Applied Intelligence (2025) 55:206
https://doi.org/10.1007/s10489-024-06115-4

Visual semantic navigation with real robots

Carlos Gutiérrez-Álvarez[1] · Pablo Ríos-Navarro[2] · Rafael Flor-Rodríguez-Rabadán[1] · Francisco Javier Acevedo-Rodríguez[1] · Roberto Javier López-Sastre[1]

Accepted: 23 November 2024

Abstract
Visual Semantic Navigation (VSN) is the ability of a robot to learn visual semantic information for navigating in unseen environments. These VSN models are typically tested in those virtual environments where they are trained, mainly using reinforcement learning based approaches. Therefore, we do not yet have an in-depth analysis of how these models would behave in the real world. In this work, we propose a new solution to integrate VSN models into real robots, so that we have true embodied agents. We also release a novel ROS-based framework for VSN, ROS4VSN, so that any VSN-model can be easily deployed in any ROS-compatible robot and tested in a real setting. Our experiments with two different robots, where we have embedded two state-of-the-art VSN agents, confirm that there is a noticeable performance difference of these VSN solutions when tested in real-world and simulation environments. We hope that this research will endeavor to provide a foundation for addressing this consequential issue, with the ultimate aim of advancing the performance and efficiency of embodied agents within authentic real-world scenarios. Code to reproduce all our experiments can be found at https://github.com/gramuah/ros4vsn.

Keywords Robotics · Embodied agents · Vision-based navigation · Artificial intelligence · Reinforcement learning

1 Introduction

Can a robotic agent navigate and interact in the real world as seamlessly as humans do? This is the fundamental question driving research within the embodied AI community. The problem is formally known as Visual Semantic Navigation (VSN), *e.g* [1–3]. However, mimicking human navigation is a challenging task for robots, particularly in unseen environments, as it requires efficient exploration and a deep understanding of the objects and structures within the space. For unknown scenarios, humans can leverage prior semantic information achieved from previous scenes to navigate in new environments, but it is still a challenging task to incorporate that knowledge into embodied agents, especially in real robotic platforms navigating in the real world. The potential of autonomous robots with these advanced navigation capabilities is vast, ranging from assistive robots that can guide individuals with reduced mobility to specific locations, to platforms that can aid in complex environments such as search and rescue operations or logistic centers.

Technically, in this work, we focus on embedding VSN models in real robotic platforms. This is our main objective. We focus on the Object-Goal Navigation (ObjectNav) [4] problem. As it is seen in Fig. 1, in ObjectNav task the agent has to navigate from a random position to certain object goals present in the scene, mainly using vision-based sensors. In contrast with traditional geometric navigation approaches, where the navigation problem is solved typically

Carlos Gutiérrez-Álvarez and Pablo Ríos-Navarro contributed equally to this work.

✉ Carlos Gutiérrez-Álvarez
carlos.gutierrezalva@uah.es

Pablo Ríos-Navarro
pablo.rios@urjc.es

Rafael Flor-Rodríguez-Rabadán
rafael.flor@edu.uah.es

Francisco Javier Acevedo-Rodríguez
javier.acevedo@uah.es

Roberto Javier López-Sastre
robertoj.lopez@uah.es

[1] Signal Theory and Communications, University of Alcalá, Alcalá de Henares 28805, Spain

[2] Computer Science, Rey Juan Carlos University, Móstoles 28933, Spain

Published online: 23 December 2024

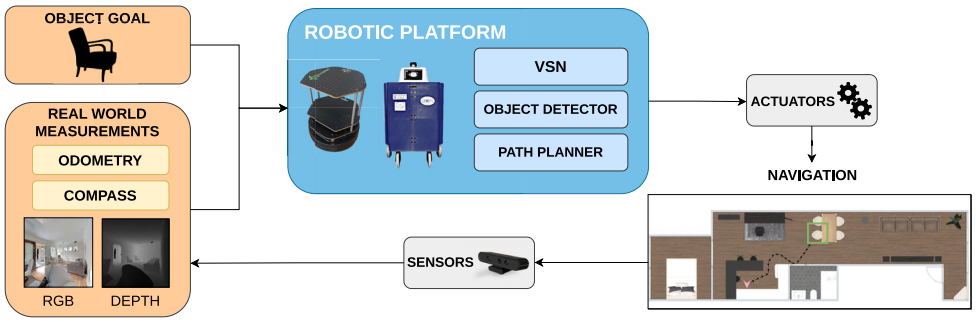


Fig. 1 ObjectNav task is a complex navigation problem. An agent needs to employ vision-based sensors to navigate from a random starting point to specific object goals within the scene. Many different hardware and software components need to be fully integrated to solve it, making it difficult to deploy and test these visual semantic navigation (VSN) models in real robots. Therefore, the best current solutions are trained and tested in virtual environments. Our goal is to bridge the gap between virtual and physical environments by providing a ROS-based framework that simplifies testing and comparing various VSN models on real robotic platforms

by using a map or generating it on the fly (e.g. SLAM), VSN models are learning-based approaches that use *no* metric map. Therefore, VSN solutions must learn visual representations of the environment to reduce the exploration time and better generalize to unseen scenes and object categories. Notably, many of these solutions combine reinforcement learning (RL) and/or imitation learning (IL) strategies with recent advances in deep learning models for visual perception to address the navigation problem in *virtual* environments, where embodied AI agents are trained and tested. However, we believe it is crucial to thoroughly investigate how the latest solutions to the VSN problem perform when deployed with real robots navigating in real-world environments. This is where our work makes its most significant contribution. Although there are VSN works that achieve *near-human* navigation performance, *e.g* [1], these systems are mainly trained and tested in virtual environments, so how these models would behave in the real world is still a question to answer.

In this work, our objective is to build true embodied agents proposing a new solution for the integration of VSN models into real robots. The main contributions of our work are as follows:

1. We first provide to the embodied agents community the ROS4VSN development, a novel Robot Operating System (ROS) [5] based architecture that allows testing and comparing different VSN models in real robots. Our ROS4VSN development is model agnostic; thus, any VSN solution can be integrated into it with ease. Section 3.1 includes all the technical details.
2. We also have embedded two state-of-the-art VSN models in two different robotic platforms. The selected models have been PIRLNav [1] and VLV [3]. To achieve this integration, it is necessary to make technical adaptations to the models so that they transition from interacting with observations provided by simulation environments to observations from the real world. We detail all these technical modifications to the models in Section 3.2.
3. Finally, we propose an experimental evaluation of the adapted VSN models, using our ROS4VSN, with two different robots, in a real world scenario (Section 4). The main question we want to address with the designed experiments is: Are the state-of-the-art VSN models able to successfully operate with real robots? We have measured their success rate in real world navigation experiments, which has allowed us to analyze the difference in performance compared to those tests in simulation environments, where embodied AI agents are typically tested. Our work shows that these VSN solutions perform noticeably differently when evaluated in real-world and simulated situations. With the ultimate goal of improving the performance and efficiency of VSN systems in real robots, we hope that our work will help to provide the groundwork for tackling this significant challenge.

2 Related work

Visual Semantic Navigation. To navigate in unfamiliar environments, traditional methods use depth sensors [6, 7] and RGB cameras [8, 9] to build geometric maps and simultaneously determine the robot's position in relation to the map. This is known as Simultaneous Localization and Mapping (SLAM) [10–12]). Typically, these SLAM models use heuristic algorithms to create graph-based representations of the environment, allowing the robot to visit the different nodes of the graph when navigating to specific points. Semantic SLAM (*e.g* [13–15]) expands upon SLAM by integrating semantic data from the environment, allowing the robot to identify and store objects in memory.

A recent approach, made possible by advances in machine learning and computer vision, involves designing navigation policies that directly train deep neural networks to learn

semantic information from visual observations in an end-to-end fashion (*e.g* [3, 16–20]). This approach is termed *visual semantic navigation* (VSN). These models often rely on the use of CNNs as visual encoders followed by RNNs; that are in charge of predicting an action distribution directly from raw input observations. The neural networks are trained using imitation learning (IL) or reinforcement learning (RL) approaches.

When IL is applied to the visual navigation problem, navigation policies are learnt from expert demonstrations (*e.g* [16, 17]). It can also be used combined with an RL fine-tuning phase to achieve better performance [1].

Other works focus on the use of an end-to-end RL approach to solve ObjectNav navigation [18, 19, 21–24] [25, 26] . Some authors have proposed combining the RL training with different strategies, like auxiliary tasks [27], improved visual representations via object relation graphs [28], semantic segmentations [29] or combining audio feedback with the visual inputs [30, 31].

Modular-learning based approaches [3, 20, 32–34] [2, 35–38] decompose the navigation process in separate modules that execute different tasks. It is common for these methods to be composed of a high-level semantic exploration module trained by RL that indicates the agent subgoals that have to be reached by a low-level navigation policy. Modular learning can be also combined with offline RL [39] techniques to leverage navigation behaviors from fixed datasets, without any additional online data collection or fine-tuning.

Finally, there are different approaches that try to tackle the problem of rapidly adapting to unseen environments in visual navigation via meta-learning [40–42]. These methods are trained on a variety of different environments (usually designated as tasks) and are able to generalize to unseen environments by learning a policy that can be quickly adapted to new environments. And the recent progress in large language models (LLMs) has led to the possibility of using them to solve the visual navigation problem [43, 44] as well. In this case, the LLMs are used as a reasoning module in charge of understanding the semantic information present on the environment. They then share this information with different modules in charge of navigating to the specified goal.

Our goal in this work is not to develope VSN approaches, but to integrate various state-of-the-art VSN models into multiple real-world robots by using our novel ROS4VSN library. Technically, we have chosen to integrate the PIRLNav [1] and VLV [3] models into two different robots. These integrations required several technical adaptations, particularly in the areas of sensor data integration and navigation planning. Overall, we are able to show how ROS4VSN allows easily testing and comparing different VSN methods in the real world. To the best of our knowledge, our work is the first to develop a model agnostic ROS package for visual semantic navigation, where multiple models can be integrated.

Simulation-to-reality transfer in robotic navigation. Deploying a model trained in simulation to a real robot is a challenging task. Due to logistical constraints, training a model in the real world -especially with RL techniques- is often impractical, prompting the use of alternative methods to address this challenge For example, [45] propose a monocular vision-based time-to-collision estimation for small drones by domain adaptation of simulated images. Their method converts simulated images into real-like synthetic images using a sim-to-real method. This is done with the aim of minimizing efforts and time invested in the collection of training datasets within real-world scenarios, while simultaneously maximizing the advantages inherent in simulated environments.

Overall, it is necessary to develop methods that allow to efficiently transfer the knowledge learnt in simulation to the real world [46]. Different approaches have been proposed to solve this problem. For instance, CAD2RL [47] system achieved remarkable success in training a collision avoidance policy entirely within a simulated environment. This breakthrough was subsequently tested on real aerial drones, with promising results. By focusing on simulation refinement [48], the accuracy of simulations can be improved by exploiting the disparities between simulated and real-world observations. In the field of locomotion, training legged robotic systems in a simulated environment and subsequently transferring the acquired policies to real-world applications [49, 50] has always been a challenging task.

For the problem of VSN, we have the studiy by [51] that shows how their approaches perform in real-world settings. However, we would like to highlight the novel contributions that our work offers. First, while [51] focuses mainly on the comparison of their navigation methods, we here, along with a similar study, release to the research community the modular ROS4VSN software architecture. Our main goal is to facilitate the prototyping of new VSN solutions on real robots. So, we offer a ROS-compatible software architecture, model agnostic, that allows a simple integration of different VSN approaches in ROS robots. In this way, future VSN solutions will be able to be tested on real robots in a convenient and straightforward manner. Second, we include in our study more recent VSN solutions than the ones reported in [51], as the PIRLNav model [1], which defines the state-of-the-art for the ObjectNav problem. Third, we also provide, for the first time, a detailed analysis on how a model directly trained with real videos, such as the VLV [3], performs in real robots. This allows us to compare, as in [51], how a modular-learning model (*i.e* VLV [3]), compares with a typical end-to-end learning approach (*i.e* PIRLNav). Interestingly, our study also concludes, like in [51], that modular-learning approaches perform better in the real world. Fourth, in our study, we employed two different robotic platforms: one

commercially available and widely used by various laboratories, and another custom-built. This demonstrates the versatility of the proposed solution, showing that it can be integrated into different robots. And finally, in our work, we propose an experimental evaluation specifically designed for testing in the real world, which can be employed in future research studies. Overall, we hope that our ROS-based library will help to further advance the field of visual semantic navigation in real robots.

3 Methodology

In this research work, our main objective has been to efficiently integrate various state-of-the-art VSN models on multiple robotic platforms. To fulfill our goals, the first step of the proposed methodology has been to develop a ROS-based solution to ease the integration of VSN solutions into real robots. It is crucial that the approach is model-agnostic, allowing the integration of any VSN-designed model. We have named this development ROS4VSN, and it is detailed in Section 3.1. Once the ROS4VSN system is available, we need to select some state-of-the-art VSN models that will allow us to experimentally evaluate them with real robots. In Section 3.2, we justify the selected VSN models and detail the technical modifications made to integrate them into ROS4VSN.

3.1 ROS4VSN: ROS for visual semantic navigation

We have designed ROS4VSN library to be modular and flexible, so that it can be easily adapted to different robots and VSN models. It is built on top of ROS *Noetic* [5] open-source robotic middleware, because of its flexibility, support, compatibility and popularity among the robotics community. ROS provides a collection of useful tools, libraries, and conventions to simplify the task of creating complex and robust robot behaviors across a wide variety of robotic platforms, which makes it perfect for our framework. We design the architecture of the framework, so it has three main capabilities: it can receive and process information from the environment, infer actions using an AI VSN model, and control the actuators of a platform to reach a specific navigation goal. It makes it easy to integrate different VSN models, since it only needs to replace the model with which the experiments are to be carried out. The architecture is divided into the following main packages, each of which plays a specific role: robot_api, camera_api, discrete_move and visual_semantic_navigation. These packages are connected to each other through ROS topics and services, and also to external hardware devices: a camera (RGB + Depth) and a differential drive robotic platform. Figure 2 shows a visual representation of the global architecture scheme, illustrating the connections between the different developed ROS packages and the hardware devices.

3.1.1 Robot API

This package is responsible for controlling the actuators of the robot and sending odometry information to the discrete_move and visual_semantic_navigation packages (depicted in Fig. 2). It is typically designed by the manufacturer of the robot, so it can be different depending on the employed platform. In our particular case, the development of the framework and experiments were done using two robots (see Fig. 8). First, a Turtlebot 2 robot, so the standard turtlebot2 [52] package is integrated as robot_api. Since the Turtlebot 2 expects the velocity commands in the */mobile_base/commands/velocity* topic, we perform a remapping to the */cmd_vel* topic from our discrete_move package. Our second robot is known as LOLA2 [53]. We have developed its complete robot_api package to guarantee the compatibility with the rest of ROS4VSN architecture.

In charge of the communication with the platform, this package is also responsible for publishing the robot's odometry information through the topic */odom* (odometry topic). This information is crucial for the visual_semantic_navigation and discrete_move packages. On one hand, the package discrete_move uses this information to adjust the velocity commands sent to the robot_api package, achieving precise and controlled movements. On the other hand, the visual_semantic_navigation package can use the odometry information to help infer the action to be executed by the robot or to help a planner reach its destination.

3.1.2 Camera API

This package is responsible for capturing RGB and depth images from the camera. It publishes them through the */camera/color* and */camera/depth* topics, respectively. The camera used in our robots is the Orbecc Astra S, an RGB plus depth camera, based on structured light technology. We have adapted the official ros_astra_camera package [54] to be integrated in our ROS4VSN architecture. The modular design of ROS4VSN allows it to be used with any other type of camera on the robots, simply by adapting this Camera API package.

3.1.3 Discrete Move package

This package has been developed with the purpose of providing a precise and customizable control of the robotic platform through discrete navigation commands. Note that this is the way most VSN models interact with their agents (*e.g* [1, 3]). In embodied AI, navigation in simulated environments is performed using discrete action commands that agents exe-

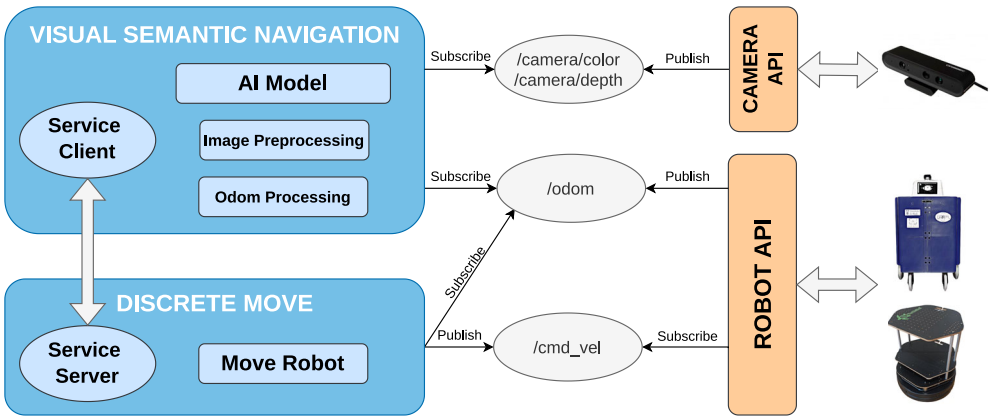


Fig. 2 Architecture scheme of the ROS44VSN framework. It shows the different packages, topics and connections within them and the hardware devices

cute to reach the specific goals: move forward (25 cm), turn left or right (30 degrees), or stop. Our ROS4VSN package acts as a server to which clients can request a set of discrete movements and configure the forward distance or turning angles. The package communications scheme is shown in detail Fig. 3. It is in charge of executing the actions requested by the visual_semantic_navigation package and sending a response when the action is completed.

Set of navigation movements

The set of movements allowed by the package consists of the following actions: TURN_LEFT, TURN_RIGHT, MOVE_FORWARD, MOVE_BACKWARD and STOP. All the actions are fully customizable in terms of distance and angle, except for the STOP action, which does not require any additional parameters since it just stops the robot. This package has been designed as a ROS service, so the communication between the visual_semantic_navigation package and the discrete_move package is done synchronously and bidirectionally. That way, the visual_semantic_navigation package can wait for the response of the discrete_move package when the action has been completed before sending any new action request. The discrete_move server is in charge of sending the right */cmd_vel* commands to the robot_api package, so the robot can execute the requested action and receive the */odometry* topic information from the robot_api package. That way, it can calculate the movement done by the robot, and stop the action when the requested action has been finished.

Embodied AI navigation environments, such as Habitat [55], are simulation environments where there are no movement errors in the agents. However, our scenario is the real world, with real robots. Therefore, ROS4VSN must integrate error control strategies. To achieve this, the discrete_move package includes two error correction strategies: one for the turn error ϵ_{turn} and one for the move straight error $\epsilon_{straight}$.

The turn error is calculated in degrees as follows,

$$\epsilon_{turn} = (\alpha_{target} - \alpha_{current}) \bmod (360) \ , \tag{1}$$

where α_{target} is the target orientation and $\alpha_{current}$ is the current orientation of the robot.

The move straight error is computed as:

$$\epsilon_{straight} = d - \sqrt{(x - x_{init})^2 + (y - y_{init})^2} \ , \tag{2}$$

where d is the displacement distance requested by the system, (x, y) encodes the current position of the robot and (x_{init}, y_{init}) defines the initial position of the robot. We consider a successful turn when ϵ_{turn} is less than 0.1 degrees and a successful displacement when $\epsilon_{straight}$ is less than 5 millimeters. Our ROS4VSN architecture continuously measures these errors to adapt the rotation and displacement movements of the robots, ensuring that they occur with the highest possible precision.

Acceleration and braking control

When developing a navigation system based on discrete commands, it is crucial to implement appropriate braking and acceleration mechanisms to achieve smooth and efficient robot navigation.

The package includes an implementation that combines a constant acceleration until the desired maximum speed is reached, with a deceleration phase to stop the robot.

The smoothness of the movement and the time needed to complete it depend on the percentage of the path in which the robot is accelerating and decelerating, as well as on the initial speed. By properly adjusting these parameters, a smoother movement and a more efficient navigation time can be achieved. Figure 4 illustrates the total distance that the robot must travel for a MOVE_FORWARD (or MOVE_BACKWARD) command. This distance is divided so

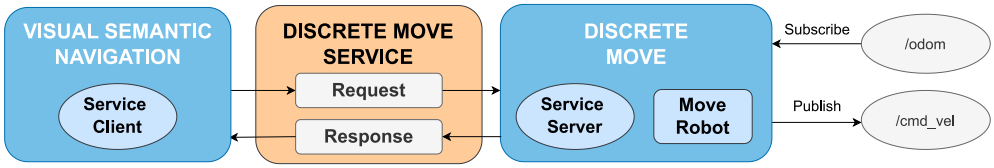


Fig. 3 Communications between visual_semantic_navigation and discrete_move packages

that the robot performs the following phases: acceleration, displacement at constant speed and deceleration. The instantaneous speed of the robot is controlled so that during the first phase, there is a uniformly accelerated motion, according to the following equation: $v = \sqrt{v_{init}^2 + 2a\epsilon_{straight}}$, where a is the desired acceleration, v_{init} represents the initial speed of the robot, and $\epsilon_{straight}$ encodes the distance covered by the robot. Note that we must continuously read $\epsilon_{straight}$ using our ROBOT API, in order to dynamically adjust the speed. For the deceleration stage, we employ an equivalent negative acceleration in the previous equation. With these equations, we can progressively adjust the linear speed (v) that is sent to the platform to obtain a smooth navigation.

Configuration parameters

The discrete move package includes the configuration file `discrete_move.yaml`, where the different parameters used for the execution can be modified. Table 1 shows the configuration parameters and its default value.

3.1.4 Visual semantic navigation package

One of the main features of our ROS4VSN software is that it allows to easily integrate different VSN models independently of the robotic platform used. To achieve this, it is essential to be agnostic with respect to the software environment needed by the particular VSN model, as each model may require different dependencies.

The goal of this package is to simplify the deployment of VSN models on real robots, providing an efficient software structure for the execution of these methods. In other words, it aims to facilitate the inference tasks of discrete movement actions that these systems produce, using the state of the robotic platform. The state is defined by the robot's position in the real world, the information provided by its sensors, and the action that was previously taken.

VSN models make decisions using mainly RGB images of the environment. However, some of them can also use additional information, such as the position and orientation of the robot, or even depth images. Therefore, this package must be responsible for: a) capturing all the information from the sensors and processing the obtained data; b) inferring with the VSN models the next navigation action; and c) communicating with the robotic platform to request the corresponding discrete motion. This process is repeated iteratively, collecting new data, making inferences and executing actions, until a stop condition is reached or an error occurs.

In our architecture, the package connects to the camera through the */camera* topic published by the camera_api module, to receive the necessary RGB and depth images. It also collects information from the robot's odometry through the */odometry* topic published by the robot_api package. Furthermore, it acts as a client of the discrete_move package to send the actions determined by the VSN model and receive confirmations about their execution. This package contains two additional submodules: image preprocessing and odom processing.

Image Preprocessing submodule This submodule is in charge of collecting and preprocessing the images from the camera of the robot. These images are necessary for the VSN model to infer the appropriate action. The package must be able to communicate with the camera in real time and receive its information. This communication is done through the */camera/color* and */camera/depth* ROS standard topics. Typ-

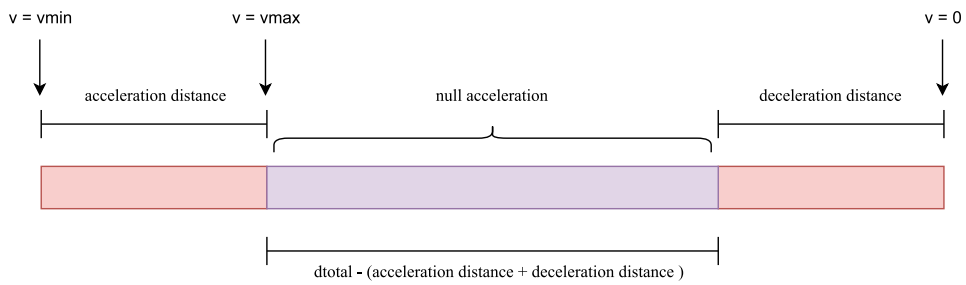


Fig. 4 Acceleration and braking control scheme

Table 1 Configuration parameters of discrete_move package

Parameter	Default Value
Linear Velocity	0.3 m/s
Angular Velocity	0.5 rad/s
Acceleration and deceleration distances	(MOVE_FORWARD distance) / 3

ically, depth images are taken using a time-of-flight camera. This type of camera can lead to noise problems, including incomplete data in certain areas of the image, noise on metallic surfaces, and the impact of scene lighting on distance measurements. To address these problems, a temporal median filter is implemented, so for a series of N depth images, its noise can be reduced by discarding outliers.

Odom Processing submodule This submodule is in charge of collecting odometry information by subscribing to the */odom* topic (*odometry topic*), published by the robot_api package. This odometry information consists of two main variables: 1) robot position, that indicates the current location of the robot with respect to its initial position; and 2) robot orientation, that defines the current direction in which the robot is oriented in relation to its initial orientation. Some state-of-the-art VSN models (*e.g* [1]) need to input these two sources of information.

Module workflow Our VSN package uses the image submodule and the odometry submodule to capture the camera images and the robot odometry data. This information can then be passed as input to a particular VSN model. Using the VSN model integrated in the package, the next navigation action that the robot must execute is inferred. Once the action has been determined, the package sends a message (*request*) to the discrete_move server to request the execution of the movement by the robot.

After sending the request to the discrete_move server, the package waits to receive a confirmation message. This message indicates whether the requested action has been performed correctly or if some problem has occurred. If for some reason an action has not been completed successfully, the server has been programmed to return $False$, which completely stops the execution of the workflow. It is important to highlight that the workflow is repeated until whether the STOP action is inferred by the model, the time limit for the episode is reached or the server responds with a message indicating some problem during the execution.

A configuration file is provided (`vsn.yaml`) containing default values for the parameters of our VSN package. By modifying this file, one can easily change the navigation target, the parameters associated with the median filter, or even the maximum number of steps allowed to be executed during a navigation exercise.

3.2 VSN models

For this research work, we have decided to adapt and integrate into our robots two state-of-the-art VSN models: PIRLNav [1] and VLV [3]. The first model, known as PIRLNav [1], is a VSN approach that has been trained with a combination of imitation learning and a RL fine-tuning. As of today, this model reports the best results in the OBJECTNAV [4] task in Habitat [55]. The second model is the VLV approach [3], which is a VSN model directly trained from YouTube videos. The VLV model makes use of such videos to learn semantic cues for an effective navigation to semantic targets in indoor scenarios. VLV is a modular learning solution that combines low-level and high-level navigation policies.

These models are complementary in the sense that they are based on two paradigms: a) imitation learning plus RL; and b) modular learning. This aspect will allow us to study, in the experimental evaluation, which type of approach yields better results in the real world. Note that we do not intend to retrain these models but rather subject them to evaluation in the real world. We aim to analyze their generalization ability for navigation outside simulation environments. It is precisely thanks to our ROS4VSN system that this can be done, as the technical modifications made to the models will be oriented towards embedding them in a ROS-based system. Next, we provide a detailed description of the modifications and adaptations made to these models so that they can be integrated into ROS4VSN and tested on real robotic platforms.

3.2.1 VLV

The first approach is known as Value Learning from Videos (VLV), developed by [3]. VLV is a modular-learning based VSN model directly trained from videos of real state agencies, taken from YouTube. In this type of video, a human records, camera in hand, the properties for sale, showcasing all the rooms they have, to generate a sort of virtual tour of the houses. Note that the videos used do not have any type of information about the navigation actions that take place during the recording, nor in what kind of rooms or what type of objects appear.

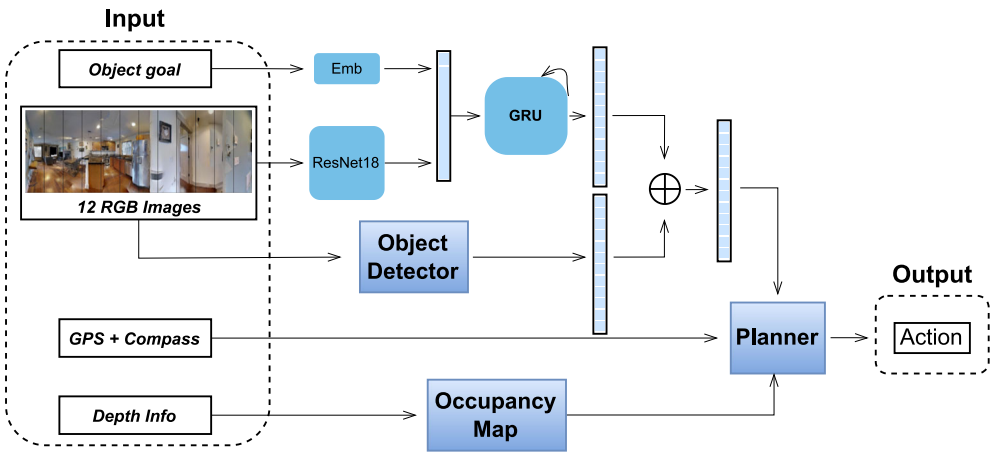
The VLV model leverages such YouTube videos to learn semantic cues for an effective navigation to semantic tar-

gets in indoor home environments. This way, the VLV model is trained to find in these videos a set of object categories. Technically, the model uses pseudo action labels obtained by running an inverse model on the navigation sequences. This inverse model is able to recognize the discrete movements that each of the transitions of the different video frames involve. Then, the navigation policies are learned following a reinforcement learning (RL) approach. VLV employs Q-learning to learn from the video sequences that have been pseudo-labeled with the actions. The learned Q-function, and the associated value function, implicitly learn semantic cues for navigation. In other words, the model learns what images lead to the desired category and what do not.

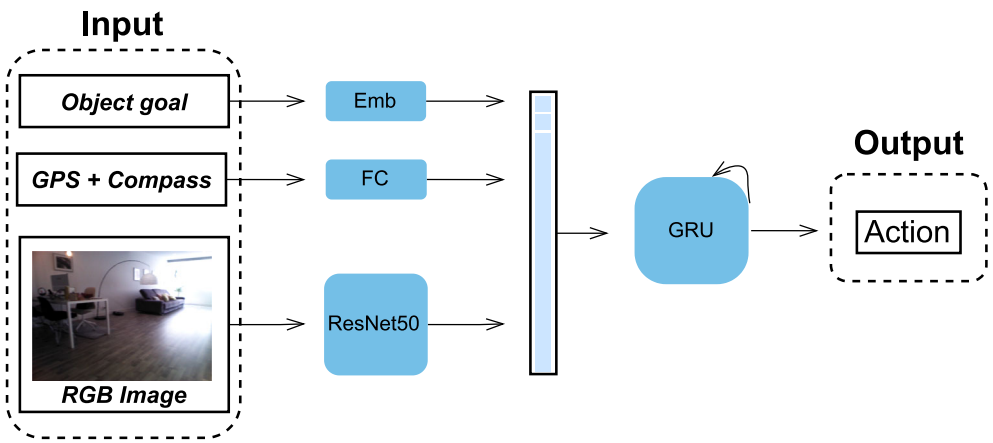
For our experiments, we had to embed the VLV model in our ROS4VSN architecture. See Fig. 5a. Technically, we integrated the two navigation policies detailed in the experiments in [3] that were tested in the virtual environment Habitat [55]. However, now, our goal is to implement them in a real robot in the real world.

This integration into our robots, using our ROS4VSN architecture, has consisted of the following steps. First, a high-level policy that stores 12 images for each node in a topological graph (obtained by rotating 12 times by 30 degrees each) is used. This high-level policy uses the learned value function score over these 12 images, and samples the most promising direction for seeking objects of a particular object category. VLV needs an object detector output to produce the final score for these images. This way, the high-level policy is equipped with a mechanism to seek the object once it has been detected. Specifically, the detector we employ is the Mask R-CNN [56], which we had to embed in our architecture as well. To navigate, this policy will use the following discrete movements/actions: MOVE_FORWARD 25cm, TURN_RIGHT 30° or TURN_LEFT 30°.

These movements are compatible with the developed discrete_move package of our ROS4VSN architecture. Once a main direction has been chosen, our approach converts it into a short-term goal by sampling a location at an offset of



(a) VLV model.



(b) PIRLNav model.

Fig. 5 VSN models integrated into our VSN-ROS

1.5 meters from the chosen node's location, in the chosen view's direction. This is done using the depth-camera and a low-level navigation policy that uses occupancy maps with a fast marching planning [57] to execute robot actions to reach the short-term goal. These two policies have been integrated into our ROS4VSN nodes, and with them, we get the robot to explore the environment.

3.2.2 PIRLNav

The second model selected for our experimental evaluation is known as PIRLNAV [1]. As of today, this model reports the best performance in the OBJECTNAV [4] task in Habitat [55].

PILRNav is a VSN approach that has been trained with a combination of imitation learning and a RL fine-tuning. The model uses behavior cloning (BC) to pre-train the OBJECTNAV policy on a dataset of 77k human demonstrations, amounting more than 2370 human annotation hours, in the HM3D-Semantics v0.1 dataset [58]. This dataset provides up to 120 different 3D reconstructions of houses all around the world (see Fig. 6 for an example of one of them). Once this BC is finished, a RL fine-tuning is used following the DD-PPO approach [59]. The policy architecture used is a simple CNN plus an RNN from [17].

In order to integrate PIRLNav into our ROS4VSN architecture, we had to perform the following actions. See Fig. 5b. The original PIRLNav needs to receive as inputs the RGB image that the agent observes, as well as the noiseless GPS and compass information offered by Habitat simulator. GPS and compass Habitat sensor provide the agent's current location and orientation information relative to the start of the episode. In our case, because PIRLNav has to be integrated in a real robot, navigating in the real world, we proceed to feed the model with the RGB images that are acquired by the cameras in our robotic platforms. GPS information is obtained through the odometry information provided by the robot. For the compass, we recover the relative orientation analyzing all the robot's turning movements. Note that these are not anymore noiseless sensors, as the ones used in the simulated world in which the PIRLNav model was trained. Fortunately, we did not observe any important impact on the performance of the model, due to this loss of precision for these sensors.

PIRLNav is therefore integrated in the ROS4VSN architecture to control de navigation of the robot as it has

Fig. 6 One of the 3D reconstructions of the HM3D-Semantics v0.1 dataset [58], used to train PILRNav

been detailed. For every captured image, as well as the GPS+compass data, the model is able to determine the next discrete movement action to be executed by our robotic platforms. The action space used for our experiments with this model is: MOVE_FORWARD 25 cm, MOVE_BACKWARD 25 cm, TURN_RIGHT 30°, TURN_LEFT 30° and STOP. The original PIRLNav was also trained to produce the discrete action LOOK_UP and LOOK_DOWN, since the simulated agent could tilt its camera. However, as it is observed in Fig. 8, in our platforms these actions are not possible. We decided to replace LOOK_UP with a MOVE_BACKWARD action, and LOOK_DOWN movement with the MOVE_FORWARD action. This choice is based on the reasoning that, by raising the camera, more of the scene is captured; moving the robot backward serves this purpose. Also, since lowering the camera provides a greater level of scene detail, moving forward is considered the most appropriate choice to replace the LOOK_DOWN action. Finally, to prevent collisions between the robot and objects in the scene, a procedure was developed that uses information from the depth image to detect obstacles at a given distance. Note that PIRLNav does not need any low-level policy as in the VLV model. The robot is controlled and navigates using only the set of discrete actions provided by the ROS4VSN model.

4 Experiments

This section describes the experimental evaluation designed for testing our developments in the real world. The goal of our experimental evaluation is to answer the following question: Are the state-of-the-art VSN models able to successfully operate with real robots? We start with a detailed description of the experimental setup, where the experimental conditions and the evaluation metric are explained. We then follow with an analysis of the results obtained in the real world.

4.1 Experimental setup

One of the primary objectives of our work is to provide a comprehensive and clear protocol for the experimental evaluation of state-of-the-art VSN models in real-world scenarios using real robots. Our goal is for other researchers to carry out similar experimental evaluations, using the same evaluation metrics, to facilitate comparisons of how different VSN models perform in real-world navigation tasks. For doing so, we propose the following experimental setup.

In a 75 m^2 apartment, we define up to 15 random starting positions (see Fig. 7). The apartment can be divided into three main areas: a bedroom, a bathroom, and a larger space that includes the kitchen, living room, and study area. This setting contains all the object categories used to train the VSN models in the experiments, such as chair, bed, plant, bathroom, monitor, table, and sofa. We encourage other researchers to conduct their experiments in real-world settings that are similar in size and characteristics, allowing the robot to navigate from at least 15 different starting positions across multiple instances.

From these positions, the robot is tasked with navigating to various object categories. Consequently, one must conduct 15 navigation experiments for each target category and measure the success of the episodes based on whether the robot reaches the designated object category in fewer than 150 discrete actions and without any collisions. This limit of 150 steps was chosen to establish a balance between the average size of the houses typically used in Habitat [55] and the apartment used in our experiments. For the evaluation metric, we propose reporting the success rate (SR) of the VSN models as the percentage of episodes in which navigation is deemed successful. An experiment is considered successful if the robot halts (when the VSN model samples the action STOP) and the Euclidean distance to the target object is less than one meter. Note that our navigation experiment mimics the evaluation performed in the OBJECTNAV [4] task, with the same metric. This is the standard experiment on which most VSN models are compared and which currently defines the state of the art.

We have used two different robots for our experiments: a Turtlebot 2, and the LOLA2 [53] platform. To do so, we had to embed our ROS4VSN in both of these platforms. This can be easily done by adapting the robot_api module described in

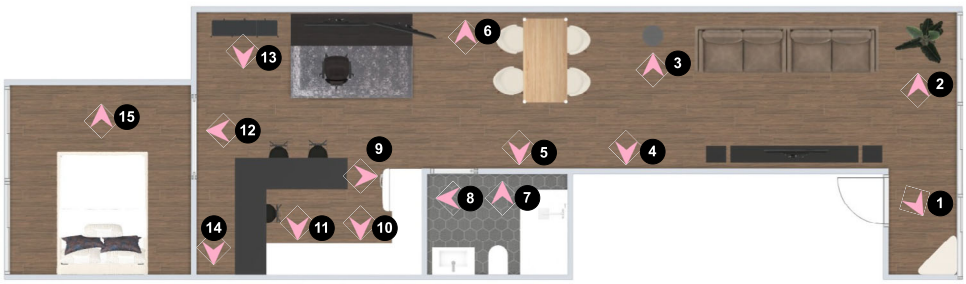


Fig. 7 Floor plan where the experiments were performed, indicating the 15 starting positions used

(a) Turtlebot 2. (b) LOLA2 platform.

Fig. 8 Pictures of the robots used in our experiments

Section 3.1.1. Figure 8 shows the family picture of the robots invited to our experiments. We mainly used the Turtlebot 2 for the navigation experiments in the apartment described. The robot LOLA2 was also used in navigation experiments, but in a different location, to test the stability of the developed system and to provide a study that contains more hours of navigation, and on different platforms. Our intention in using two different platforms has been to provide evidence of the generalizability of our architecture, demonstrating that it can be tested on different robots.

When collecting information during the experiments, we developed a procedure to record relevant data for each trial. This procedure stores the unique identifier for each episode, the sequence of actions performed, and the category of object searched. In addition, during the tests, qualitative information about the trajectory followed by the robot was recorded. In particular, all the images observed by the robot during its trajectories have been saved.

Table 2 Real world success rate against simulation

Models	SR (Real World)	SR (Virtual Environment)
VLV [3]	29.33%	39%
PIRLNav [1]	21.11%	65%

For the experiments, the following hardware-software setup has been used, as it is shown in Fig. 9. The modular architecture of the developed ROS4VSN system was used to deploy it in a distributed manner. This architecture allows separating the execution of ROS packages on different devices, as long as they are connected to the same network. The robotic platforms were equipped with a laptop. This device was used to establish the communication with the robotic platform and the camera by executing the robot_api and camera_api packages. At the same time, the discrete_move package was executed to receive the actions to be executed by the robot. On the other hand, we used a workstation to run the VSN nodes, which was equipped with an i7-1165G7 processor and an NVIDIA GeForce RTX 2060 graphics card. We provide code to reproduce all our experiments at https://github.com/gramuah/ros4vsn.

4.2 VSN navigation results

We detail in this section the main results obtained during the navigation experiments with our robots, including both quantitative and qualitative results.

Following the experimental setup detailed in Section 4.1, we have obtained the following results. Remember that we provide in this study an analysis of the SR for two state-of-the-art VSN models, running in a Turtlebot 2 platform. For every VSN model, *i.e* VLV, and PIRLNav, we have measured their SR for the different object categories they can navigate to.

Table 2 compares the performance obtained by VLV and PIRLNav approaches when they are tested in the real world (*i.e* our experiments) and in a virtual environment (*i.e* the experiments reported in their respective papers). The first thing we observe is the difference in terms of SR. The SR for

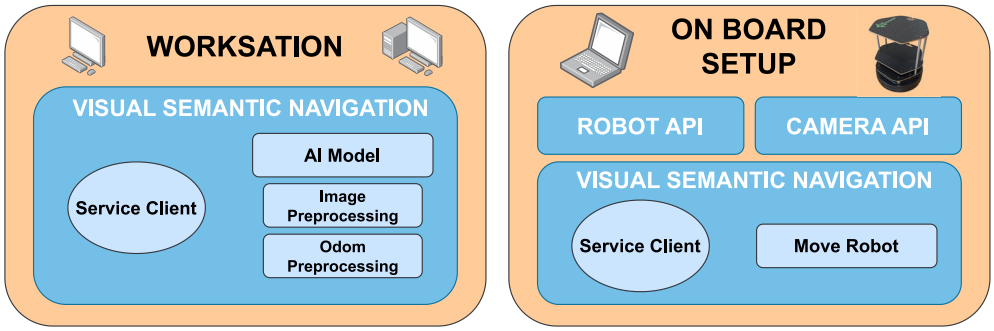


Fig. 9 Hardware-software architecture for the development of experiments

Table 3 VLV VSN experiment

Object Goal	Successful episodes	SR	Avg. number of actions
Chair	6/15	40%	30
Sofa	6/15	40%	65
Table	6/15	40%	42
Bed	3/15	20%	39
Toilet	1/15	6,67%	42

We report the number or successful episodes over 15, the corresponding SR per-object goal, and the average number of actions taken to reach the target

the PIRLNav model drops from 65% to 21%, while the VLV model loses ∼ 10 percentage points in this metric. One of the conclusions of our study is that there is a considerable gap between the behavior of these models in the real world and in the simulation environments in which they are trained. This indicates that further research in this direction is needed. Interestingly, the results obtained in our real-world experiments are not consistent with the performance difference that already existed between the models in the simulation environments: VLV is the winner in the real world! As we analyze in the discussion section below (See Section 4.4), we believe that this behavior is due to the impact of the object detector that VLV integrates, but PIRLNav does not. While the difference between VLV and PIRLNav SR in the virtual environments is of 26 percentage points, in the real world this gap becomes of just 8 percentage points.

In the following, we analyze in detail the results reported by each of the models. We start with the VLV model. Table 3 reports the SR for every of the target categories used in our experiments. Chairs, tables, and sofas are the categories that are easiest to navigate to. In analyzing various trials with the robot using the VLV model with ROS4VSN, there were no successful outcomes from starting positions 10 and 12 (see Fig. 7). Additionally, only one success was observed from positions 1, 2, and 13. Notably, our VLV implementation can reach most targets in under 60 steps.

Figure 10 shows qualitative results for four navigation experiments with the VLV model. We provide five representative images of the navigation experiments. Two successful and two unsuccessful cases are presented. In the first experiment (first row), the robot quickly reached the Table, as the detector easily identified it in the images. In the second experiment, the robot took a detour to the Chair because the detector failed to detect it initially. The model predicted a point near the chair but out of view, causing the robot to move closer only after it became visible. In the third and fourth experiments, the robot started in a challenging position in front of a refrigerator. Due to noise in the depth image, the target point calculation was inaccurate, leading to collisions with the wall in both cases.

We analyze now in detail the results reported by PIRLNav model. Table 4 shows the SR obtained for the PIRLNav model integrated into our ROS4VSN architecture. We detail the SR reported for every object category. The agent was able to more easily locate the most common objects in the house, such as the chair and the monitor. The abundant and well-

Fig. 10 VLV qualitative navigation results. The first two rows show two successful cases, where the robot reached the target, while the last two rows show two situations where the navigation experiment failed

Table 4 PIRLNav VSN experiment

Object Goal	Successful episodes	SR	Avg. number of actions
Chair	5/15	33,33%	49
Monitor	5/15	33,33%	91
Sofa	5/15	33,33%	70
Bed	3/15	20,00%	97
Toilet	1/15	6,67%	61
Plant	0/15	0,00%	82

We report the number or successful episodes over 15, the corresponding SR per-object goal, and the average number of actions taken to reach the target

distributed presence of these objects facilitated the agent's work. The model inferred the action STOP on both objects a total of five times. Large objects, such as the sofa and bed, showed slightly lower results. Although easily visible from multiple locations in the dwelling, the presence of only one of these objects made it difficult for the robotic agent to spot them. The number of times the STOP action was sampled for these categories was substantially reduced. With the toilet, we have one of the most complex challenges in navigating the robot in this home. The robot did not have full visibility of the toilet until it managed to fully enter the bathroom, having to pass through the narrow door without colliding. Finally, for the category plant, the robot was not able to locate this category in any of the 15 attempts. Even though the plant was visible on multiple occasions during navigation, the agent did not manage to head towards this object. Overall, considering all the categories, the SR for the PIRLNav model is of 21.11%.

To conclude our analysis of the PIRLNav model, we provide some qualitative results. Figure 11 shows the navigation trajectories for four different experiments. We provide five representative images of the navigation experiments. Two successful and two unsuccessful cases are presented.

In the first experiment, the robot starts from the refrigerator and navigates through the house until it reaches the sofa. This episode is carried out in 69 actions and ends when the model infers the action STOP in front of the sofa. The second experiment is also a success story, but this time the robot starts navigating from the kitchen. The robot leaves the kitchen and navigates to the nearest chair. This episode is performed in 36 actions and ends with the STOP action determined by the model. The third experiment shows a case of navigation failure, where the robot targeting the plant hits an obstacle. In this episode, the robot navigates for 61 actions until it hits the couch. Despite visualizing the plant from far away, when trying to approach it, the robot ends up crashing. In the last episode, another case of navigation failure is shown while the robot was trying to make its way to the bed. As it can be seen in this episode, the model had difficulty getting through the bathroom door without hitting itself.

Fig. 11 PIRLNav qualitative navigation results. The first two rows show two successful cases, where the robot reached the target, while the last two rows show two situations where the navigation experiment failed

Table 5 Time spent and traveled distance for both robots during the experiments

Robot	Time (Hours)	Distance (km)
LOLA2	8	1.12
Turtlebot 2	30	4.10
Total	38	5.22

We provide a video (see https://youtu.be/nD0JBWNCMGg) with more qualitative results for both of the VSN models used in our experiments.

4.3 Stability analysis

In this section, we analyze the stability of the navigation solution proposed, showing how it can robustly navigate a considerable distance, in two different robots and over two different scenarios. The robots successfully navigated over more than 5 kilometers in less than 38 hours in two dynamic environments. The robots operated without direct assistance throughout the experiments, being automatically operated by the VSN models integrated into our ROS4VSN software architecture (Table 5).

4.4 Discussion

The main question we wanted to address with the designed experiment has been: are the state-of-the-art VSN models able to successfully operate with real robots? This implies knowing the SR that these models are capable of delivering when tested in the real world, and not in the virtual environments where they were trained. Note that we selected two VSN models that were originally trained with images of the real world. Our intention was to reduce as much as possible the influence of domain shift, which we know affects artificial intelligence systems. Our study confirms that there is still room for improvement so that these models can achieve the same SR in real robots. We expect, therefore, that our ROS4VSN library plays a fundamental role in this line of research.

Analyzing the particular behavior of the models, we can provide the following interesting discussion. The integration of the VLV and PIRLNav models within our ROS4VSN architecture has proven to be successful. It has resulted in a mobile agent capable of navigating in closed environments autonomously, obtaining many experiments where the robotic platforms reach the target class without complications and following logical and direct trajectories. This navigation is comparable to that observed within simulated environments. For the VLV model that integrates an object detector, we have observed that this fact has a significant impact on the agent's navigation, especially when it is close to the target class. Although the object detector does not significantly affect the general exploration, its impact becomes crucial when the robot is in the vicinity of the target. At this crucial stage of navigation, the object detector provides a significant advantage by guiding the robot to the target more effectively. This explains the difference of performance we have observed between VLV and PIRLNav in the real world.

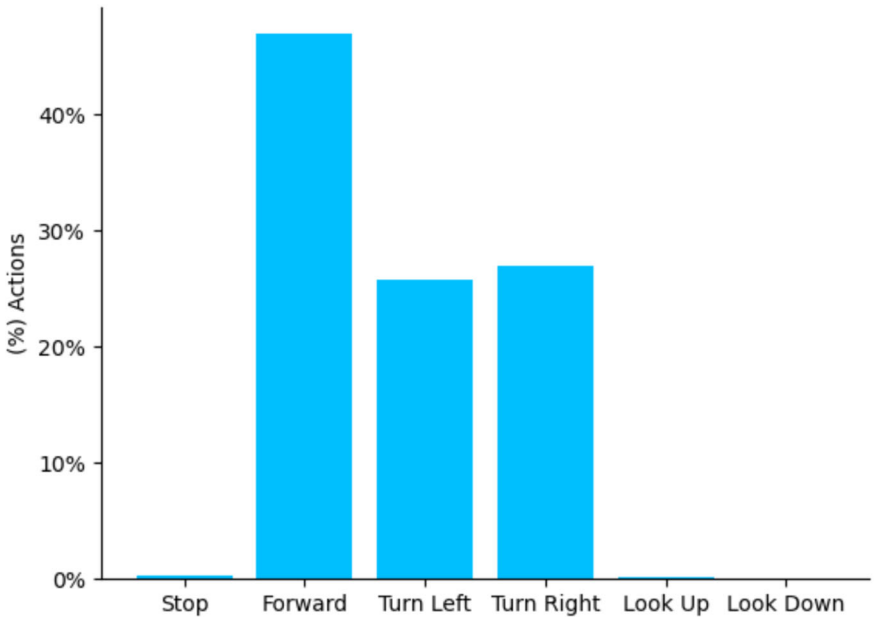


Fig. 12 Histogram of navigation actions sampled by PIRLNav model

In terms of qualitative aspects of navigation, we believe that the PIRLNav model is better than the VLV model. Note that VLV, every time the high-level policy has to make a decision, needs the robot to turn completely on itself, taking 12 captures on which it will decide which direction to move forward. This can be observed in the provided video. This feature slows down navigation, although it could be solved with some specific hardware. In contrast, PIRLNav offers a more direct navigation experience. It is interesting to observe the type of action sampling that PIRLNav performs while being executed on our robots. Figure 12 shows a histogram corresponding to the distribution of navigation actions performed by PIRLNav. First, one can observe that the LOOK_UP and LOOK_DOWN actions have hardly been selected. This allows us to affirm that the impact of the adaptations we have made to replace these actions by backward and forward movements, respectively, could hardly have had a considerable impact on the final results. Second, the most popular actions, as they are the ones that motivate the exploration of the environment, are those of advances and turns. The stop action was sampled 31 times. This is a 0.2% as it is reflected in the histogram provided. We believe that work should be done on solutions to increase the number of times the stop action is selected, but to do so reliably.

Finally, our study confirms some of the conclusions reported in recent works, *e.g* [51]. Modular-learning models, such as VLV, perform better than end-to-end learning approaches, *e.g* PIRLNav, when tested in the real world.

5 Conclusions

To conclude, we have presented a ROS-based framework for visual semantic navigation named ROS4VSN that allows to easily test and compare different VSN models in real robots. Using ROS4VSN, we have been able to embed two cutting-edge VSN models into two distinct real robotic platforms. The chosen models are PIRLNav [1] and VLV [3]. To seamlessly integrate these models, technical modifications have been needed. These adaptations ensure a smooth transition for the models, enabling them to shift from interacting with observations generated in simulation environments to those obtained from the real world. We have also offered a thorough experimental evaluation to showcase how these VSN approaches behave when navigating in the real world. Our novel framework shows a robust stability, being able to run for a considerable distance, in two different robots, without any human intervention. Our study and results show that the performance of state-of-the-art VSN models is significantly lower in the real world than in the virtual environments where they were trained. We expect that our efforts will lay the foundation for addressing this significant challenge.

Funding This research was partially funded by projects: NAVISOCIAL, with reference 2023/00405/001 from the University of Alcalá; NAVIGATOR-D, with reference PID2023-148310OB-I00 from the Ministry of Science and Innovation of Spain.

Statements and Declarations

Conflicts of Interest The authors have no relevant financial or non-financial interests to disclose.

Publication Status This research is already publicly available on arXiv (https://arxiv.org/abs/2311.16623) as a pre-print.

References

1. Ramrakhya R, Batra D, Wijmans E, Das A (2023) PIRLNav: pretraining with imitation and rl finetuning for ObjectNav. In: CVPR
2. Cai W, Wang T, Cheng G, Xu L, Sun C (2024) DGMem: Learning visual navigation policy without any labels by dynamic graph memory. Appl Intell
3. Chang M, Gupta A, Gupta S (2020) Semantic visual navigation by watching youtube videos. In: NeurIPS
4. Batra D, Gokaslan A, Kembhavi A, Maksymets O, Mottaghi R, Savva M, Toshev A, Wijmans E (2020) ObjectNav revisited: on evaluation of embodied agents navigating to objects. In: arXiv. arXiv:2006.13171
5. Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng A (2009) ROS: an open-source robot operating system. In: ICRA, Workshop on Open Source Robotics
6. Newcombe RA, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison AJ, Kohi P, Shotton J, Hodges S, Fitzgibbon A (2011) KinectFusion: Real-time dense surface mapping and tracking. In: International symposium on mixed and augmented reality
7. Thrun S, Fox D, Burgard W, Dellaert F (2001) Robust monte carlo localization for mobile robots. Artif Intell 128(1):99–141. https://doi.org/10.1016/S0004-3702(01)00069-8
8. Jones ES, Soatto S (2011) Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. Int J Robot Res 30(4):407–430. https://doi.org/10.1177/0278364910388963
9. Sattler T, Maddern W, Toft C, Torii A, Hammarstrand L, Stenborg E, Safari D, Okutomi M, Pollefeys M, Sivic J, Kahl F, Pajdla T (2018) Benchmarking 6dof outdoor visual localization in changing conditions. In: CVPR
10. Abaspur Kazerouni I, Fitzgerald L, Dooly G, Toal D (2022) A survey of state-of-the-art on visual slam. Expert Syst Appl 205:117734
11. Campos C, Elvira R, Rodríguez JJG, M Montiel JM, D Tardós J (2021) Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Transactions on Robotics
12. Labbé M, Michaud F (2022) Multi-session visual slam for illumination-invariant re-localization in indoor environments. Frontiers in Robotics and AI
13. Zhang L, Wei L-Y, Shen P, Wei W, Zhu G, Song J (2018) Semantic slam based on object detection and improved octomap. IEEE Access

14. Rosinol A, Abate M, Chang Y, Carlone L (2020) Kimera: an open-source library for real-time metric-semantic localization and mapping. ICRA
15. Jin S, Dai X, Meng Q (2023) focusing on the right regions - guided saliency prediction for visual slam. Expert Syst Appl 213:119068
16. Ramrakhya R, Undersander E, Batra D, Das A (2022) Habitat-Web : learning embodied object-search strategies from human demonstrations at scale. In: CVPR
17. Yadav K, Ramrakhya R, Majumdar A, Berges V-P, Kuhar S, Batra D, Baevski A, Maksymets O (2023) Offline visual representation learning for embodied navigation. In: ICLR
18. Gutiérrez-Maestro E, López-Sastre R.J, Maldonado-Bascón S (2019) Collision anticipation via deep reinforcement learning for visual navigation. In: IbPRIA
19. Khandelwal A, Weihs L, Mottaghi R, Kembhavi A (2022) Simple but effective: CLIP embeddings for embodied AI. In: CVPR
20. Chaplot DS, Gandhi D, Gupta A, Salakhutdinov R (2020) Object goal navigation using goal-oriented semantic exploration. In: NeurIPS
21. Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L, Farhadi A (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: ICLR
22. Wijmans E, Kadian A, Morcos A, Lee S, Essa I, Parikh D, Savva M, Batra D (2020) DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In: ICLR
23. Liu X, Guo D, Liu H, Sun F (2022) Multi-agent embodied visual semantic navigation with scene prior knowledge. IEEE Rob Autom Lett 7(2):3154–3161. https://doi.org/10.1109/LRA.2022.3145964
24. Yadav K, Majumdar A, Ramrakhya R, Yokoyama N, Baevski A, Kira Z, Maksymets O, Batra D (2023) OVRL-V2: A simple state-of-art baseline for imagenav and objectnav. ArXiv
25. Xu D, Chen P, Zhou X, Wang Y, Tan G (2024) Deep reinforcement learning based mapless navigation for industrial AMRs: advancements in generalization via potential risk state augmentation. Appl Intell
26. Yokoyama N, Ramrakhya R, Das A, Batra D, Ha S (2024) HM3D-OVON : A dataset and benchmark for open-vocabulary object goal navigation. IROS
27. Ye J, Batra D, Das A, Wijmans E (2021) Auxiliary tasks and exploration enable ObjectGoal navigation. In: ICCV
28. Yang W, Wang X, Farhadi A, Gupta A.K, Mottaghi R (2018) Visual semantic navigation using scene priors. ICLR
29. Mousavian A, Toshev A, Fiser M, Kosecka J, Davidson J (2018) Visual representations for semantic target driven navigation. ICRA
30. Wang H, Wang Y, Zhong F, Wu M, Zhang J, Wang Y, Dong H (2023) Learning semantic-agnostic and spatial-aware representation for generalizable visual-audio navigation. IEEE Rob Autom Lett
31. Kondoh H, Kanezaki A (2023) Multi-goal audio-visual navigation using sound direction map. ArXiv
32. Staroverov A, Muravyev K, Yakovlev K, Panov AI (2023) Skill fusion in hybrid robotic framework for visual object goal navigation, vol 12. https://doi.org/10.3390/robotics12040104. https://www.mdpi.com/2218-6581/12/4/104
33. Li Z, Zhou A (2023) RDDRL: a recurrent deduction deep reinforcement learning model for multimodal vision-robot navigation. Appl Intell 53:23244–23270
34. Zhou K, Guo C, Zhang H (2022) Improving indoor visual navigation generalization with scene priors and markov relational reasoning. Appl Intell 52(15):17600–17613
35. Kang J, Chen B, Zhong P, Yang H, Sheng Y, Wang J (2024) HSPNav: Hierarchical scene prior learning for visual semantic navigation towards real settings. ICRA
36. Wang J, Soh H (2024) Probable object location (polo) score estimation for efficient object goal navigation. ICRA
37. Wasserman J, Chowdhary G, Gupta A, Jain U (2024) Exploitation-guided exploration for semantic embodied navigation. ICRA
38. Yokoyama N, Ha S, Batra D, Wang J, Bucher B (2024) VLFM: Vision-language frontier maps for zero-shot semantic navigation. ICRA
39. Shah D, Bhorkar A, Leen H, Kostrikov I, Rhinehart N, Levine S (2022) Offline reinforcement learning for visual navigation. In: CoRL
40. Wortsman M, Ehsani K, Rastegari M, Farhadi A, Mottaghi R (2019) Learning to Learn How to Learn: self-adaptive visual navigation using meta-learning. CVPR, pp 6743–6752. https://doi.org/10.1109/cvpr.2019.00691
41. Luo Q, Sorokin M, Ha S (2021) A few shot adaptation of visual navigation skills to new observations using meta-learning. In: ICRA, pp 13231–13237
42. Zhang S, Li W, Song X, Bai Y, Jiang S (2022 Generative meta-adversarial network for unseen object navigation. In: ECCV)
43. Huang W, Xia F, Shah D, Driess D, Zeng A, Lu Y, Florence PR, Mordatch I, Levine S, Hausman K, Ichter B (2023) Grounded decoding: Guiding text generation with grounded models for robot control. ArXiv
44. Zhou K-Q, Zheng K, Pryor C, Shen Y, Jin H, Getoor L, Wang XE (2023) ESC: Exploration with soft commonsense constraints for zero-shot object navigation. ArXiv)
45. Kim M, Ladosz P, Oh H (2022) Monocular vision-based time-to-collision estimation for small drones by domain adaptation of simulated images. Expert Syst Appl 199:116973
46. Kadian A, Truong J, Gokaslan A, Clegg A, Wijmans E, Lee S, Savva M, Chernova S, Batra D (2020) Sim2Real predictivity: Does evaluation in simulation predict real-world performance? IEEE Rob Autom Lett
47. Sadeghi F, Levine S (2017) CAD2RL: Real single-image flight without a single real image. In: Robotics: science and systems
48. Son D, Yang H, Lee D (2020) Sim-to-real transfer of bolting tasks with tight tolerance. In: IROS
49. Hwangbo J, Lee J, Dosovitskiy A, Bellicoso D, Tsounis V, Koltun V, Hutter M (2019) Learning agile and dynamic motor skills for legged robots. Science Robotics
50. Agarwal A, Kumar A, Malik J, Pathak D (2022) Legged locomotion in challenging terrains using egocentric vision. In: CoRL
51. Gervet T, Chintala S, Batra D, Malik J, Chaplot DS (2022) Navigating to Objects in the Real World. Sci Rob
52. Ltd. K (2023) ROS wrapper for Kobuki base Turtlebot 2. https://github.com/yujinrobot/kobuki.git
53. Nasri N, López-Sastre RJ, Pacheco-da-Costa S, Fernández-Munilla I, Gutiérrez-Álvarez C, Pousada-García T, Acevedo-Rodríguez FJ, Maldonado-Bascón S (2022) Assistive robot with an ai-based application for the reinforcement of activities of daily living: Technical validation with users affected by neurodevelopmental disorders. Applied Sciences
54. Ltd O (2023) ROS wrapper for Astra camera. https://github.com/orbbec/ros_astra_camera
55. Szot A, Clegg A, Undersander E, Wijmans E, Zhao Y, Turner J, Maestre N, Mukadam M, Chaplot DS, Maksymets O, Gokaslan A, Vondruš V, Dharur S, Meier F, Galuba W, Chang A, Kira Z, Koltun V, Malik J, Savva M, Batra D (2021) Habitat 2.0: Training home assistants to rearrange their habitat. In: NeurIPS
56. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: ICCV
57. Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. In: Proceedings of the National Academy of Sciences

58. Ramakrishnan SK, Gokaslan A, Wijmans E, Maksymets O, Clegg A, Turner J, Undersander E, Galuba W, Westbury A, Chang AX, Savva M, Zhao Y, Batra D (2021) Habitat-Matterport 3D Dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In: NeurIPS
59. Wijmans E, Kadian A, Morcos AS, Lee S, Essa I, Parikh D, Savva M, Batra D (2019) DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In: ICLR

Carlos Gutiérrez-Álvarez received the B.S degree of Physics in 2018 and the M.S degree of Mathematical Engineering in 2020, both from the Complutense University of Madrid. In 2021 he joined the Signal Theory and Communications Department of the University of Alcalá, where he is currently pursuing his Ph.D. His research interests include, dense video captioning, reinforcement learning, robotics and computer vision.

Pablo Ríos-Navarro received his B.S. degree in Telecommunication Technologies Engineering in 2020 from Alcalá University and his M.S. degree in Computer Vision in 2023 from Rey Juan Carlos University. In 2023, he joined the Department of Computing and Statistics at Rey Juan Carlos University, where he is currently pursuing his Ph.D. His research interests include machine learning, computer graphics and computer vision.

Rafael Flor-Rodríguez-Rabadán received the B.S. degree in Telecommunication Technologies Engineering in 2021 from the University of Alcalá, where he is currently pursuing the habilitating M.S. degree in Telecommunications Engineering. In 2022, he was awarded the Spanish Ministry Collaboration Grant to conduct research on semantic visual navigation using semantic segmentation in deep learning applications.

Francisco Javier Acevedo-Rodríguez received the M.Sc. degree in electronic engineering from the Univesity of Alcalá, in 1998, and the Ph.D. degree from the University of Alcalá, in 2009. He currently leads the Department of Signal Theory and Communications, University of Alcalá. His research interests are in the field of pattern recognition and signal processing, especially on those projects in the electrochemical field and those dedicated to people with special needs.

Roberto Javier López-Sastre received a Master of Electrical Engineering from the University of Alcalá, Spain in 2005. He is Associate Professor at the Department of Signal Theory and Communications of the University of Alcaláa. He serves as program committee member/reviewer of the major computer vision conferences ICCV, ECCV, and CVPR. His current research interests include action detection, advanced video processing models, continual learning, reinforcement learning and robotics.