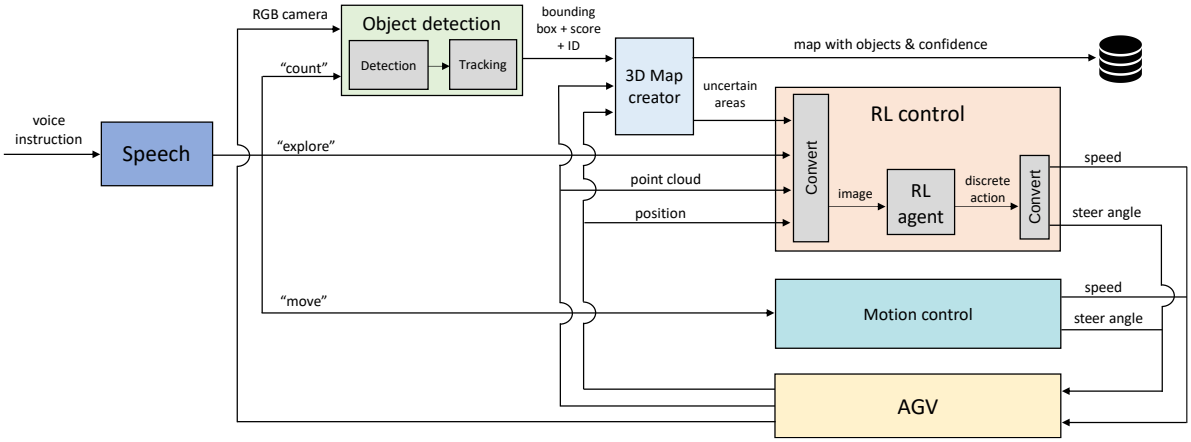

Figure 2. Multi-modal software architecture

at making the simulators more realistic [36]. This however resulted in the agent overfitting to aspects of the simulator [37] [38].

An additional benefit of RL based approaches over classic navigation approaches, is that they are able to solve semantically defined tasks. As classical navigation approaches are typically limited to navigation to coordinates. RL-based navigation approaches have outperformed such classic agents in specific settings in coordination-based navigation [39], and have been able to also handle tasks such as finding objects through only specifying the type of object, or exploring the environment in an intelligent and an efficient manner [40].

The usage of RL navigation to improve the confidence of detections in a map has also been addressed [41], but only in simulation.

III. Case Study: AGV for Warehouse Inventory

A. *Multi-Modal AI architecture*

Figure 2 an overview of the software architecture and interface between the platform and the different AI modules. Apart from the data that comes from the AGV (RGB image, point cloud and position) a voice instruction is the input that serves as user interface. The overall output are the control commands to the AGV and the map with the inventory counts.

B. *Spoken Language Understanding*

If the operator wants to give a speech command, he/she can either press a button and then start talking, or enable the open microphone feature and say a pre-defined keyword to indicate that an instruction will be given.

We have reused the user interface and AI model from a previous work [4] and retrained it to work in English for a new set of tasks. First, a command is available to start a new inventory session ("count" Figure 2). Then, there are 3 options available: steer the AGV manually, trigger the RL autonomous exploration ("explore" in Figure 2), or further give speech instructions to control the movement of the AGV ("move" in Figure 2), such as "forward", "a little bit to the left", or "stop".

C. *Reinforcement Learning Navigation*

We address the sim-2-real gap in the sensing part by using lidars, which are more robust to sensor noise. While lidar-based simulations are often very compute-intensive, our approach allows fast simulations by rendering obstacles into top-down images containing the lidar data, without any need for ray casting. Rack locations are similarly added as a second image channel, and a third channel contains past vehicle positions. This 3-channel image in the ego view (see Figure 3) determines the only input of the RL agent. The same 3-channel image is created in the real setup:

- The obstacles channel comes from a projection of the 3D lidar point cloud to the plane parallel to the floor.
- The second channel contains the areas to direct the exploration, which come from the detection module. A 3D point cloud is projected as in the first channel.
- The third channel contains the past trajectory, which is obtained by concatenating the last positions given by the AGV positioning system.

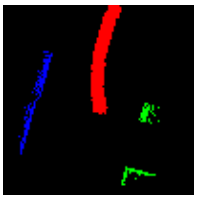

Figure 3. Input image to the RL agent. Blue are obstacles, green represents uncertain areas and red is the past trajectory

Simulations use a kinematic model of the AGV to bridge the sim-2-real gap in the acting part. The RL policy utilizes a discrete set of 15 actions, that map to specific steering angles and forward speeds. At a low speed (0.3 m/s) the vehicle can turn at 3 different angles (small, medium and large) to the left, and 3 to the right. The vehicle can also go straight. This makes a total of 7 actions, which are also available for backward moving. The 15th action allows to go forward straight at a higher speed (0.5 m/s). The simulation environments are randomly generated to create several rack configurations and generalize to any warehouse setting. We use Proximal Policy Optimization (PPO) [42] to train the agent.

 ISBN: 978-1-68558-053-7

D. Object Detection and tracking

The detector uses an RGB image as input and produces bounding boxes with associated confidence scores. We do not use depth sensors or lidar. The reason is that training models which use these sensors would require 3D annotations, generally not available in industrial datasets. An alternative is to label point clouds, which is prohibitive, and therefore we opt only for 2D object detection applied on RGB images.

We use the 3D lidar sensor, available in the navigation module, to obtain depth information which is pixel-by-pixel aligned with the RGB images. This approach provides better depth accuracy than depth cameras. The point cloud from the lidar is projected on the camera plane, with some inflation proportional to the depth value, leading to higher inflation for closer points. This provides a richer depth image, as illustrated in Figure 4. The projection of a point cloud into a camera plane only works well only if the two sensors are mounted close enough, which is the case for our platform.

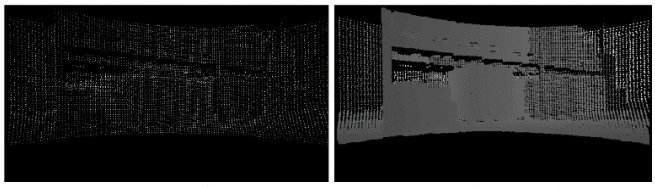

Figure 4. Depth image from the point cloud without inflation (left) and with inflation (right)

We choose the Yolov7 detector [12], as it is one of the latest open-source detectors with a better trade-off between accuracy and real time performance. Starting from a pre-trained version on COCO dataset [43], 4 videos recorded in the test warehouse have been annotated, making a total of around 1500 frames. The detector is trained to detect only one class, which is the cardboard box.

We select BYTETrack [16] as an object tracker, because it can be easily coupled with any other detector and yields to good accuracy in the MOT20 [44] benchmark. The main building block is a Kalman filter [22] with a constant speed model for the bounding box position and size of the detections. In most cases, trackers are employed in applications with a static camera and moving objects, while we use a moving camera with static objects. We have slightly modified the default version to be able to tune the covariance matrices Q and R of the Kalman filter in order to put a higher confidence on the detections (measurements) than in the model (constant speed motion). Especially when the camera is turning, the model will be less reliable, so we want to give higher importance to the new detections. Tracking provides unique IDs across frames, but does not solve the problem of tracking objects when they re-enter the camera FOV after some time. This will be addressed in the 3D map creator.

E. 3D map creator

The individual 2D detections, the generated depth image and the AGV location in the warehouse are inputs to the 3D map creator, which is responsible to merge new detections to the ones in the map. This way, it keeps an updated version of the counted items locations, which are represented as cuboids with an ID, confidence score, internal point cloud, center, width, height and depth. The 3D map also keeps track of the uncertain areas, which are represented in the same way but with a negative value for the ID. Algorithm 1 shows the pseudo-code of the map creator, including also the object detector and tracker.

Algorithm 1: Pseudo-code of the inventory monitoring

Input: sequence S with image I, lidar point cloud L and vehicle position P; threshold for tracking T_t; detection confidence threshold for counting T_d; position confidence threshold for counting T_p
Output: goods map M (list of objects with ID, confidence score, point cloud and 3D cuboid)

```
Initialization: M ← 0
for I, L, P in S do
    Dets = detector(I)
    Tracks = tracker(Dets, T_t) % Tracks contain an ID, confidence score and bounding box
    Depth = project_pointcloud(L)
    for Track in Tracks do
        Depth_filtered = filter_depth(Depth, Track) % Depth with padding
        O_track = to_pointcloud_object(Depth_filtered, T_d) % object with point cloud, ID (<0 for uncertain) and confidence fields
        O_filtered = filter_poitcloud(O_track, T_p) % SOR, passthrough, SAC filters + ID becomes <0 if uncertain position
        O_world = to_world(O_filtered, P) % transform from ego view
        O_current = compute_cuboid(O_world) % add 3D box to object
        Test = overlap_test(O_current, M) % compare to all map objects
        if Test then
            M = merge_to_map(O_current, M) % discard new ID & merge
        else
            M = add_to_map(O_current, M) % new detection added to map
        end
        M = voxel_grid_filter(M)
        M = delete_uncertain_areas(M)
    end
end
Return M
```

Figure 5. Algorithm for inventory monitoring

For each new frame the algorithm iterates over the bounding boxes from the tracker. For each track, the corresponding depth pixel values are retrieved with a padding to discard pixels that may belong to the background. Then, depth values are converted back to a point cloud per detection. This point cloud goes through a filtering process that includes a Statistical Outlier Removal (SOR), a passthrough filter to remove far points and a SAmple Consensus (SAC) test: using the domain knowledge that boxes have flat surfaces and that they are never seen from above, we fit a plane and require it to be vertical in the world coordinate system.

At this point, we have for each detected object a point cloud, which generally contains points on the main surface of the box. There are two reasons to consider it 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 in detection.

- Uncertainty in the object location: In case the SAC plane is too far away, has a low number of inliers, or is not seen frontally (the boxes are too much at the side of the image), then the corresponding object is marked as uncertain in position.

The point cloud is finally transformed using the vehicle location into world coordinates, and a 3D cuboid that encloses the point cloud is computed.

Then, all the detections are merged with the map. There are two possibilities:

- The ID of the current detection is already in the map. In that case, the default option is to merge it with the map's object with the same ID. However, it could be the case that the 2D tracker fails, so an overlapping volume comparison is done with all the other detections already in the map, and if there is enough overlapping, the current detection is merged with the map object with more overlapping volume.
- 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, the new detection is merged into the matched object in the map.

When a detection is merged to one in the map, the point clouds are concatenated and then reduced using a voxel grid filter. The confidence is updated to the maximum of the ones being merged, and the centroid and vertex locations are updated fitting a cuboid to the point cloud. Since only one surface per box is considered, the cuboid corners are extended so each dimension is bigger than a user defined minimum object size. The current vehicle position and relative viewpoint respect the detection are used to know the direction of the extension.

Uncertain detections are merged in a similar way as certain ones. Certain and uncertain detections are never merged between them. When an uncertain detection with a particular ID becomes certain, all the uncertain data is deleted. Moreover, whenever there is a certain detection being added or merged to the map, nearby uncertain detections are deleted. Finally, in case that the AGV gets close enough to an uncertain detection and it remains uncertain, the object is completely discarded, since after having a good viewpoint the certainty did not increase enough, so it is assumed to be a detection false positive.

IV. EVALUATION

A. *AGV forklift platform*

An open experimental platform has been built on top of an AGV, automating a standard pallet forklift [45]. Localization is provided by a commercial system with reflector landmarks with known positions across the warehouse. Triangulation allows to get the AGV position with an accuracy of the order of few centimeters.

Two Ouster OS1 lidar with 64 vertical layers have been used. They have a vertical field of view of 45° and a maximum range of 120 m. They are placed in the front and the back of the AGV, and they are merged into a single point cloud that has a full 360-degree coverage. A camera (Zed mini) is used for inventory detection and is placed at the front of the forklift.

Figure 6 illustrates the hardware architecture and principal communications in the AGV. Robot Operating System (ROS) is used as a middleware to provide communication between the different perception modules. Then, control commands are sent to a motion module via ethernet, which is responsible for executing the actions on the AGV. There is a safety system mainly based on safety scanners that stops the forklift in case of an expected imminent collision.

The dynamics of the forklift can be summarized in the kinematic bicycle model [46]. This model is used in the RL training bridge the sim2real gap in the actuation. In Figure 7 the vehicle model can be seen.

The kinematics for a forklift AGV are defined by the following equations [47]:

$$\dot{x} = V(t)\ cos\ \theta(t)$$
$$\dot{y} = V(t)\ sin\ \theta(t)$$
$$\dot{\theta} = \frac{V(t)\ tan\ \delta(t)}{l - a\ tan\ \delta(t)}$$

The following values apply for this work AGV: l = 1.5m, a = 0:15m The forward velocity is denoted as V and δ is the steering angle in radians.

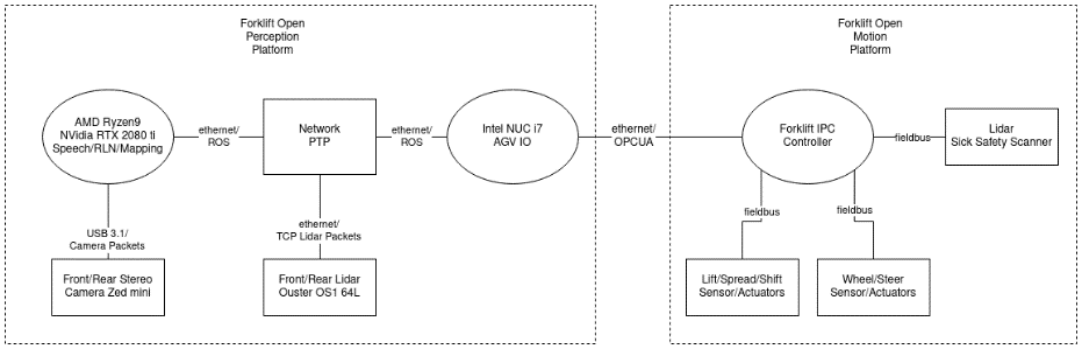

Figure 6. Hardware architecture

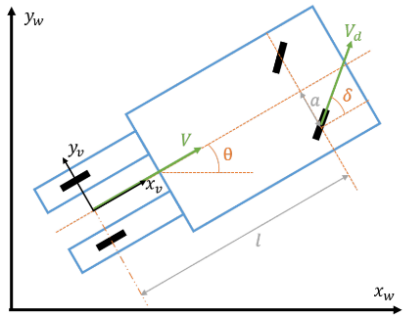

Figure 7. Bicycle kinematic model for AGV

B. *Experimental Results*

We have integrated all the algorithms in the forklift AGV platform and performed several online real-time experiments. Figure 8 shows the available inventory visualization in an experiment sequence. The locations of the racks are provided by the user and are only employed to improve the visualization, as they are not part of the algorithm. In the Figure 8 top image it is seen how several boxes in the middle rack have already been detected while in another rack there are uncertain detections. White points denote areas with low detection certainty, while grey points correspond to low certainty in location. Those areas direct the navigation to move closer, and once better viewpoints are obtained, they become certain detections that are added to the inventory, as seen in the middle image. Finally, in the bottom image it is seen how after performing a loop around the middle rack, the previous 2 racks are seen again, but only new objects are added to the inventory count. Detections that are assigned to an object already in the map are merged, and the object location is slightly adjusted accordingly if necessary.

TABLE 1 contains the results for object detection. We have used a test subset of 188 frames of around 30 seconds where the vehicle goes towards a rack and then performs a turn. The "*Detector alone"* row contains the results of the detector without any tracking or merging on the map. Then, the following rows represent the results for different ablations on the map creator, where the thresholds to track (T_t) and to count (T_d, T_p) are modified. *H* represents a version where the several thresholds for the position certainty are high, while *L* is for low values. We denote as $T_t=0$ the case where the 2D tracker is not used. The results include the precision and recall values, as well as the number of detected uncertain objects that are remaining in the map at the end of the sequence. A distinction is done between remaining uncertain objects that would become true and false positives if added to the count.

Although accuracy values in the *"Detector alone"* are high, all versions with the 3D map creator have a higher precision and similar or higher recall. Depending on the thresholds to track the objects and to count them in the inventory, the trade-off between precision and recall changes. In our application a high precision would be desired, while we expect to improve the recall by the active navigation. Results show there is still room for improvement in the directed exploration, since there are several true positive uncertain detections that were not yet included in the map. Alternatively, counting and position thresholds could be further reduced to count those uncertain detections and increase the recall, but that would reduce precision. Results show how the usage of a 2D tracker ($T_t \neq 0$) helps to avoid false positives, as seen in the TABLE 1 uncertain detections.

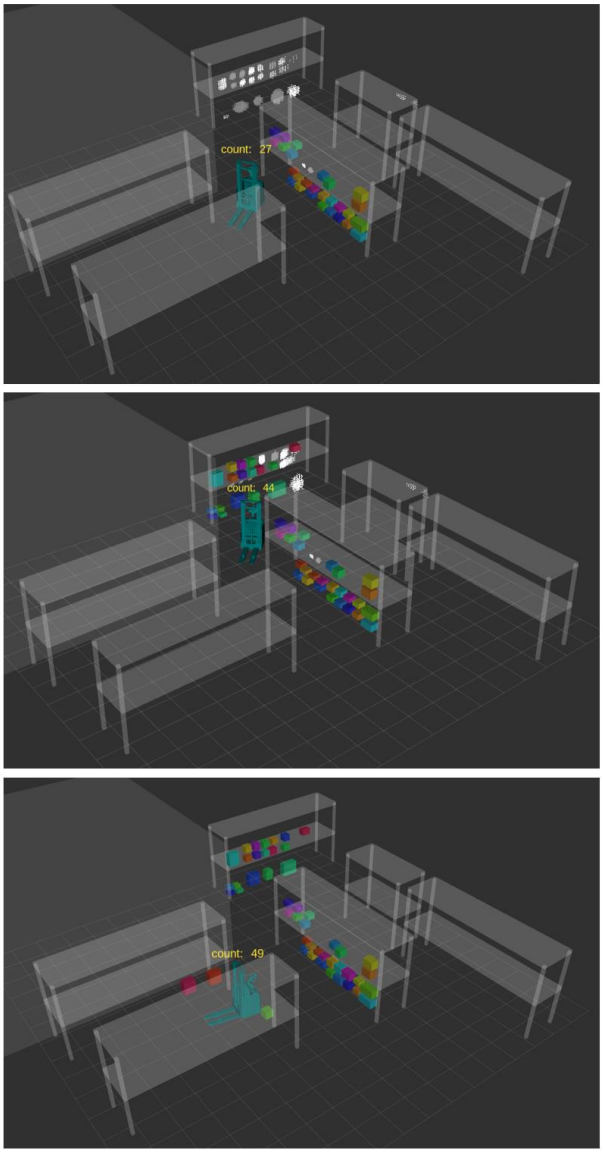

Figure 8. Sequence of the forklift around some racks in a warehouse.

TABLE 1. RESULTS OF THE OBJECT DETECTION

	Precision	**Recall**	**Uncertain (T/F)**
Detector alone	0.89	0.85	-
T_t**=0.3,** T_d**=0.9,** T_p**=***H*	1	0.76	9/0
T_t**=0.3,** T_d**=0.5,** T_p**=***H*	1	0.81	7/0
T_t**=0,** T_d**=0.5,** T_p**=***H*	1	0.81	7/7
T_t**=0.3,** T_d**=0.9,** T_p**=***L*	0.96	0.86	5/0
T_t**=0.3,** T_d**=0.5,** T_p**=***L*	0.97	0.89	3/0
T_t**=0,** T_d**=0.5,** T_p**=***L*	0.94	0.86	3/8

 ISBN: 978-1-68558-053-7

Results show how, by using spatial-temporal information of the same object while actively navigating to obtain better viewpoints, we can rely in a less accurate detector and achieve higher accuracy results on the high-level task of inventory count. This directly translates into a faster set up of the detector (less required labeled data, less time doing hyperparameter tuning, etc.), which is critical to reduce the implementation time of the solution in a new or modified warehouse. In this direction, the usage of an instance segmentation detector would have provided pixel level detections, which could be better matched to depth information leading to better position accuracy in the map. However, this would have increased the inference rate and the labeling effort. Our results show, how by post-processing the lidar data and registering to the inventory only detections with high position accuracy, a bounding box detector is enough instead of a more advanced pixel level instance segmentation detector.

V. CONCLUSIONS

The multi-modal approach presented in this work showed how uncertain detections in the vision module can be used in a navigation module, in our case based on RL, to improve the accuracy of the high-level task of warehouse inventory monitoring. If navigation can get better viewpoints directed by the detection module, the accuracy of the object detector is highly increased when applied to a time sequence. For a feasible industrial implementation, the set-up time in new environments should be minimal, and this work has proven how by considering uncertainty in the detections, a fast detector that is not accurate enough at frame level can achieve high accuracy at a task level.

The merging approach used to create the 3D map ensures that the same object is not counted twice. This is necessary in case of exploration which might re-visit the same areas. However, the current approach assumes that in each inventory session no objects are moved or removed. This is a limitation that needs to be addressed in future research.

This work also showed how the sim-2-real gap in RL can be addressed to successfully navigate in an unknown environment. Moreover, the navigation and the inventory modules are loosely coupled, which extends flexibility. The navigation module could be combined with other types of tasks where directed exploration is needed, such as finding an object or as intelligent patrolling. Beside the warehouse management application, the presented approach could also be used in other domains such as AGVs in agriculture or construction applications. We foresee future work on the RL navigation module with extended evaluation of the directed exploration navigation. We expect a performance comparison both in the simulation and real world setups between agents trained with different reward functions (e.g., general coverage vs directed exploration).

ACKNOWLEDGMENT

This research is done in the framework of Flanders AI Research Program (https://www.flandersairesearch.be/en) that is financed by EWI (Economie Wetenschap & Innovatie), and Flanders Make (https://www.flandersmake.be/en), the strategic research Centre for the Manufacturing Industry who owns the AGV infrastructure. The authors would like to thank everybody who contributed with any inputs to make this publication.

REFERENCES

[1] A. Gunasekaran, H. Marri, and F. Menci, "Improving the effectiveness of warehousing operations: a case study," *Industrial Management & Data Systems*, vol. 99, no. 8, pp. 328-339, 1999.

[2] J. Moon, S. Lim, H. Lee, S. Yu, and K.-B. Lee, "Counting System Based on Object Detection Using Deep Learning," *Remote Sensing*, vol. 14, no. MDPI, p. 3761, 2022.

[3] C. Stachniss, J. J. Leonard, and S. Thrun, "Simultaneous localization and mapping," *Springer Handbook of Robotic*, no. Springer, pp. 1153-1176, 2016.

[4] A. B. Temsamani et al., "A multimodal AI approach for intuitively instructable autonomous systems : a case study of an autonomous off-highway vehicle," *The Eighteenth International Conference on Autonomic and Autonomous Systems*, pp. 31-39, 2021.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in neural information processing systems*, vol. 28, p. 1137–1149, 2015.

[6] K. He, X. Zhang, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016.

[7] P. Sun, Y. Jiang, et al., "What makes for end-to-end object detection?," *Proceedings of the 38th International Conference on Machine*, pp. 9934-9944, 2021.

[8] X. Zhou, D. Wang, and P. Krahenbuhl, "Objects as Points," in *arXiv preprint arXiv:1904.07850*, 2019.

[9] P. Tokmakov, J. Li, W. Burgard, and A. Gaidon, "Learning to track with object permanence," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10860-10869, 2021.

[10] Q. Wang, Y. Zheng, P. Pan, and Y. Xu, "Multiple object tracking with correlation learning," *Multiple object tracking with correlation learning*, pp. 3876-3886, 2021.

[11] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13708-13715, 2020.

 ISBN: 978-1-68558-053-7

[12] 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," in *arXiv preprint arXiv:2207.02696*, 2022.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," in *arXiv preprint arXiv:2004.10934*, 2020.

[14] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, "Transmot: Spatial-temporal graph transformer for multiple object tracking," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4870-4880, 2021.

[15] C. Liang, Z. Zhang, X. Zhou, and B. a. H. W. Li, "One more check: making “fake background” be tracked again," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1546-1554, 2021.

[16] Y. Zhang et al., "ByteTrack: Multi-object Tracking by Associating Every Detection Box," *European Conference on Computer Vision*, pp. 1-20, 2022.

[17] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes," *International Conference on Robotics and Automation (ICRA)*, pp. 9552-9557, 2019.

[18] X. Weng, J. Wang, D. Held, and K. Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[19] H. Wu, W. Han, C. Wen, X. Li, and C. Wang, " 3d multi-object tracking in point clouds based on prediction confidence-guided data association," in *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[20] P. Chu, and H. Ling, "Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6172-6181, 2019.

[21] J. Zhu et al., "Online multi-object tracking with dual matching attention," *Proceedings of the European conference on computer vision (ECCV)*, pp. 366-382, 2018.

[22] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Fluids Eng*, vol. 82, pp. 35-45, 1960.

[23] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8844-8854, 2022.

[24] P. Sun et al., "Transtrack: Multiple-object tracking with transformer," in *arXiv preprint arXiv:2012.15460*, 2020.

[25] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *IEEE international conference on image processing (ICIP)*, pp. 3464-3468, 2016.

[26] J. Wu, J. Cao, L. Song, Y. Wang, M. Yang, and J. Yuan, "Track to detect and segment: An online multi-object tracker," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12352-12361, 2021.

[27] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *IEEE international conference on image processing (ICIP)*, pp. 3645-3649, 2017.

[28] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu, "Quasi-dense similarity learning for multiple object tracking," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 164-173, 2021.

[29] T. Feng, L. Jiao, H. Zhu, and L. Sun, "A Novel Object Re-Track Framework for 3D Point Clouds," *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 3118-3126, 2020.

[30] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *Journal of Manufacturing Systems*, vol. 54, no. Elsevier, pp. 152--173, 2020.

[31] M. G. Bellemare et al., "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. Nature Publishing Group, pp. 77-82, 2020.

[32] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A General Navigation Model to Drive Any Robot," in *arXiv preprint arXiv:2210.03370*, 2022.

[33] E. Wijmans, I. Essa, and D. Batra, "How to Train PointGoal Navigation Agents on a (Sample and Compute) Budget," *21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1762-1764, 2022.

[34] H. Zhu et al., "The ingredients of real-world robotic reinforcement learning," in *arXiv preprint arXiv:2004.12570*, 2020.

[35] G. Dulac-Arnold et al. , "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. Springer, pp. 2419-2468, 2021.

 ISBN: 978-1-68558-053-7

[36] M. Savva et al., "Habitat: A platform for embodied ai research," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339-9347, 2019.

[37] A. Kadian et al., "Sim2real predictivity: Does evaluation in simulation predict real-world performance?," *IEEE Robotics and Automation Letters*, vol. 5, no. IEE, pp. 6670-6677, 2020.

[38] J. Truong et al., "Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation," in *arXiv preprint arXiv:2207.10821*, 2022.

[39] D. Mishkin, A. Dosovitskiy, and V. Koltun, "Benchmarking classic and learned navigation in complex 3d environments," in *arXiv preprint arXiv:1901.10915*, 2019.

[40] T. Chen, S. Gupta, and A. Gupta, "Learning Exploration Policies for Navigation," in *International Conference on Learning Representations*, 2018.

[41] D. S. Chaplot, M. Dalal, S. Gupta, J. Malik, and R. R. Salakhutdinov, "SEAL: Self-supervised embodied active learning using exploration and 3d consistency," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13086-13098, 2021.

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

[43] T.-Y. Lin et al., "Microsoft COCO: Commeon objects in Context," *13th European Conference in Computer Vision*, pp. 740-755, 2014.

[44] P. Dendorfer et al., "Mot20: A benchmark for multi object tracking in crowded scenes," in *arXiv preprint arXiv:2003.09003*, 2020.

[45] A. Bartic, "Autonomous vehicles can perform an increasing array of tasks all by themselves," Flanders Make, 28 April 2020. [Online]. Available: https://www.flandersmake.be/en/blog/autonomous-vehicles-can-perform-increasing-array-tasks-all-themselves. [Accessed 1 February 2023].

[46] P. Polack, F. Altche, B. d'Andrea-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," *IEEE intelligent vehicles symposium (IV)*, no. IEEE, pp. 812-818, 2017.

[47] K. Jung, J. Kim, J. Kim, E. Jung, and K. Sungshin, "Positioning accuracy improvement of laser navigation using UKF and FIS," *Robotics and Autonomous Systems*, vol. 62, pp. 1241-1247, 2014.

 ISBN: 978-1-68558-053-7