# A Multi-modal AI Approach for Intuitively Instructable Autonomous Systems

Ferran Gebellí Guinjoan[1], Erwin Rademakers, Anil Kumar Chavali, 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 framework to intuitively instruct and control Automated Guided Vehicles. We define a general multi-modal AI architecture, which has a loose coupling between three different AI modules, including spoken language understanding, visual perception and Reinforcement Learning navigation. We use the same multi-modal architecture for two different use cases implemented in two different platforms: an off-road vehicle, which can pick objects, and an indoor forklift that performs automated warehouse inventory. We show how the proposed architecture can be used for a wide range of tasks and can be implemented in different hardware, demonstrating a high degree of modularity.**

***Keywords - AI based autonomous systems; Multi-modal AI; Natural language processing; deep learning; neural networks; reinforcement learning***

## I. INTRODUCTION

Autonomous Guided Vehicles (AGVs), which are often also referred as Autonomous Mobile Robots (AMRs), are becoming more and more popular in industrial applications. In previous works [1] [2] we presented two particular use cases where multi-modal AI leverages AGV tasks. In this paper, we propose a multi-modal AI framework that allows to intuitively and easily (re-)configure an AGV to perform different and variable tasks. The proposed multi-modal software architecture has a loose coupling between the different modules, which allows to easily exchange the components and deploy them in different hardware units.

AGVs can pick up and deliver materials around a manufacturing facility or warehouse [3]. However, with the continuously increase of mass customization [4], a return on investment of production AGVs can only be obtained if these AGVs can easily perform large variability of tasks and / or deal with large variability of products.

Task scheduling has been done by a central entity for a fleet of AGVs following predefined configurations. But driven by flexibility, robustness and scalability requirements, the current trends in AGV systems are customization and decentralization [5]. In a decentralized architecture, an AGV broadcasts the information about its states in a local way and decides which actions to take [6].

Although new generations of AGVs are highly instrumented with different sensors, they are more suited for long-distance transportation of materials between multiple destinations, and tuned for repetitive and predictable tasks [7].

(Re-)configurating AGVs to perform multiple tasks in a non-predictable environment remains, however, a challenge today in industrial settings due to dynamically changing environments. Classic navigation pipelines typically need to construct a map by scanning the environment with sensors, such as lidars [8], 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 industry settings, it is common that the environment is modified after short periods of time. This exposes the need for an increased flexibility in the whole navigation approach.

Research on a voice controlled AGV remains in the level of performing basic operations (e.g., moving with constant speed) in a prescribed path [9].

In this work we show how a general multi-modal architecture can be applied on two different use cases, which run on two different platforms (Figure 1). On the one hand, we implement an application on an off-road vehicle where the main task is to pick certain objects. On the other hand, we deploy an automated inventory monitoring on a forklift. In both cases, an operator can intuitively instruct the AGV by speech interaction that can be done locally or remotely.

In Section II, the common multi-modal AI architecture is presented. Section III explains the off-road vehicle use case, while Section IV describes the forklift use case. Finally, Section V contains the conclusions.

Figure 1. Platforms used for off-road vehicle picking objects (left) and forklift warehouse automated inventory (right) use cases.

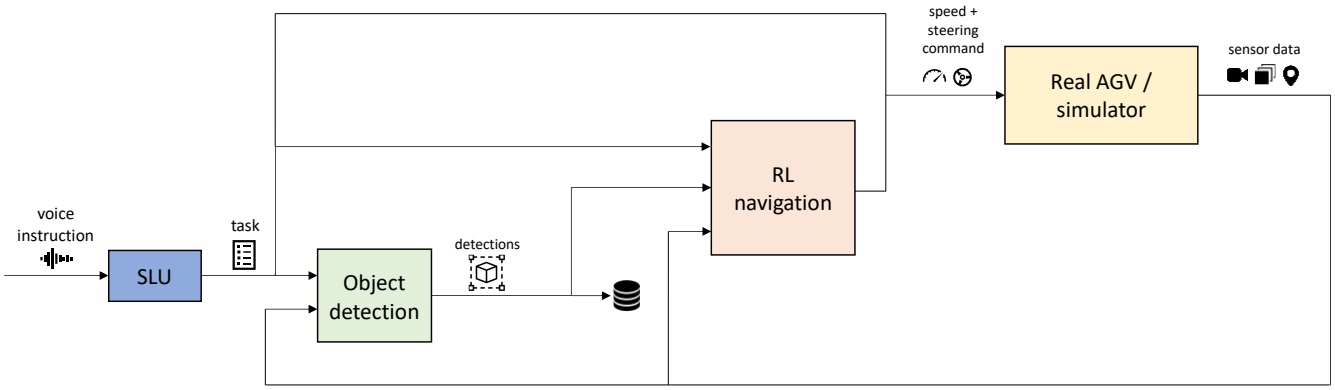


Figure 2. General architecture for a multi-modal AI autonomous platform.

## II. Multi-Modal AI Architecture

The presented multi-modal AI architecture (Figure 2) is a general software architecture for the implementation of autonomous vehicles that based on AI can perform a particular set of tasks, instructed by speech. The architecture defines the different modules and interface, and can be implemented in different platforms with different hardware typologies. Even within the same implementation, different modules can run in different hardware units. Because the architecture exhibits a loose coupling, the modules can be easily exchanged for other models or algorithms, as far as they share the same interface. The proposed interface has human-understandable signals, which helps to improve the explainability of the system. The suggested architecture has a directed flow of information between the modules, which is represented by arrows in Figure 2. This defines and constrains the exchange of information between the different modules. The architecture has 4 main building modules: **(i)** Spoken Language Understanding (SLU), **(ii)** association between speech cues with sensor data for objects detection and localization, **(iii)** RL for navigation, which uses information from, speech, vision and sensor data and (**iv**) vehicle platform, which receives motion commands and sends processed sensor data.

**(i)** Spoken interaction offers fast and natural interaction with machines and AGVs, while operators keep their hands and eyes free for other tasks. The task of a SLU component is to map speech onto an interpretation of the meaning of a command, while taking the variability in the input signal into account: differences in voice, dialect, language, acoustic environment (noise, reverberation), hesitation, filled pauses and pure linguistic variation. Traditionally, SLU is approached as a cascade of Automatic Speech Recognition (ASR) mapping speech into text followed by Natural Language Understanding (NLU) mapping text onto meaning. This cascaded approach tends to propagate and inflate ASR errors and requires application-specific textual data, which is unnatural to acquire. Instead, this work uses End-to-End SLU (E2E SLU), where spoken instructions are directly mapped onto meaning without textual intermediate representations. The output of the speech module is a semantic definition of the task, which is then used by all the other modules. This module provides the unique interface where the user can provide inputs.

**(ii)** For agents to interact with the environment, they must process and understand visual input, i.e., extract the semantically relevant cues from the environment in order to execute the desired task. Should the input be provided from an RGB camera, a plethora of Deep Learning techniques could be leveraged to achieve visual understanding. Deep Learning techniques rely on Neural Networks, commonly (pre-)trained on large-scale general-purpose datasets, e.g., for visual recognition [10] such as object detection [11]. Since our goal is to interpret a language-based instruction, we need to locate the object(s) in the environment. To this end, we build on state-of-the-art object detection methods. Given an RGB input, the object detector's role is to locate (detect) the relevant objects. This serves as a backbone to perform multi-modal interaction by associating the representation of the language-based instruction with the representation of the spatial layout of the scene (2D location and categories of the detected objects). The RGB can be enhanced with depth information (RGBD camera or lidar) and vehicle localization for precise 3D location of the detections in world coordinates. The output of the vision module is used by the navigation module. However, for some tasks (e.g., automated inventory, finding/locating an object, getting attributes of an object, etc.) the output of the vision module is itself the principal result of the task, and is saved in a database, which the user can access.

**(iii)** Egocentric navigation is one of the core problems intelligent systems need to master. An agent needs this skill not only to execute the task at hand, but also to navigate, in order to collect experience that can be used to learn from. In the presented approach we have chosen for an end-to-end learning-based navigation approach. Such an approach is able to outperform Simultaneous Localization and Mapping (SLAM) based approaches [12], it does not suffer from propagation errors due to mapping errors, and excels in visually sparse environments [13]. In our architecture, we foresee several available RL agents, each one trained for a specific set of tasks. The navigation module receives the task directly form the speech module, and switches to the appropriate RL agent. Sensor data coming directly from the platform is used for dynamic obstacle avoidance and general

exploration. Finally, the output of the vision module is used to direct the navigation to ensure that the exploration is done considering the relevant objects. As we need to train the RL agent in a simulation environment due to the large amount of required interactions, it is very important to couple the RL agent with a simulator that has the same interface as the real platform. Therefore, it is necessary to bridge the sim-2-real gap in two points: the acting gap and the observation gap. On the one hand, the acting gap refers to the interaction of the agent into the environment. For our architecture, this means making sure that the speed and steering commands have similar effects both in the real world and simulator. On the other hand, bridging the observation gap requires not only that the sensor data is similar in simulation and reality, but also that the simulator is able to produce similar object information as it would come from the real object detection.

(**iv**) The vehicle platform receives the control commands (set speed and steering wheel angle) from the navigation module. However, it can also be controlled directly by speed in case the speech task is directly affecting only navigation (e.g., "move slightly to the right slow"). The platform provides sensor data from the environment (camera, lidar and localization data) to the vision and navigation modules.

## III. CASE STUDY – AUTONOMOUS OFF-HIGHWAY VEHICLE

In this case study, the vehicle is able to navigate towards a specific object, which is in the field of view, given a speech command [1]. The AGV used in this case study consists of the off-highway tractor developed at Flanders Make [14]. To perceive the environment we use cameras, lidars, a GNSS system and a microphone. The sensors data is then processed in separate computing platforms and stored on middleware (ROS), from where the Speech and Vision units send the information to the control block. This later is divided in two levels, (i) a High-level controller that controls the tractor via a state machine and (ii) a Low-level controller, built in a dSpace platform [15], that controls the trajectory such that velocity and heading can be followed. The output signals are sent to different actuators that consist of the brakes, throttle, steering and fork implement that are controlled via servo motors. Autonomous vehicle upgrades to deal with Multi-modal AI

An example of intuitive instructions given by an operator to the AGV to execute a task and their high level interpretations by the Multi-modal AI framework, described in this paper, is illustrated in Figure 3.

The instruction: '*Pick up the red pallet and put it on the truck*', needs first to be communicated to the computer that runs the speech AI module (described in Section A). In the next level, a vision module, where real time 2d vision data is processed and fed to a pretrained NN, allows objects classification and their association to different attributes such as object's type, color, etc. (as described in Section B). The AGV should then move towards the recognized object. This step is supported by the association made so far between speech and vision data as well as the navigation data. This later makes use of the cartesian coordinates of the AGV in the navigation space and the reinforcement learning module (as described in Section III.C) that allows to estimate the optimal trajectory between the AGV and the object of interest.

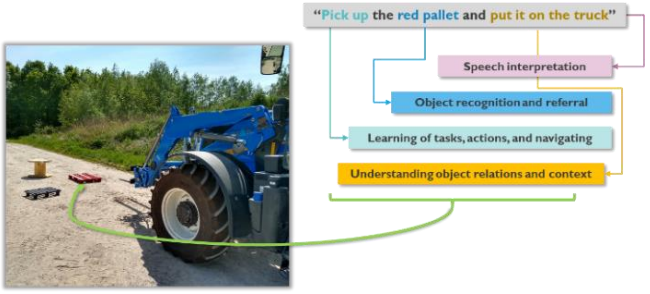


Figure 3. Example of speech-based instruction and multi-modal mapping.

In order to implement and demonstrate the Multi-modal AI framework, The AGV is updated by a newly installed system for interfacing through speech with a dedicated PC. This PC is also used for developing and testing the neural networks. It is equipped with a powerful Nvidia GPU and a new headset microphone for giving audio commands. The autonomous tractor internally uses ROS to communicate between the different sub-systems. Originally it was only used sparingly in the autonomous tractor, mainly to communicate lidar sensor data. After the system upgrade, also the control unit, the dedicated PC and the Nvidia Drive platform have a ROS interface. While the Nvidia Drive could technically runs the neural networks, for more convenience, during testing we installed the neural networks on the dedicated PC. Data from the cameras on the Nvidia Drive, LiDAR and navigation all come in as ROS messages while for speech a simple microphone is connected to the PC. The output of the multi-modal setup is the location of a specific object together with the task the tractor must complete. This information can be communicated through ROS to the navigation module.

### A. *Spoken language understanding*

#### 1) *Speech data generation*

To train the SLU model, training dataset with audio fragments is made. It is important that the recorded speech seems natural, as if the participants are really interacting with the AGV. To this end, we believe that a visual feedback to the participant would be very useful. Therefore, a simple automotive simulator called Webots [16] was used and a set of API calls were written in order to control the simulated tractor in the simulated environment (Figure 4).

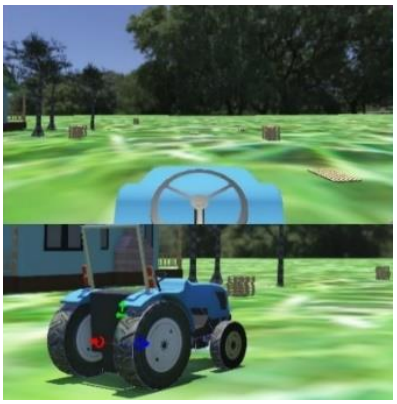

Figure 4. Simulator that provides visual feedbacks to participants for speech recording.

The participants are given some high-level objectives and it is up to them to control the tractor with speech commands in order to fulfil these tasks. With the '*high-level*' objectives (in contrast with explicitly providing the primitive commands to the participants) we aim to improve the variability of commands that participant's would naturally choose to control the tractor. Every time the participants speak a relevant command, the experiment supervisor presses a button to invoke the correct API call. This way, we already have some automatically generated annotations linking the participant's speech command to the supervisor's API call invocation. We recorded the audio in Audacity in WAV format using a headset microphone and a separate standalone microphone. The commands were mainly basic control commands like turning a direction or driving speed. A total of 14 people who speak *Dutch* language (different dialects) were recorded with mixed female and male voices.

*2) SLU model architecture & training*

Classical *semantic frames* are used for representing the semantics of an utterance. A semantic frame is composed of *slots* (e.g., "direction") that take one of multiple *slot values* (e.g., "forward" or "backward"). This encoding represents the affordances of the AGV and corresponds to API calls with parameters filled in. The task of the SLU component is to map an utterance (spoken command) to a completed semantic frame. The SLU architecture follows the encoder-decoder structure first described in [17] and later refined in [18] to allow for encoder pretraining for ASR targets on generic *Dutch* data. The decoder is trained on the task-specific data.

The encoder encodes an utterance in a single high-dimensional embedding in two steps. The first step maps MEL-filterbank speech representations to letter probabilities using a transformer network [19] preceded by a down sampling CNN, trained maximal cross-entropy between predicted and ground truth transcriptions in a 37-letter vocabulary. The training data consist of 200 hours of Flemish speech with its textual transcription from the CGN corpus [20], fourfold augmented with noise (0-15 dB) and reverberation (sampled from [21]) to achieve acoustic robustness. The second step counts bigram occurrence frequencies of all letter pairs across the utterance and repeats the same while skipping one position in the bigram, resulting in a $2(37^2) = 2738$ dimensional utterance embedding.

The decoder maps the utterance embedding onto a multi-hot encoding of the slot values via non-negative matrix factorization (NMF) [22] as described in [17]. Other than in the pretraining stage, the training pairs here do not require textual transcription, but are pairs of speech with the completed semantic frame. Here, a neural network could be taken as well, but the chosen decoder has several advantages: (1) it requires few training data, (2) it retrains in a fraction of a second when user interaction data becomes available and (3) it establishes a bag-of-words model making the SLU system less sensitive to the rather free word order in *Dutch* (at least compared to *English*). Learning a stricter word order would require more task-specific training data exhibiting the word order variability.

The approach is evaluated on the Grabo corpus [23], which contains a total of 6000 commands to a robot spoken by ten *Flemish* speakers and one *English* speaker. The commands were recorded with the participants' own hardware in a quiet room at their homes. The semantics are described in eight different semantic frames describing driving, turning, grabbing, pointing, etc. using one (e.g., "*close* gripper") to three (e.g., "*quickly* drive *forward a little bit*") of ten slots (e.g., angle, direction, etc.), which can take between two and four different values. In total, 33 different meanings occur in the data. The accuracy is evaluated as the F1-score for slot values as a function of the number of task-specific training examples. The trained decoder is speaker-specific. The average accuracy over speakers is plotted in Figure 5 and shows that with the minimal of 33 training utterances, i.e., one example per meaning, an accuracy of over 98.5% is reached. The performance saturates around 180 task-specific utterances.

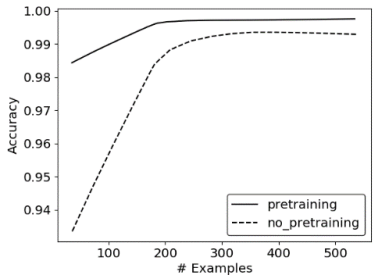


Figure 5. F1-score as a function of the task-specific training examples.

*3) SLU model validation*

For deployment we set up a docker container to run all the code. We developed a user interface to be able to easily visualize the results of the SLU model and provide training examples for training the decoder. In this interface, it is possible to record samples, open the microphone so the tractor can listen, give feedback to the model and retrain the model. After each command is given the confidence value of the prediction is estimated. Commands with sufficient confidence are forwarded to the tractor through ROS to the control PC.

The initial accuracy of the model depends a lot on the person giving the commands and their accent. But we were able to achieve high levels of accuracy of more than 90 percent in the noisy tractor environment using an active learning approach. In this approach, the operator can give feedback samples to retrain the model. In this experimental set-up, repeating an instruction in 5 instances proved to achieve high accuracy (90%). The retraining flow is quite time-efficient and takes less than a second to retrain.

*B. Visual perception*

*1) Vision AI Objects detection and classification*

*a) Vision data generation*

The dataset for training the vision model contains images with mostly objects that the AGV can pick up. This means mostly pallets and boxes of varied materials, shape and sizes containing materials like bobbins and wooden planks. This data was recorded on the Flanders Make local site, spread over two occasions: one on an early cloudy morning in spring

TABLE 1. QUANTITATIVE EVALUATION OF THE VISION AI TRAINED MODEL

| | Vid. 1 | Vid. 2 | Vid. 3 | Vid. 4 | Vid. 5 | Vid. 6 | Vid. 7 | Vid. 8 | Vid. 9 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| mAP | 55.04 | 40.90 | 56.03 | 66.42 | 68.35 | 50.50 | 65.25 | 61.9 | 51.42 | 57.30 |

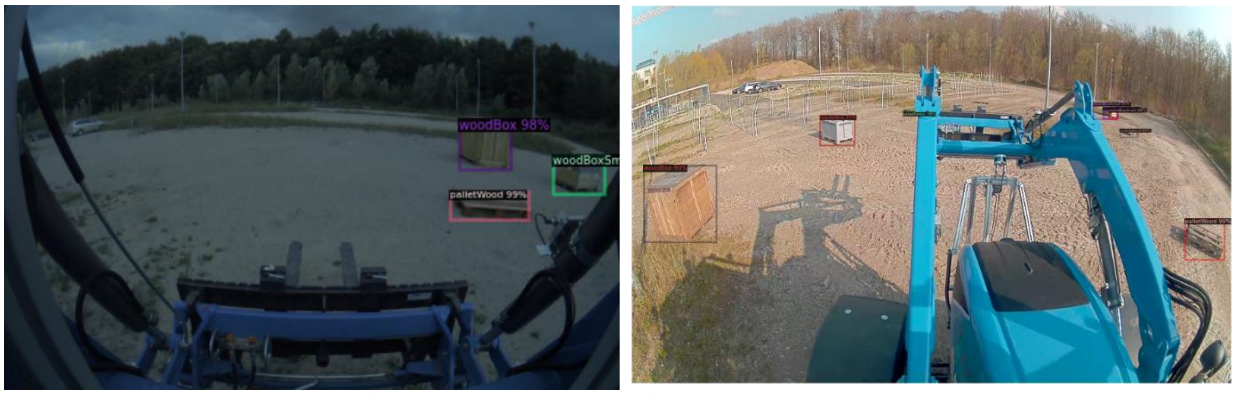


Figure 6. (left) all objects are correctly classified, (right) some objects are not detected.

and one just after noon in summer with sunny weather. Every image was recorded with a resolution of 960 x 608 pixels. The entire dataset contained 1100 images, derived from 9 videos. Each of these videos recorded one configuration of objects from many angles.

*b) Vision NN architecture & training*

The main building block of the vision pipeline is the object detector. It gets an RGB image $I$ as input, where $I \in \mathbb{R}^{3 \cdot H \cdot W}$ and $H$ and $W$ are the image height and width respectively. The model we use is a state-of-the-arts two-stage object detector, where in the first stage, a region proposal network generates regions of interest for the image, and in the second stage, bounding boxes and object classes are predicted for each proposal, which exhibits an objectness score above a certain threshold. The region proposal network generates region proposals by sliding a spatial window over features map obtained from a Convolutional Neural Networks (CNN), i.e., a backbone. Additionally, the object detector includes a Feature Pyramid Network [24], a fully-convolutional module, which generates features maps at different levels, thus enabling the model to recognize objects at different scales. The object detector we use is a Faster R-CNN [25], with a ResNet101 backbone [26], pre-trained for general purpose object detection on COCO [11].

Even though less resource intensive FasterR-CNN backbones exist, such as MobileNets [27], given our computational budget, we find the FasterR-CNN variant we use to yield the best tradeoff between detection performance and speed (near real-time).

The model's outputs are object bounding boxes and classes with a confidence score for each. The confidence score for the predicted class is obtained as the Softmax probability of the highest scoring class.

We perform fine-tuning of the Faster R-CNN on images consisting of scenes from the environment, where the objects of interest are annotated with bounding boxes and classes. The images we use are video frames, extracted from 9 videos of the AGV navigating the environment while encountering the objects. Considering that the amount of data at our disposal is limited, we determine the optimal hyperparameters by training the object detector in a leave-one-out fashion, such that we train on a subset of 8 videos and perform evaluation on the remaining one. We iterate this process until we train a separate model on all unique subsets. The final model performance is averaged over each of the videos. We evaluate the model's performance using the standard COCO [11] mean average precision (mAP). The final model, i.e., the model used in the AGV, is trained on all 9 videos using the hyperparameters determined during the leave-one-out training/evaluation process.

We train the model for 5 epochs with a learning rate of 1e-4. We perform random horizontal flip data augmentation, enabling us to synthetically increase the dataset size and make the detector invariant to such transformations of the data. We sample a subset of 128 region proposals to estimate the regression and classification loss of the region proposal network.

We quantitatively evaluate each of the trained models on the videos, which were held-out during training. In TABLE **1** the lowest score is highlighted in red, while the highest scoring one is green. Overall, we observe that the performance is relatively high across all different videos (57.3 mAP). We further observe that the performance on Video 2 (Vid. 2), is significantly lower compared to the average performance. To inspect the reason for the lower performance, we qualitatively inspect the samples from Video 2 as discussed below.

We qualitatively evaluate the object detector's performance by visualizing the predictions on the held-out videos during training. In Figure 6 (left), we observe that the model correctly predicts all objects, which is in line with our expectations as the objects are fully visible and of a reasonable size. On the other hand, in Figure 6 (right) we observe several mis-detected objects of a frame from Video 2. We conclude that even though the model performs well, it struggles to recognize objects, which are (1) far from the camera (small size), and (2) occluded in the environment – both of which are active areas of object detection research.

*2) Visual grounded SLU*

To deal with the data sparsity, and to be able to ground (localize) the speech model output in the image, we perform discretization of the spatial layout (the bounding boxes and classes obtained as output from the object detector). To be specific, we perform mapping of both modalities to a canonical space, where we later measure the similarity between the output of the speech model and each of the detected objects in the image. To that end, we encode each detected object as a collection of one-out-of-k encodings of its label (box, pallet, etc.), material (wooden, plastic, etc.), size (regular or small), and location in the image. Note that the object category, material and size are jointly predicted by the object detector as object class. Lastly, we quantize the location of the object, i.e., we represent the object's location based on the object's horizontal and the lower vertical position. We showcase the grid over the image including the spatial references according to the x and y axes in Figure 7.

Finally, we represent each detected object as a vector of size 12, where we allocate 3, 2, 3, 3, 1 indices for the object's class, material, x-location, y-location and size respectively. When measuring the similarity between the speech model output (a vector of size 12 as well) and each encoded object detection, we explore different weighting strategies for each object attributes, which we discuss next.

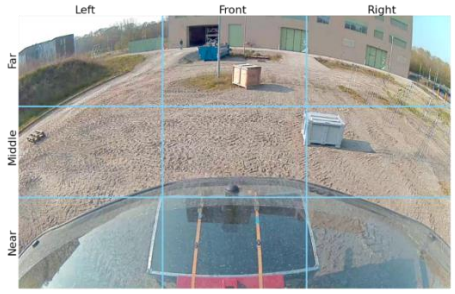


Figure 7. Grid over image with object's spatial reference.

*3) Adding Spatial relations*

We evaluate different strategies for measuring the similarity between each (discretized) object detection and the speech model output. The output is a bounding box, which represents the grounding location of the instruction. We evaluate each grounding strategy on two variants of the dataset, namely (1) a descriptive variant, where the objects are commonly described based on their attributes, e.g., pick up the wooden box, and (2) a spatial variant, where the referred object is described based on its location in the frame, e.g., pick up the box furthest on the left. The grounding strategies we evaluate are:

1. Random matching (RM): A naïve baseline, where we ground the speech given instruction to a randomly selected bounding box. We establish a lower bound on the grounding performance with this baseline.
2. Basic matching (BM): We obtain the dot-product between the one-hot encodings of speech instruction and each object detection, representing the similarity.
3. Weighted matching (WM): We (re-)scale the contributions of the individual elements in the dot-product with pre-defined weights.
4. Confidence matching (CM): We represent the speech model with the confidence scores.
5. Weighted confidence matching (WCM): We use confidence scores for the speech model output and additionally weight the individual contributions using the pre-determined weights.

We perform evaluation using the standard grounding accuracy metric, where we score a hit if the predicted grounding bounding box has intersection over union (IoU)>0.5 with the ground truth box. For the random baseline, we perform inference 5 times and report the average performance. Through grid-search, for the weighted modules (WM, WCM) we use a weight of 0.1, 0.7, 0.2, and 0.05 for the spatial indicators, the object class, the object material and the object size respectively. We report the results in TABLE 2.

We observe consistent gains when we weigh (WM) or use the speech model confidence scores (CM) in the grounding, compared to the baseline basic matching (BM) method. Additionally, a combination of the weight and confidence matching (WCM) yields superior results across the different data (descriptive, spatial) and significantly outperforms the other methods. Lastly, even though the spatial data is more challenging than the descriptive data, the WCM module performs well, indicating that by re-weighting and adding confidence scores, we can ground spatial speech data reasonably well.

*C. Reiforcement learning based navigation*

In this section, the navigation part of the Multi-modal AI and its association with the speech-vision data is described. The currently developed proof of concept consists of a simulation environment with the hardware in the loop.

To make this simulator as close to real life as possible, a 3-D scan of the test environment by using an aerial scanning using a drone with photogrammetry capabilities that allows us to map images to a high fidelity 3-D twin of the area. This twin was then imported to the simulator for the purpose of reinforcement learning.

*1) RL archiecture & training*

The presented Reinforcement Learning (RL) approach makes use of the DD-PPO (Decentralized Distributed Proximal Policy Optimization) architecture [28] (Figure 8). The Reinforcement Learning (RL) approach is able to map high dimensional inputs to discrete actions. The DD-PPO model consists of a visual pipeline, for which in our case we use a ResNet18 [26].

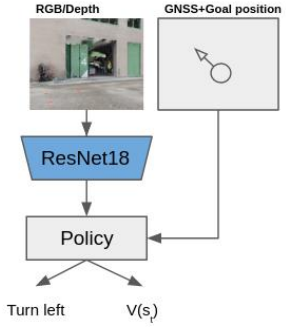


Figure 8. DD-PPO architecture overview.

TABLE 2. EVALUATION OF THE AI MODEL WITH SPATIAL RELATIONS

| Method | Dataset type | |
|---|---|---|
| | Descriptive | Spatial |
| RM | 25.91 | 17.14 |
| BM | 65.91 | 59.52 |
| WM | 70.45 | 57.94 |
| CM | 76.14 | 62.70 |
| **WCM** | **79.55** | **65.87** |

The resulting learned visual representation is concatenated together with a GNSS sensor. This output is then passed onto a recurring policy consisting of 2 Long short-term memory (LSTM) **[29]** layers. The final outputs of the model consists of a state value estimation, and an action distribution from which actions (move forward, turn left, turn right and stop) can be sampled. The stop-action should be executed by the agent when positioned less than 2 meters of the goal position. As inputs for the model we tested a single depth camera, a single RGB camera, or a combination of both RGB and depth. We use these sensors as they are cheap and widely available. The camera is positioned on the front of the AGV.

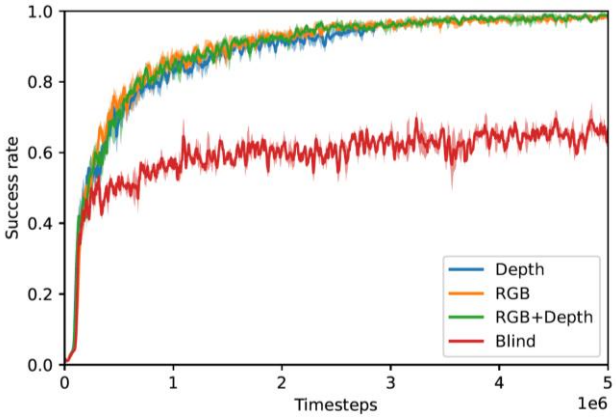


Figure 9. Training performance. The blind agent can perform basic navigation by relying on the GNSS sensor, however to further improve to near perfect results an additional RGB of depth sensor is required to detect and avoid collisions.

To train the agent we use the improvement in geodesic distance between the agent and the goal position as a dense reward signal. A slack penalty of -0.01 is subtracted on each step, and a termination bonus of 2.5 is awarded upon successfully utilizing the done action. We train the agent entirely in the Habitat simulator [12] where a photorealistic scan of the environment is used. This allows the agent to interact with the terrain in a safe way. While in this case we trained the agent to specifically work on a single environment, DD-PPO also allows generalization to unseen environments, given enough different training environments and training samples. Figure 9 shows the required number of interactions with the environment. These results indicate that in this setting the agent relies mostly on the GNSS sensor, as the blind agent performs reasonably (60% success rate after 5M training interactions). However, by adding either a depth or RGB sensor the agent achieves near perfect navigation capabilities on the training set after 5M interactions with the simulated environment.

*2) RL validation*

Realizing Reinforcement learning on a large autonomous platform brings in multiple challenges to the board. For safety concerns, the approach to validate the system was to use a Hardware-in-loop setup (Figure 9) along with the digital twin of the environment. The main input from the real world was the signal from the GNSS receiver (Septentrio AsteRx-U) on the AGV, which was then mapped to the digital twin coordinates system. The GNSS had a dual antenna setup, which could then provide the heading of the platform as well. Using a cloud-based service updates were provided in real time to the simulator/digital twin environment to position the simulated tractor same as the one in real world. The output from the simulator was the suggested trajectory to the goal pose.

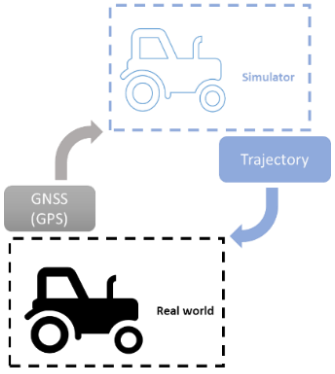


Figure 10. Hardware In Loop setup (overview).

To evaluate the navigation capabilities of the agent, we created a holdout dataset. This holdout dataset contains goal positions the agent did not see during training. TABLE 3 contains the results of 100 tested episodes. In TABLE 3, the success rate indicates the amount of episodes the agent could complete successfully. The Success weighted by Path Length (SPL) measurement also considers the length of the path taken.

TABLE 3. SUMMARY OF TESTED EPISODES

| Sensors | Success Rate | SPL | Avg. Collisions |
|---|---|---|---|
| **RGB** | 100% | 0.9454 | 0.4355 |
| **Depth** | 100% | 0.8882 | 0.1129 |
| **RGBD** | 100% | 0.9272 | 0.5161 |
| **Blind** | 91.94% | 0.7294 | 4.3548 |

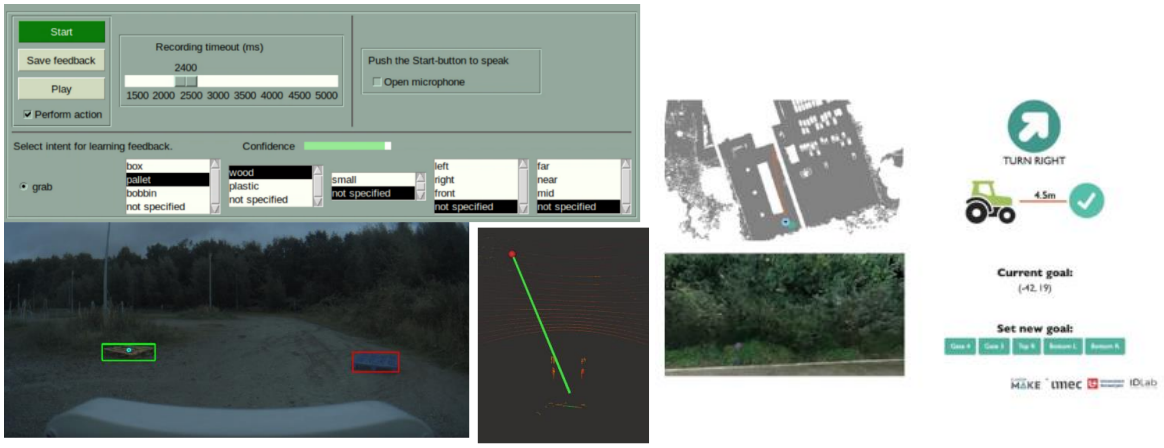


Figure 11. Snapshot of the demonstrator of the AGV Multi-modal AI framework: (top left), the Speech model interface, (bottom left), the Vision model interface, (right), the Navigation digital twin interface, (bottom middle), the estimated trajectory between the AGV.

### D. *AGV Multi-modal AI Demonstration*

To demonstrate the full methodology, we combined the methods respectively described in Sections A, B and C in one demonstrator implemented in the AGV. We added all the information in a new docker environment to be able to run on the dedicated PC in the AGV. There is a similar user interface compared to the SLU model where you can record your voice and use the NLP model to predict the voice commands. These commands consist of the description of the object and the task the AGV should do. Then the fusion model uses this information to link an object description with a detection from the vision model to predict the location of the describer object on the image. As a last step the lidar data is used to link the 2D location on the image to a 3D location of the object in the world coordinate space. This location can then be sent further as a goal to the control systems together with the described task from the NLP model. A significant improvement could be made in the parameters of the fusion model. There was a bias against using spatial information in the voice command. The material of the object is more difficult to extract on the image than its location, so using the location for finding the correct object is more reliable. Hence, we tuned some of the weights to have a bigger focus on this kind of information. Another small improvement could be made to the audio side. The person dedicated to controlling the AGV added some voice samples and gave feedback to the model through the user interface. This way the model was more confident in recognizing their accent and way of talking. With regards Navigation, although the approach is not fully implemented in the rea system, the approach can already be demonstrated by Hardware-In-the-Loop. In this setting an instance of the simulator is constantly synchronized with the AGV. This is done by using the GNSS position from the real-world AGV to set the position of the agent in the simulator. We can use the digital twin to generate trajectory paths. These generated trajectories can then be used in the real-world by the AGV. A snapshot from the full demonstrator is depicted in Figure 11.

## IV. CASE STUDY – INDOOR AUTOMATED INVENTORY

The second case focuses on the task of automated inventory of unknown warehouse settings [2]. The goal is to explore with a good tradeoff between navigation time and inventory accuracy. An open experimental platform has been built on top of an AGV, automating a standard pallet forklift [30]. 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.

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 [31]. This model is used in the RL training bridge the sim2real gap in the actuation. In Figure 12 the vehicle model can be seen.

The kinematics for a forklift AGV are defined by the following equations [32]:

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

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

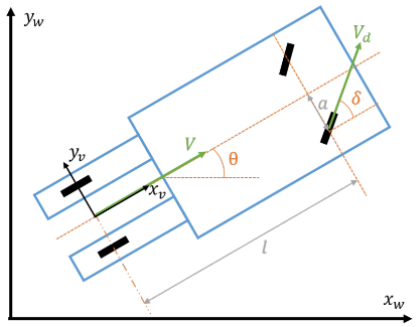


Figure 12. Bicycle kinematic model for AGV.

### A. *Spoken language understanding*

We have used the same SLU module than the one from the previous use case and we have retrained it to work for a new set of tasks. For this use case we have trained the model in English, showing that the speech recognition can work well in different languages given pairs of audio signals and tasks.

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. Three different kind of possible tasks have been selected for this use case. First, a command is available to start a new inventory session ("count"). Then, there are 3 options available: steer the AGV manually, trigger the RL autonomous exploration ("explore"), or further give speech instructions to control the movement of the AGV ("move"), such as "forward", "a little bit to the left", or "stop".

### B. *Visual Perception*

#### 1) *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 13. 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.

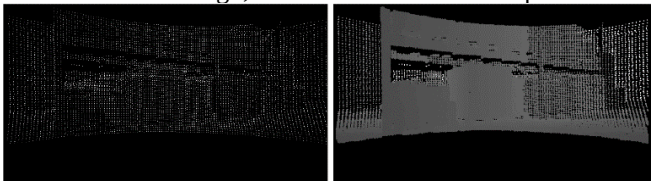

Figure 13. Depth image from the point cloud without inflation (left) and with inflation (right).

We choose the Yolov7 detector [33], 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 [11], 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 [34] as an object tracker, because it can be easily coupled with any other detector and yields to good accuracy in the MOT20 [35] benchmark. The main building block is a Kalman filter [36] 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.

#### 2) *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.

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.

**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 14. Algorithm for inventory monitoring

- 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.

### *C. Reiforcement learning based 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 15) 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.

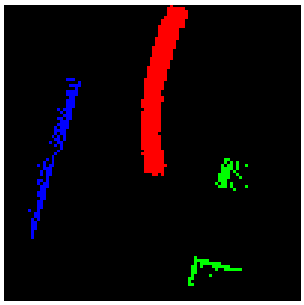

Figure 15. Input image to the RL agent. Blue are obstacles, green represents uncertain areas and red is the past trajectory

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) [37] to train the agent.

### *D. AGV Multi-modal AI Demonstration*

We have integrated all the algorithms in the forklift AGV platform and performed several online real-time experiments. Figure 16 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 16 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 4 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 4.

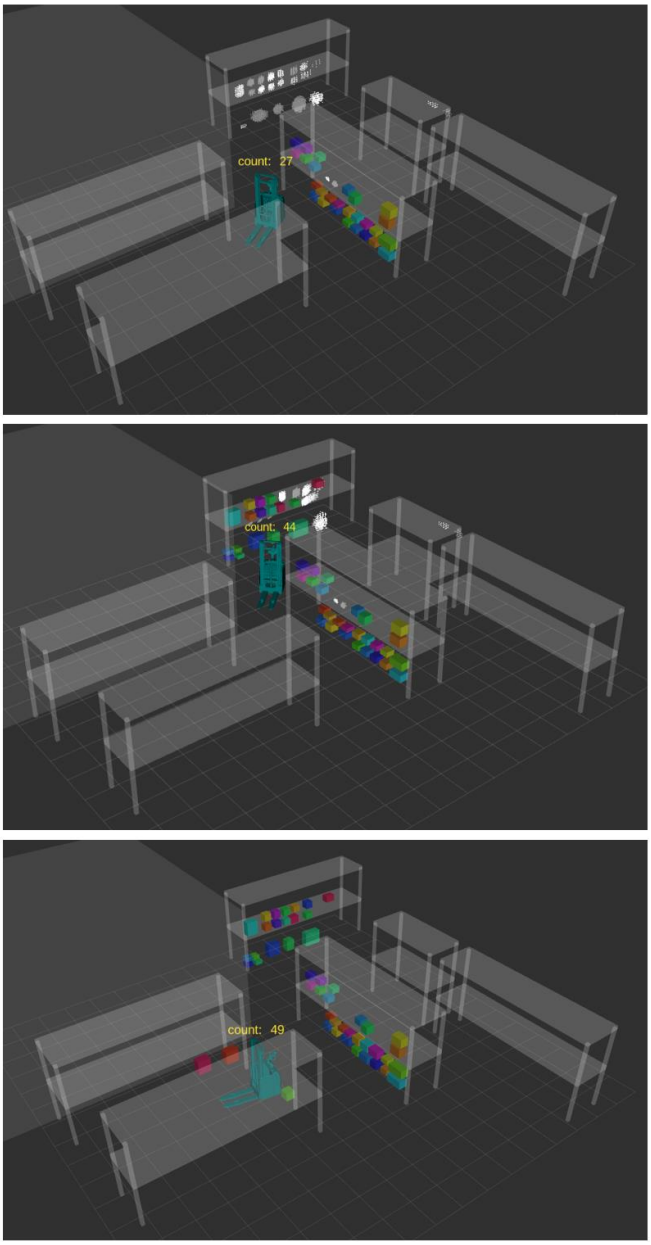


Figure 16. Sequence of the forklift around some racks in a warehouse.

TABLE 4. 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 |

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. CONCLUSION

In this work, we developed and demonstrated a multi-modal AI framework that allows to intuitively instruct production AGVs to perform multiple tasks. The interface with operators is allowed by speech interaction that is decoded through an AI NLP model to translate speech commands to interpretable instructions by all the components of the AI Framework. Associations with vision and navigation data are done to be able to perform a wide range of tasks. We show how the loose coupling between the modalities creates a an architecture which is general enough to be applied in a wide set of tasks for two different use cases, which run on very different hardware platforms. Moreover, the loose coupling of the modules provides a clear interface between the modalities (e.g., task, object detections, motion commands) which is interpretable by humans, thus leveraging the explainability. The main outputs of the system are the control commands that enable the vehicle navigation, and relevant task information (e.g., location of objects) which is provided to the user. The demonstrators remain, however, a research proof of concept (to demonstrate the approach) and require different improvements before effective industrial usage. This includes, amongst others, training with larger datasets (speech, vision, navigation) and evaluation in an extended number of scenarios. Moreover, bridging the sim-2-real gap for the RL navigation is still a challenge in terms of achieving the necessary robustness for industrial applications.

## 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. 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, 2022.

[2] F. Gebelli Guinjoan et al., "A Multi-modal AI Approach For AGVs: A Case Study On Warehouse Automated Inventory," *The Nineteenth International Conference on Autonomic and Autonomous Systems*, pp. 25-33, 2023.

[3] D. Li, B. Ouyang, D. Wu and Y. Wang, "Artificial intelligence empowered multi-AGVs in manufacturing systems," in *ArXiv abs/1909.03373*, 2019.

[4] L. Radder and L. Louw, "Mass customization and mass production," *The TQM magazine*, vol. 11, pp. 35-40, 1999.

[5] 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, pp. 152-173, 2020.

[6] D. Herrero-Perez and H. Martinez-Barbera, "Decentralized coordination of automated guided vehicles," *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, vol. 3, pp. 1195-1198, 2008.

[7] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri and S. Z. Md Dawal, "Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization," *PloS one*, vol. 12, p. 12(3): e0169817, 2017.

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

[9] S. HT and C. Arjun, "Design of Voice Controlled Automated Guided Vehicle," *International Journal of Science Technology & Engineering*, vol. 3, pp. 90-93, 2017.

[10] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 6, pp. 84-90, 2017.

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

[12] 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.

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

[14] Flanders Make, "Automated off-highway vehicle test platform," [Online]. Available: https://www.flandersmake.be/en/testing-validation/product-validation/automated-off-highway-vehicle-test-platform. [Accessed 10 2 2023].

[15] dSpace, "Real-time testing system (dSpace)," [Online]. Available: https://www.dspace.com/en/pub/home.cfm. [Accessed 10 2 2023].

[16] Cyberbotics, "Webots - Open source robot simulator," [Online]. Available: https://cyberbotics.com/. [Accessed 2 10 2023].

[17] B. Ons, J. F. Gemmeke and H. Van hamme, "Fast vocabulary acquisition in an NMF-based self-learning vocal user interface," *Computer Speech & Language*, vol. 28, pp. 997-1017, 2014.

[18] P. Wang and H. Van hamme, "Pre-training for low resource speech-to-intent applications," in *arXiv preprint arXiv:2103.16674*, 2021.

[19] A. Vaswani et al., "Attention is all you need.," *Advances in neural information processing systems*, vol. 30, 2017.

[20] N. Oostdijk, "The Spoken Dutch Corpus. Overview and First Evaluation," in *Proceedings of LREC*, 2000.

[21] RWTH Aachen, "Aachen Impulse Response Database," [Online]. Available: https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/. [Accessed 10 2 2023].

[22] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.

[23] KU Leuven, "ALADIN: Adaptation and Learning for Assistive Domestic Vocal Interfaces," [Online]. Available: https://www.esat.kuleuven.be/psi/spraak/downloads/. [Accessed 10 2 2023].

[24] T.-Y. Lin et al., "Feature Pyramid Networks for Object Detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117-2125, 2017.

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

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

[27] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[28] E. Wijmans et al., "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," *arXiv preprint arXiv:1911.00357*, 2019.

[29] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, pp. 1735-1780, 1997.

[30] 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].

[31] 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)*, pp. 812-818, 2017.

[32] 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.

[33] 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.

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

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

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

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