Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Robot@VirtualHome, an ecosystem of virtual environments and tools for realistic indoor robotic simulation

David Fernandez-Chaves [a,b,*], Jose-Raul Ruiz-Sarmiento [a], Alberto Jaenal [a], Nicolai Petkov [b], Javier Gonzalez-Jimenez [a]

[a] *Machine Perception and Intelligent Robotics group (MAPIR), Department of System Engineering and Automation, Biomedical Research Institute of Malaga (IBIMA), University of Malaga, Spain*

[b] *Johann Bernoulli Institute of Mathematics and Computing Science, University of Groningen, The Netherlands*

ARTICLE INFO

Keywords:
Virtual environments
Robotic simulation
Service robots
Semantic maps
Robotic dataset
Unity

ABSTRACT

Simulations and synthetic datasets have historically empower the research in different service robotics-related problems, being revamped nowadays with the utilization of rich virtual environments. However, with their use, special attention must be paid so the resulting algorithms are not biased by the synthetic data and can generalize to real world conditions. These aspects are usually compromised when the virtual environments are manually designed. This article presents *Robot@VirtualHome*, an ecosystem of virtual environments and tools that allows for the management of realistic virtual environments where robotic simulations can be performed. Here "realistic" means that those environments have been designed by mimicking the rooms' layout and objects appearing in 30 real houses, hence not being influenced by the designer's knowledge. The provided virtual environments are highly customizable (lighting conditions, textures, objects' models, etc.), accommodate meta-information about the elements appearing therein (objects' types, room categories and layouts, etc.), and support the inclusion of virtual service robots and sensors. To illustrate the possibilities of *Robot@VirtualHome* we show how it has been used to collect a synthetic dataset, and also exemplify how to exploit it to successfully face two service robotics-related problems: semantic mapping and appearance-based localization.

1. Introduction

Service robotics research has traditionally resorted to datasets for the design and validation of techniques supporting the diverse robotic abilities, such as perception (vision, olfaction), navigation (map building, robot motion), or HRI (face recognition, speech recognition), among others. Datasets like KITTI (Geiger, Lenz, Stiller, & Urtasun, 2013), Microsoft COCO (Lin et al., 2014), Robot@Home (Ruiz-Sarmiento, Galindo, & González-Jiménez, 2017), or LFW (Huang, Ramesh, Berg, & Learned-Miller, 2007), to name a few, have contributed and will continue to contribute significantly to that end. However, these datasets are timestamped versions of the world, that is, they consist of *snapshots* captured at a certain time instant and from a certain position. Notice that this seriously hampers their exploitation in problems requiring a robot freely moving in the environment while capturing data, like those related with active perception (Bajcsy, Aloimonos, & Tsotsos, 2018), navigation (Cheng, Cheng, Meng, & Zhang, 2018), reinforcement learning (Burgueño, Ruiz-Sarmiento, & Gonzalez-Jimenez, 2021; Zhao, Queralta, & Westerlund, 2020), etc. Moreover, the gathering and processing of real world data is a complex and resource-intensive task, which does not line up with the hunger for data of modern robotics techniques (mainly those based on Deep Learning).

In this context, virtual environments have emerged as a promising alternative for alleviating those issues with simulations that are close to reality (Beattie et al., 2016; Wu, Wu, Gkioxari, & Tian, 2018). As for service robotics, they permit the introduction of virtual agents replicating real robots, their sensory systems (cameras, 2D laser scanners, sonars, etc.), and actuators (motors, arms, etc.). This way, robots can freely and safely operate in said environments, whose content and conditions are fully known and controlled. Gazebo (Koenig & Howard, 2004a), Stage (Gerkey, Vaughan, & Howard, 2003), or VirtualHome (Puig et al., 2018) are popular examples of tools enabling these simulations. One of the major challenges that faces the development of robotic models with

* Corresponding author at: Machine Perception and Intelligent Robotics group (MAPIR), Department of System Engineering and Automation, Biomedical Research Institute of Malaga (IBIMA), University of Malaga, Spain.

E-mail addresses: davfercha@uma.es (D. Fernandez-Chaves), jotaraul@uma.es (J.-R. Ruiz-Sarmiento), ajaenal@uma.es (A. Jaenal), n.petkov@rug.nl (N. Petkov), javiergonzalez@uma.es (J. Gonzalez-Jimenez).

https://doi.org/10.1016/j.eswa.2022.117970
Received 27 July 2021; Received in revised form 28 January 2022; Accepted 22 June 2022
Available online 5 July 2022
0957-4174/© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

synthetic data is the possible gap between simulated and real worlds. That is, the design of the virtual environments must be such that allows for the tuning of models transferable to real situations (Cabon, Murray, & Humenberger, 2020; Zhao et al., 2020). For example, while operating in a virtual environment, a robot aiming to recognize its surrounding objects requires that the appearing synthetic objects look realistic so the tuned models can successfully operate in real world conditions after their deployment (Josifovski, Kerzel, Pregizer, Posniak, & Wermter, 2018). Multiple resources exist providing virtual environments to the researchers community (*e.g.* Structured3D (Zheng, Zhang, Li, Tang, Gao, & Zhou, 2020)) however, they are usually designed by an operator according to his/her mindset, hence bias can appear and the simulation-real gap can be significant. Continuing the last example, a robot recognizing objects using contextual information, that is relations among objects typically appearing together, rooms where they appear, etc. (Ruiz-Sarmiento, Galindo, & González-Jiménez, 2015; Ruiz-Sarmiento, Galindo, Monroy, Moreno, & Gonzalez-Jimenez, 2019), requires that said relations mimics real ones. Otherwise, the robot will learn unrealistic contextual information, which will lead to a wrong operation.

This paper contributes *Robot@VirtualHome*, an ecosystem of virtual environments and tools supporting realistic robotic simulations with the novelty of including a collection of 30 virtual houses being replicas of real setups and a dedicated set of built-in solutions to recurrent issues. In contrast to other simulation tools or synthetic datasets which provide environments designed from the knowledge of designers, each virtual house has been designed out of real houses' resources (plans, images, point clouds, etc.) hosted on *idealista*,[1] a popular real estate website in Spain. In this way, each virtual house has the same room layout as the real one it mimics. In addition, objects appearing in real houses have also been replicated using virtual objects. These are 3D models of the same type (*e.g.* chair, table, microwave, etc.) as the objects found in the real environment and which are placed in the equivalent location to the position where the objects they recreate are. All these objects, as well as all the rooms, have been manually labeled in order to also provide ground truth information about the types of the elements they imitate, hence augmenting the geometric and appearance information with semantic knowledge.

This collection of houses is accompanied by a set of built-in tools that boost the simulations by: enabling a straightforward environment customization (lighting conditions, objects' models, etc.), managing the meta-information about the elements appearing in said environments (categories of objects and rooms, rooms connectivity, etc.), and supporting the simulation of arbitrary robots by means of customizable virtual agents which can be equipped with virtual sensors simulating real ones. Conveniently, we provide two virtual sensors commonly used in service robotics: a laser scanner and an RGB-D camera. The latter, in addition to the intensity and depth images, also provides an instance segmentation mask indicating the object to which each pixel belongs to and its category. These virtual agents can be also interfaced with external robotic algorithms, e.g. running on the Robotics Operating System, via a web protocol, hence enhancing their development. As an illustrative example, Fig. 1 shows one of these virtual environments, along with different pieces of information that can be extracted from them, and the tools developed for their management.

All these built-in functions make *Robot@VirtualHome* a multidisciplinary ecosystem of tools useful for realistic robotic simulations where a wide variety of problems can be addressed, including: relocalization based on similarity of appearance (Li et al., 2020; Taira et al., 2019), scene understanding exploiting semantic knowledge (Naseer, Khan, & Porikli, 2018; Ruiz-Sarmiento et al., 2015), active-vision (Haskins, Mentch, Botch, & Robertson, 2020; Straub & Rothkopf, 2021), generalization of knowledge in reinforcement learning (Buatois, Flumian, Schultheiss, Avarguès-Weber, & Giurfa, 2018; Kang, Belkhale, Kahn, Abbeel, & Levine, 2019; Pan, You, Wang, & Lu, 2017), or semantic mapping (Martins, Ferreira, Portugal, & Couceiro, 2019; Ruiz-Sarmiento et al., 2019), etc., to name a few. Interested parties can find the complete *Robot@VirtualHome* project shared publicly at: https://github.com/DavidFernandezChaves/RobotAtVirtualHome.

To exemplify the versatility and the advantages of this ecosystem, it has been used to built a dataset of sensory information from the totality of virtual houses provided, which results in an additional contribution of this work. This dataset is composed of more than ~113K captures of images (including RGB and depth images, as well as instance segmentation mask) and 2D laser scans during five raids of a robot on each house. Each raid has been conducted under different appearance conditions, e.g. some raids took place during the day with lights off while others were carried out during the night with lights on and using different object models. The interested reader can reach the dataset together with an API to handle it at https://zenodo.org/record/4610098#.YP_1t477RaY.

In this paper we also provide two additional use cases where we resort to *Robot@VirtualHome* as workbench. In the first one, we build a semantic map for each house and automatically evaluate their quality resorting to the provided ground truth information. In the second one, we expose a case of appearance-based localization in which, after carrying out a learning step, the robot must locate itself in a house with different appearance conditions.

In the following section we summarize works related to the one presented here. Then Section 3 provides an overview of *Robot@VirtualHome*, while Section 4 describes in depth the tools that comprise it and their utilities. Next, in Section 5 we report on the computational resource demands of *Robot@VirtualHome*, present the dataset built and expose two use cases in which we exploit and analyze the potential of this ecosystem. We conclude with the conclusions of the paper and future directions in Section 6.

2. Related work

In this section we discuss related works from the two research topics that most closely touch our proposal, virtual environments (Section 2.1) and datasets (Section 2.2).

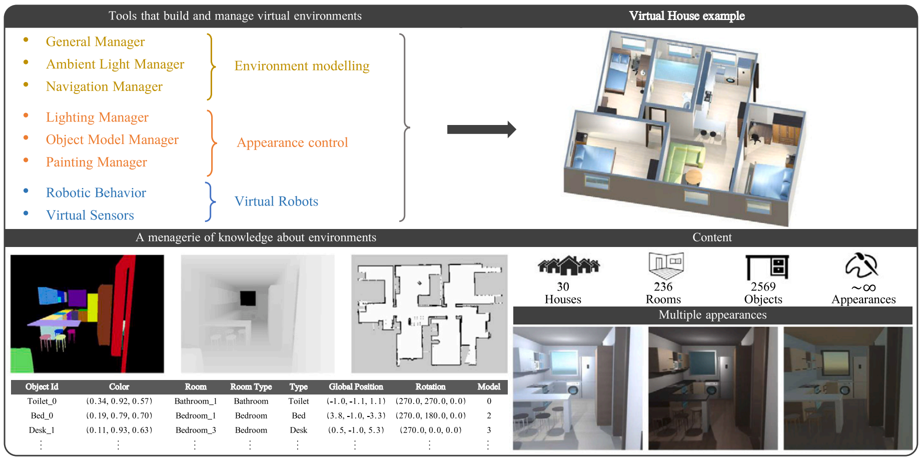
2.1. Virtual environments

Widely known robotic tools for simulation such as Gazebo (Koenig & Howard, 2004b) or Mujoco (Todorov, Erez, & Tassa, 2012) have greatly pushed the frontiers of robotic research in proposals of all kinds, serving as workbenches for testing and comparing their success. Recently, new simulation tools are appearing taking advantage of 3D virtual environments. For example, Wu et al. (2018) proposed a 3D environment to be used as a workbench in machine learning research. It contains 3D virtual houses equipped with a diverse set of fully labeled 3D objects, textures and scene layouts, based on the *SUNCG* dataset (Song, Yu, Zeng, Chang, Savva, & Funkhouser, 2017). Generalizable agents can be built in this virtual environment, presenting as an use case a robotic navigation system tasked to reach targets specified by semantic concepts.

Some of these tools are emerging as part of video game development engines (*e.g.* Unreal Engine or Unity[2]), which have been widely developed in recent years due to the rise of the video game industry, allowing the easy creation and managing of 3D virtual environments. For instance, Qiu and Yuille (2016) created *UnrealCV*, an open-source plugin for Unreal Engine 4 to construct virtual worlds where agents can perceive, navigate and take actions guided by AI algorithms. Similar to our proposal, *UnrealCV* allows the user to change the properties of the

[1] www.idealista.com.

[2] https://unity.com/.



Object Id	Color	Room	Room Type	Type	Global Position	Rotation	Model
Toilet_0	(0.34, 0.92, 0.57)	Bathroom_1	Bathroom	Toilet	(-1.0, -1.1, 1.1)	(270.0, 270.0, 0.0)	0
Bed_0	(0.19, 0.79, 0.70)	Bedroom_1	Bedroom	Bed	(3.8, -1.0, -3.3)	(270.0, 180.0, 0.0)	2
Desk_1	(0.11, 0.93, 0.63)	Bedroom_3	Bedroom	Desk	(0.5, -1.0, 5.3)	(270.0, 0.0, 0.0)	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 1. A schematic of all the tools involved in the simulation of each virtual house is shown above. In yellow, the modeling tools mainly in charge of the initialization of the virtual environment. In orange the tools that configure the appearance of the environment. In blue, those tools in charge of the virtual agents. At the bottom left they are shown some of the different types of information that can be obtained from virtual environments: a semantic instance mask, a depth image, an occupancy grid map, and an extract of a table showing the knowledge that is available for each object in the environment. On the bottom right, an example of three images taken from the same pose in the same house with different appearance conditions.

world by modifying materials or reflectances among others. Another example, this time developed in Unity, is *VirtualHome*, proposed by Puig et al. (2018). This is a simulator that contains pre-designed virtual houses in which artificial agents can be incorporated. These agents are programmed using Scrath, which is a descriptive language that uses atomic (inter)actions that they have implemented such as pick up or switch on/off. This work attempts to automatically generate programs from natural language descriptions, potentially allowing non-expert users to teach robots a wide variety of novel tasks.

Other relevant simulation environments are focused on specific domains such as communication language (Brockman et al., 2016; Miller et al., 2017), or video game strategy (Synnaeve et al., 2016; Tian, Gong, Shang, Wu, & Zitnick, 2017; Vinyals et al., 2017). We provide a versatile virtual environment that is interesting for several research areas such as scene understanding or 3D navigation.

2.2. Datasets

Virtual environments can play the role of a synthetic place-centric dataset (Ruiz-Sarmiento et al., 2017; Xiao, Owens, & Torralba, 2013) since both provide comprehensive information about the entire working environment, not being limited to certain views or portions of it. Examples of these synthetic environments could be *Replica*, by Straub et al. (2019), containing 18 highly photo-realistic 3D indoor scene reconstructions, or *Structure 3D*, proposed by Zheng et al. (2020), which is a large photo-realistic dataset which contains 3500 scenes thought out by a professional interior designer. In both cases they aim to achieve a high level of realism to bridge the gap with datasets from real environments. This is important, as datasets from real environments are often preferred, as the knowledge acquired with machine learning techniques can be applied to real world conditions with little modifications. In terms of place-centric datasets collected from real environments, there are many remarkable examples such as *Robot@Home* proposed by Ruiz-Sarmiento et al. (2017), or *Matterport3D* by Chang et al. (2017). These datasets contain knowledge about multiple real houses in the form of RGB images, point clouds or instance segmentation masks among others. In the first case, the data were gathered by a robot which multiple RGB-D cameras and a laser scanner, while in the second case, panoramic images were taken at specific locations in the robot working environment, which was reconstructed by a image matching process.

These datasets are a rich source of different types of contextual/semantic information, for example, how objects are placed in the environment according with their functionality, or how rooms are connected based on their type. These contextual relations are usually blurred when synthetics environments are considered, since they are established by the designer according to his/her (possibly biased) knowledge. Nevertheless, the information provided by real datasets could be defined as static, it is like it is, being limited to the environmental conditions at the time it was collected, and to the points in the environment from where it was gathered. This, for example, prevents the free navigation of an agent in the inspected environment.

Robot@VirtualHome aims to boost the research in different robotics-related tasks by bringing together the benefits of virtual environments and datasets collected from real setups. To this end, the proposed ecosystem includes virtual houses that are based on real scenes and holds relevant parts of the semantic knowledge they contain. Moreover, to endow said houses with a high level of versatility, different aspects like lighting conditions, object models, etc., are fully customizable. This ecosystem also supports the design of virtual agents that represent real robots. These agents can extract knowledge from the environment through the simulation of sensors, or through tools provided that are integrated in the virtual environment, for example the type of an object detected, or identify in which room the agent is located.

3. *Robot@VirtualHome* ecosystem overview

Realism is a recurrent limitation in synthetic environments. This is not only a purely aesthetic aspect, but it also concerns environmental meta-information, such as the layout and the occurrence of the types of objects in a room, which convey semantic knowledge of the scene. To mitigate the dissimilarities between reality and its virtual replica, our ecosystem includes 30 manually designed virtual houses mimicking real ones, hence avoiding the loss of information or possible biases caused by the criteria of a designer. This amounts to a total of 236 rooms with 2500+ virtual objects. The set of virtual houses incorporated attempts to contemplate different styles of contemporary houses. For example, some houses are spacious with more than 5 rooms, while others follow a minimalist style with the living room and the kitchen in a single area. *Robot@VirtualHome* attempts to provide realistic meta-information about the types of objects, rooms and the relations between them. For that, each virtual house has been designed using resources (plans, images, point clouds, etc.) from *idealista.com*, a popular real estate website in Spain. Virtual houses keep the same layout and proportions as their real counterparts. Moreover, in every room, virtual objects have been manually inserted replicating the existing ones in

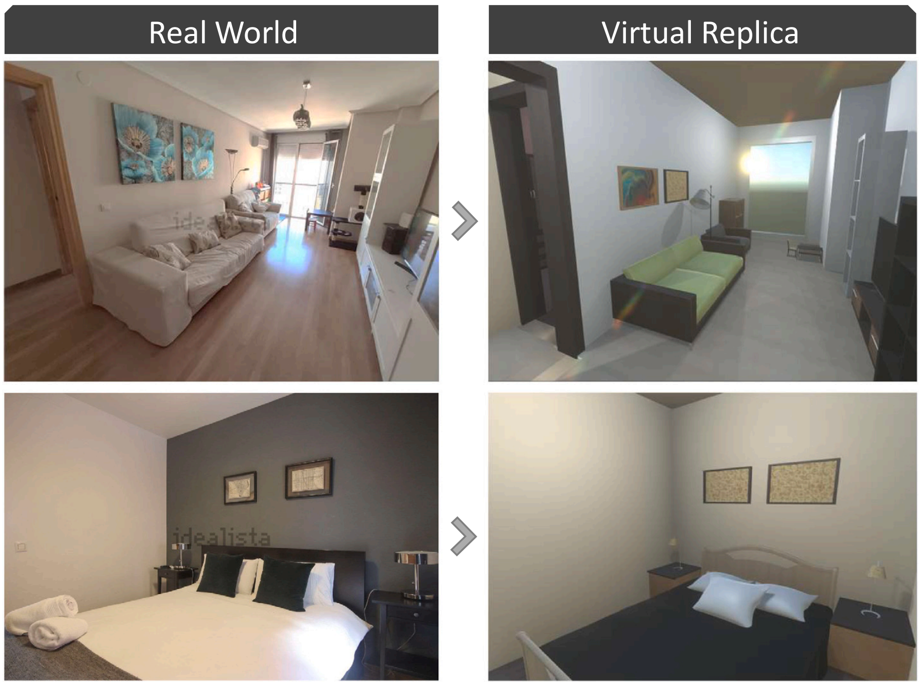


Fig. 2. Examples of the virtualization process. On the left, images of a living room and a bedroom in a real house. On the right, the virtual counterpart implemented in *Robot@VirtualHome*.

the real rooms that they mirror. Fig. 2 shows an example of this virtualization process in a living room and a bedroom.

The virtual objects populating houses are individual instances of a set of more than 60 types of generic virtual objects representing typical household items such as sinks, beds, microwaves, fridges, lamps, couches, etc. Each virtual object consists of a 3D representative model and a set of labels. These labels provide semantic information about the type of object that is virtually represented, which can be served as ground truth information in benchmark tests. An example of these labels could be the kitchen counter which is labeled both “counter” and “kitchen furniture”. Similarly, each of the rooms that make up the houses has also been manually labeled with the type of room it represents, e.g. *kitchen, bathroom, corridor*, etc. All this information can be easily retrieved by the tools integrated in *Robot@VirtualHome*, which support high-level queries like listing all the objects within a given room, get the position of an object, or to look up the objects of a specific type.

Also, we have developed a set of integrated tools to manage virtual environments and simulations in a simple way (see Section 4). These tools can access and modify both the environment, such as ambient light, or the robot’s free navigation area, as well as aspects of the house’s appearance, for example by changing the models of the objects or the paint on the walls among others. Another tools support the simulation of physical robots by the virtual agents. Specifically, these tools simulate sensors such as cameras, or laser scanners, and provide the virtual agents with decision making processes.

Both the houses and the tools have been designed as part of the Unity game engine, a video game development framework that is becoming increasingly relevant in the robotics field as it provides powerful tools for the design and control of virtual environments (Hu & Meng, 2016; Juliani et al., 2018; Navarro, Fdez, Garzón, Roldán, & Barrientos, 2018; Roldán et al., 2019). The tools we offer at *Robot@VirtualHome* exploit Unity’s features to customize simulations in a variety of ways and make the ecosystem more accessible to both people familiar with Unity and users new to the framework. From the point of view of the proposed system, Unity provides solutions for dealing with the calculation of lights, shadows and reflections projections in real time, the instantiation and management of 3D virtual models, or a tracing ray system, among others. Although there are other popular game engines such as Unreal Engine, we have opted for Unity because it has a very smooth learning curve (Linowes, 2015) and the community that supports it is overwhelmingly large. Furthermore, as we have shown in previous works (Fernandez-Chaves, Ruiz-Sarmiento, Petkov, & Gonzalez-Jimenez, 2019, 2020), Unity can be connected to robotic systems such as the widely known Robot Operating System (ROS) (Quigley et al., 2009) via the web-sockets protocol. In this way, a link between robotics and video game development is provided, allowing the user to exploit the most powerful features from both fields. This fact is so relevant that the development team of Unity has announced that it will officially support the ROS2 interconnection.

4. The ecosystem’s core: the tools

As well as the virtual houses, we offer in *Robot@VirtualHome* a set of built-in tools that facilitate the customization of the simulations. These tools have been developed in a modular way, so that, if needed, they can be replaced by other implementations or disabled. Based on the function they perform, they can be categorized into (recall the upper part of Fig. 1):

i. *Environment modelling* tools: deal with the building of 3D virtual environments and support the collection of data from the environment by the rest of the tools (see Section 4.1),
ii. *Appearance Control* tools: customize the environment according to the preferences of each simulation (Section 4.2), and
iii. *Virtual Robot* tools: are concerned with the management of virtual agents that mimic robots (see Section 4.3).

Next sections describe these tools in greater detail grouped by their category.

4.1. Environment modeling

Among the tools that model the virtual environment, the ***Environment Manager*** is the main one. It is in charge of instantiating and

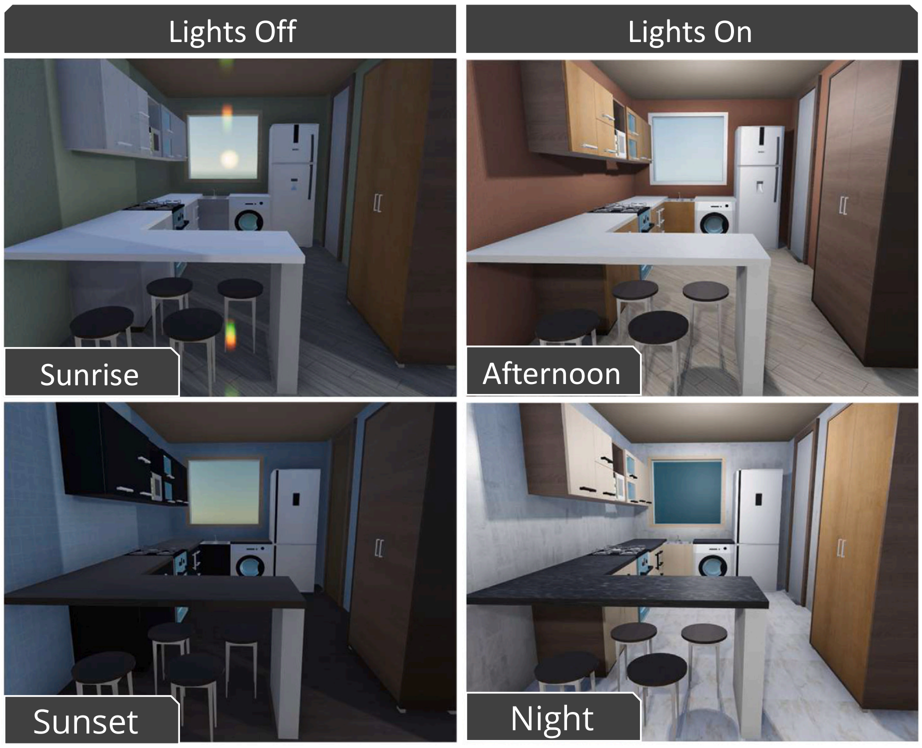


Fig. 3. Images taken from the same perspective using different appearance settings. (i) Up-Left: at sunrise, with ceiling room light off and no wall or floor painting. (ii) Up-Right: in the afternoon, with ceiling room light on and no wall or floor painting. (iii) Down-Left: At sunset, with the ceiling light of the room off and with wall and floor paintings. (iv) Below-Right: At night, with the ceiling light of the room on, and paintings in random mode.

initializing one of the virtual houses included in *Robot@VirtualHome*, propagating the appearance configuration to the rest of the systems based on a series of parameters in a configuration file. Those parameters have the form of the tuple $\mathcal{S} = \{\mathcal{L}, \mathcal{D}, \mathcal{M}, \mathcal{O}, \mathcal{A}\}$, where:

- $\mathcal{L}$ defines the light configuration,
- $\mathcal{D}$ the state of the doors,
- $\mathcal{M}$ the floor and wall paint configuration,
- $\mathcal{O}$ the models of the objects in the house, and
- $\mathcal{A}$ contains the information related to virtual agents.

Once the environment has been initialized, the *Environment Manager* saves a file which can be used as a footprint of the virtual environment in order to reproduce it again in future simulations. This file contains the set of objects $\mathcal{O} = \{o_1, \dots, o_n\}$ present in the house where each object o_i is codified by the following information: (i) the object unique id (representing its name), (ii) an associated single color (for mask images), (iii) the room in which it is located, (iv) the type of such a room, (v) the object type, (vi) its global position $\mathbf{T}_W$, vii) its global orientation $\mathbf{R}_W$, and (viii) the model chosen to represent it.

After that, the *Environment Manager* instance a virtual agent at the position where the virtual object called "Station" is located. This virtual object has no appearance and is only used to indicate where the robot starts in each house, however, it can be modified to, for example, have the appearance of a charging station.

Another tool that model the virtual environment is the **Ambient Light Manager**. It modifies the ambient light of the virtual environment. Formally, it is configured by the parameter $l_a \in \mathcal{L}$, which defines the rotation of the sun w.r.t the origin of coordinates of the house. This also can change the intensity and color of the virtual sun, which affects the ambient light and the virtual sky. This is based on the Unity's powerful light system which can calculate shadows, reflections, or even the refraction of textures. As a result, the simulations are affected by a virtual day and night cycle, changing aspects such as brightness, light tone or even the effect of flares on the lens, as shown in Fig. 3.

To be able to control the motion of virtual agents in a similar way as we control the motion of real robots, we have incorporated the **Navigation Manager**. This tool can be configured with the particular characteristics of a given robotic platform (*e.g.* its dimensions or maximum climbing slope) in order to calculate (at the beginning of a simulation) its free navigation area when operating in a virtual environment. Thus, paths can be calculated over said free navigation area in a similar way to traditional robotic navigation systems (*e.g.* those in the ROS navigation stack[3]). The free navigation area is affected by walls and virtual objects as well as by doors, hence for a given house it is different depending on the door settings used in each simulation. Recall that these door settings are provided by $\mathcal{D} = \{d_1, \dots, d_m\}$, where $d_i \in \{\text{on, off, random, customized}\}$

4.2. Appearance Control

Another interesting set of tools in *Robot@VirtualHome* are those oriented to provide different appearances of the same virtual environments. These tools are under the umbrella of the *Appearance Control* category, and are in charge of modifying the visual appearance of the virtual environment based on the configuration of a given simulation. Concretely, these tools are the *Lighting Manager*, the *Object Model Manager* and the Painting Manager.

While the *Ambient Light Manager* controls the simulation of the light in the rooms according to the day and night cycle, the ***Lighting Manager*** controls the simulation of artificial lights in houses. Both are configured by the set $\mathcal{L} = \{l_a, l_1, \dots, l_p\}$ in $\mathcal{S}$. While l_a sets the ambient light (recall Section 4.1), each $l_i \in \{\text{on,off,random}\}$ configures how the light point i is initialized. This tool also relies on the Unity's light control system to manage the different light sources in the house, both the ceiling lights and the lamps in each room. These points of light

[3] http://wiki.ros.org/navigation.

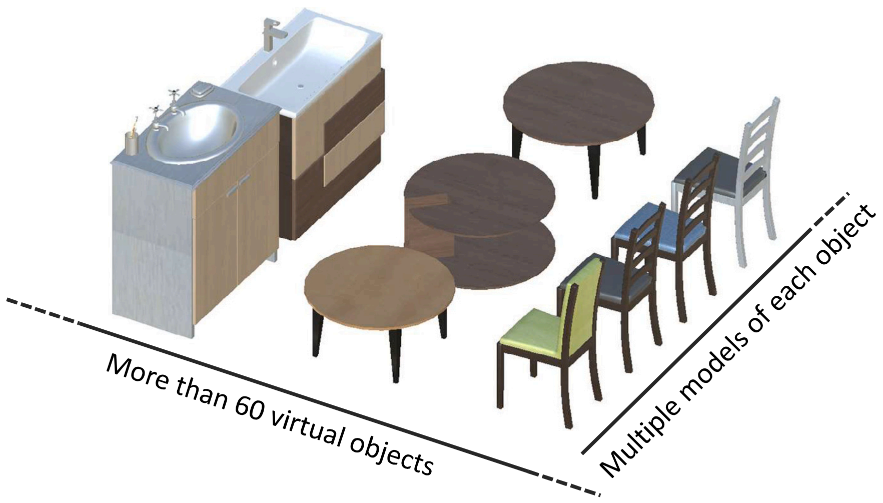


Fig. 4. 3D models associated with the object types "washroom", "small table" and "chair". Each virtual object can adopt different 3D models and/or different textures while maintaining the same proportions and location.

generate darker shadows than the ambient light, which have a great impact on the visual appearance of the environment. The images on the right side of Fig. 3 show an example of the effect of turning on the ceiling light in a room. Notice how in such scenarios the refrigerator shows a flash of light and the furniture casts shadows. Unlike the room lamps that have several possible 3D models, the ceiling lights do not have models to avoid the appearance of large shadows in the rooms, though they could be easily incorporated if needed.

It is worth mentioning that the *Lighting Manager* is one of the tools that requires more hardware resources, so it can be affected by the computational capabilities of the devices where the simulations are carried out. Nevertheless, Unity let us to adapt the rendering options to achieve varying levels of realism according to such capabilities.

In order to have a greater variability of virtual environments without modifying existing object types or locations, each virtual object can adopt different 3D models (see Fig. 4), which are handled by the **Object Model Manager**. Given a certain home from *Robot@VirtualHome*, at the start of a simulation, the *Object Model Manager* takes care of loading a 3D appearance model for each object o_i within it according to its predefined configuration in $\mathcal{O}$. This means that, for example, a chair can be shaped with a sloping or straight back and be made of metal or wood, being a chair in any case. This tool can be further configured to randomly choose one model, a specific one, or to inherit the chosen model from another virtual object. This feature allows the user to create sets of objects with a matching style, such as is usually the case of kitchen furniture (as shown in Fig. 3). *Robot@VirtualHome* has been structured to facilitate the addition of new virtual objects, or new models of existing ones, such providing highly customizable virtual environments.

In terms of visual appearance, we also provide the **Painting Manager**, which is based on the previous ones to modify the wall and floor paintings in each room. This tool can be configured by means of the parameters in $\mathcal{S}$, concretely those in $\mathcal{M} = \{m_1, \dots, m_q\}$, so $m_i \in$ {plain color, texture_1, …, texture_r, random}. An example of the visual effect of these paintings can be seen in Fig. 3, where the images show walls and/or floors with different colors and textures. This feature is especially useful for keypoint based vision systems, which often require surfaces to have texture in order to be able to extract information.

4.3. Virtual Robot

Robots can be represented within the virtual environment by simple models such as geometric primitives, or by complex ones that faithfully represent the appearance of real robots. Regardless of this, the robots' simulation is performed through agents controlled by the *Virtual Robot* tools. Both the geometric design of the virtual agent (such as the robot model, sensors' extrinsic calibration, etc.) and the configuration of the theses tools are assembled in $\mathcal{S}$, concretely in $\mathcal{A} = \{a_1, \dots, a_r\}$ where r stands for the number or virtual agents, using high-level models commonly used in Unity called "prefabs". *Virtual Robot* tools can be divided according to their functionality into: i) those that allow to control the movement of virtual agents, when and which data to capture, etc. –that is, those defining a behavior–, and (ii) tools that simulate sensor readings.

It is worth mentioning that the actions performed by virtual agents can arise externally to the virtual environment through applications connected through the web-socket protocol or by the previously mentioned tools that simulate the agent's behavior. *Robot@VirtualHome* incorporates the necessary resources to establish a connection and exchange information with the well-known ROS ecosystem. If this alternative is chosen, the simulation of the robot from the virtual environment side would involve visual aspects and sensors' simulation, leaving decision making (*e.g.* how it navigates) to said external systems.

However, Unity is versatile enough to allow the creation of complex decision making tools such as those incorporated in modern service robots. This option presents some useful features that speed up the development with respect to using external robotic platforms, since being integrated in the virtual environment, hardware aspects can be simplified without having to resort to other simulation software, for example, the level of the robot battery or motor control. Furthermore, since these tools are integrated in the same ecosystem as the other tools, they have an easy access to information about the virtual environment, such as the exact position or type of surrounding objects. To facilitate and exemplify the simulation of robotic behaviors on Unity, *Robot@VirtualHome* includes three basic tools that modify the behavior of the robot configured by the parameter $a_b \in a_i$, where $a_b \in$ {None, Wanderer, Grid, Manual}.

- **Wanderer**: At the beginning of the simulation, the projections of the ceiling lights onto the floor are calculated, obtaining a set of points of interest per room: $P^r = \{l_1^r, \dots, l_n^r\}$ where r is the room number and n is the number of ceiling lights in the room. These points are then shifted to coincide within the robot's free navigation area. After this, all the sets are joined into one which is used as a reference to guide the robot: $Wps = L^0 \cup L^1 \cup \dots \cup L^R$ where R is the number of rooms in the house. This set (Wps) can be tracked cyclically, in order or randomly. The result is a sequence of images from small position increments that show at least once all the rooms in the house.

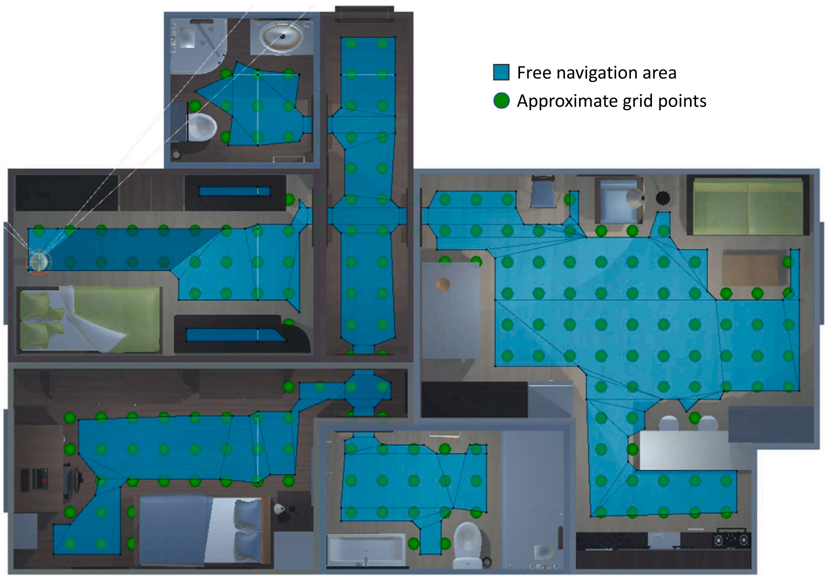


Fig. 5. Floor plan image of the virtual house 15. The blue area corresponds to the free navigation area of the virtual robot. Green spheres represent the nodes that the robot tries to reach using the grid behavior to capture data.

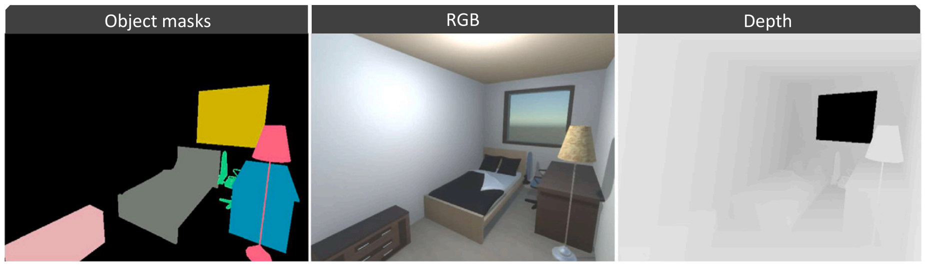


Fig. 6. Images returned by the *Smart Camera*. On the left, image of instance segmentation masks, where each color represents a different object. In the middle, the RGB intensity image. On the right, the depth image.

- **Grid**: At the beginning of the simulation, a grid of equidistant nodes of configurable distance is generated covering the whole robot's navigation area (see Fig. 5). Similar to the previously explained tool, these nodes form a set which the robot follows in an orderly fashion. While during the path the robot does not capture data, at each node it performs a 360° turn capturing data. This tool facilitates the acquisition of images from multiple perspectives of the environment, which is very useful in some robotic applications such as appearance-based localization.
- **Manual**: This is a standard remote control tool, with which we can control the virtual robot's motion by means of a keyboard.

Besides, all of these behaviors can be configured to save a log file containing the pose (position $\mathbf{T}_W$ plus orientation $\mathbf{R}_W$) and room name in which the robot captured sensory information.

Regarding the gathering of information from the virtual environment, the robot can use both the *Environment Manager* to obtain ground truth information, as well as tools to simulate sensors. Sensors can be modeled to capture data from the virtual environment in the same way as robots' sensors would in a real environment. For example, cameras can be simulated using the virtual cameras already built into Unity, or a robotic laser scanner can be simulated using ray tracing. However, knowledge about the virtual environment is absolute, unlike in real setups where uncertainties appear.

Robot@VirtualHome incorporates two tools that simulate common robotic sensors to bridge the gap between virtual and real environments: i) what we have called the *Smart Camera* and ii) a *Laser Scanner*. The first one simulates the behavior of a standard RGB-D camera enhanced to access the semantic information of the virtual environment. This tool can capture intensity (RGB) and depth images, as well as instance segmentation masks (see Fig. 6). These images can be compressed in *JPG* format and sent to ROS through the connection maintained by the virtual agent, thus simulating the performance of a real camera. For further similarity, this tool can be specifically configured with intrinsic parameters like those of real cameras so that deformations can be considered, sensor size can be modified, etc. (see Zuñiga-Noël, Ruiz-Sarmiento, and Gonzalez-Jimenez (2019) for more details). For convenience, the *Smart Camera* returns the intrinsic calibration matrix K from a given sensor size (s_x, s_y), a focal length (f) and an image resolution (r_x, r_y) based on:

$$K = \begin{pmatrix} f * r_x/s_x & 0 & r_x/2 \\ 0 & f * r_y/s_y & r_y/2 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

In addition, to improve the realism of the images obtained by this sensor, *Robot@VirtualHome* incorporates "Post Processing Stack", a popular post-processing package for images in Unity. This allows the virtual camera to incorporate effects such as lens distortions, motion blurring, light flare effects, etc. Concerning the capture of instance segmentation masks, the *Smart Camera* exploits the shader system available in Unity to efficiently obtain images with color masks where each color is identified with an instance of a virtual object.

The laser scanner is one of the most used sensors in robotics, so for the sake of usability, we have designed a tool that simulates the behavior of a real laser scanner in virtual environments. This tool uses Unity's built-in ray tracing system to measure the distance between the

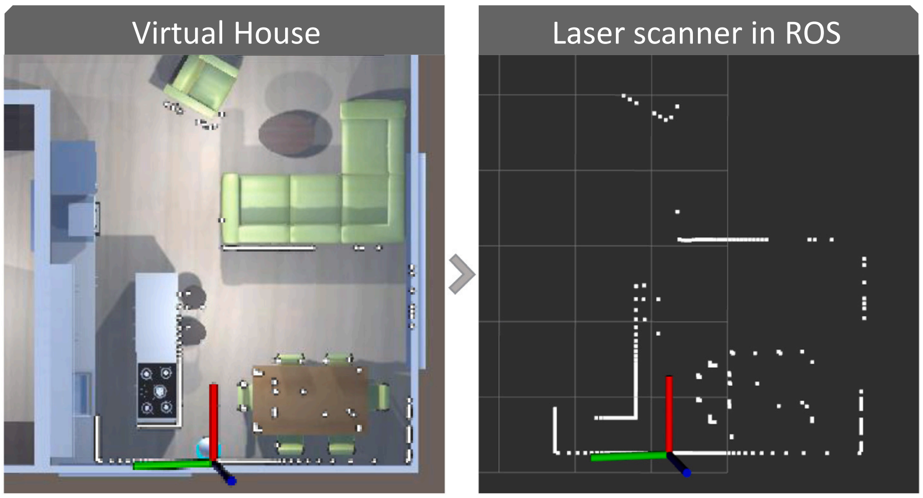


Fig. 7. Simulation of a robot equipped with a 360° laser in one of the virtual houses. On the left are shown the points in the virtual environment where the laser beams have collided. This information can be transferred to ROS as if the laser were a real instrument equipped on a robot. On the right, a snippet from RVIZ (a typical ROS application for visualizing data) showing the data received from the virtual laser.

sensor and the surrounding environment. This tool also incorporates specifically parameters that configure the realism obtained, modifying aspects such as field of view, resolution, accuracy, or maximum operation range to simulate the limitations of real sensors. In this sense, the output of the simulated laser is an ordered set of distances. As with the camera, this virtual sensor can exploit the virtual agent's connection to ROS to send measured distances as if it were a real sensor. As an example, Fig. 7 shows a scene where the virtual laser measurements are displayed both in the virtual environment and in RVIZ, a ROS data viewer.

5. Running simulations

To show a glimpse of the possibilities offered by *Robot@VirtualHome*, we present some use cases in which we analyze and prove the performance of the simulations. In this section we first perform a quantitative analysis of the resources required to launch simulations with *Robot@VirtualHome*, as well as the performance during the execution of a simulation (see Section 5.1). Next in section Section 5.2 we give details on the design of a virtual agent to simulate a commercial robot used in the use cases. In Section 5.3, we present the *Robot@VirtualHome* dataset, which contains information from five raids on each house with different appearance configurations. Finally, we exhibit two approaches to robotics-related problems using the presented ecosystem as workbench: semantic mapping (see Section 5.4) and appearance-based localization (see Section 5.5).

5.1. Demand on computational resources: quantitative analysis

To measure the ecosystem's demand on computational resources, we have measured the memory allocation and CPU usage in thirty simulations, one for each virtual house, using the Unity built-in profile. The simulations were performed using randomized configurations and also connecting the agent and virtual sensors to ROS. These performance tests, as well as the building of the dataset and the use cases later discussed, have been performed using a computer equipped with an Intel Core i7-5700HQ processor at 2.70 *GH*, a RAM memory with 2×8 *GB* DDR3 at 800 MHz, and a graphic card NVIDIA GeForce GTX 960M with a memory of 2 *GB*.

Fig. 8 shows on the left, as an example, a time lapse representing the time the CPU takes to process each frame in a simulation. Notice that although it provides information on one simulation, it is quite representative since all simulations reported similar figures. This graph shows how CPU usage is fairly constant with an average use of 6 ms to dispatch a frame ($> 60FPS$), allowing the simulations to run smoothly, with some occasional peaks. These spikes are due to the process of packaging and sending the virtual sensor data to ROS, which is computationally expensive. On average, sending images (intensity, depth and semantic instance masks) from the smart camera takes 55.54 ms, while sending measurements from the laser scanner takes 19.90 ms. To ensure that the fluidity of the simulations are not compromised by the high consumption of these processes, we run them in parallel threads to the main one in order not to block the rendering of the scene. In this way we spread the computational cost among several frames, hence the highest peak of the graph is 37.87 ms.

In terms of memory, the simulations have used on average 1.01 G. Fig. 8 shows the average measured allocations where 192.35 Mb (19.0%) has been used in storing the textures, 267.18 Mb (26.4%) for the meshes and 1.3 Mb (0.01%) for the materials. The remaining 54.59% is used by Unity for its internal processes. Note that memory allocation is handled automatically by Unity so there may be slight differences on other computers.

5.2. Modeling and simulating robots by means of virtual agents

To achieve realistic results in the applications shown, we have designed a virtual agent that simulates a commercial robot. Specifically, we have imitated the Giraff robotic platform (González-Jiménez, Galindo, & Ruiz-Sarmiento, 2012), which has proven to be valid in a variety of indoor applications (Fernandez-Chaves et al., 2020; González-Jiménez et al., 2012; Ruiz-Sarmiento et al., 2017). This robot is designed for telepresence purposes and has a laser scanner at the base to locate itself and a tilted camera at the top to interact with people.

To replicate this robot, we used a virtual agent with a 2D laser scanner attached at the base at a height of 0.25 m. This laser was configured to cover a 360° range with 1° increments between each measurement working at a frequency of 0.5 Hz. It should be noted that in virtual environments there is no uncertainty so measurements are accurate, however, Gaussian noise can be used to simulate realistic measurements. Regarding the camera, we have used a *smart camera* placed at a height of 1.59 m and with a 10° rotation in the pitch axis, resulting in a lower front view similar to the camera used in Giraff robots. In addition, the following intrinsic parameters were used to replicate the real camera mounted on the robot: sensor size of 21.0x15.2 mm, focal length of 18 mm, field of view of 45° and lens shift of (0.0)mm and VGA resolution (640*x*480px). In terms of enhancement,

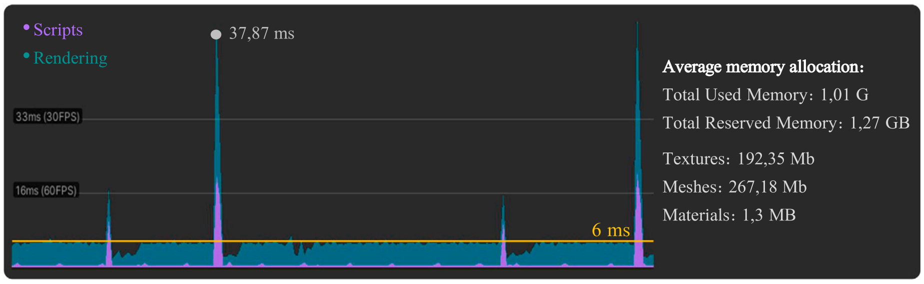


Fig. 8. On the left, a time lapse graph extracted from a simulation where the CPU time required to process each frame is plotted. The purple color represents the time used to process the scripts, and green is the time used for rendering. In yellow, the overall average performance. On the right, the average memory allocation of all simulations.

Table 1
Settings used in the appearance tools for the acquisition of the dataset.

	Ambient Light $l_a \in \mathcal{L}$	Lamps in rooms $\{l_1, \dots, l_p\} \in \mathcal{L}$	Ceiling lighting of the rooms $\{l_1, \dots, l_p\} \in \mathcal{L}$	Painting Walls and Floors $\{m_1, \dots, m_q\} \in \mathcal{M}$	Object Models $\{o_1, \dots, o_j\} \in \mathcal{O}$
Raid 0 (a_b = Grid)	100° - Day	Off	On	Plain Color	Basic preset model
Raid 1 (a_b = Wanderer)	100° - Day	Off	On	Plain color	Basic preset model
Raid 2 (a_b = Wanderer)	100° - Day	Off	Off	Random	Random
Raid 3 (a_b = Wanderer)	180° - Sunset	Random	Off	Random	Random
Raid 4 (a_b = Wanderer)	200° - Night	Random	On	Random	Random

we take advantage of the potential of this virtual tool to capture images not only of intensity, but also of depth and semantic masks. Concerning the navigation parameters, in all the experiments presented in this work, the configuration was as follows: maximum linear velocity of 0.25 units/s, maximum angular velocity of 25°/s and an acceleration of 3 units/s^2. It should be noted that one unit is equivalent to one meter, considering the scale of the houses in virtual environments as reference. All the parameters mentioned in this subsection are collected in a Unity prefab as $a_1 \in \mathcal{A}$ to be used during the simulations.

5.3. The Robot@VirtualHome dataset

The dataset presented here is constituted by information gathered by means of the virtual agent previously designed during five raids in each of the thirty houses. Specifically, we have collected: (i) images of intensity, depth and instance segmentation masks, (ii) 2D laser scans, (ii) occupancy maps, (iii) logs with the ground truth of the objects and models used and (iv) logs with information regarding the trajectory followed that links the data with the pose from where they have been collected. It should be noted that the occupancy maps for each house have been obtained from the ROS gmapping package[4] by means of the sensory information collected by the virtual laser scanner included in *Robot@VirtualHome*.

For the first raid in each house, we used the grid behavior tool, which was configured with a separation between nodes of 0.5 m and 10 data captures per node, i.e., in each node a capture was performed every 36 degrees. In the remaining four raids we used the wanderer behavioral tool configured to capture data every 0.5s. Table 1 shows the appearance tool settings used in each raid. These settings cover a wide range of appearance variety, having samples with different daytime shades, different object models, etc.

The resulting dataset is composed of a total of 113278 captures, giving a total of 339.834 images. Table 2 details how many captures have been made in each raid. In raids where the robot used a grid behavior, the robot captured more images per house. This was expected since the aim of this behavior is to obtain observations from all its reachable locations. For example, in the case of house 23, with has 14 rooms (indeed it is the largest house), 2,990 captures were taken. On the other hand, the wandering behavior made the robot go through all

Table 2
Overview of the dataset content with the number of captures, objects and rooms observed for each house. W1, W2, W3 and W4 refer to the four raids carried out with the wanderer behavior.

House	Number of captures					Objects	Rooms
	Grid	W1	W2	W3	W4		
1	1750	423	624	716	378	107	10
2	1610	232	273	221	398	68	5
3	1710	986	1002	687	693	114	12
4	2240	904	1319	988	952	118	10
5	1680	1127	1321	1143	1099	102	10
6	1120	361	476	574	591	81	8
7	1080	635	682	714	479	89	9
8	1180	558	916	564	491	97	8
9	1440	725	795	698	525	85	9
10	870	441	397	415	362	82	7
11	680	161	131	158	207	42	5
12	1040	330	440	432	364	62	7
13	1560	596	522	664	404	84	9
14	2390	677	747	827	719	109	9
15	1870	613	486	441	456	74	6
16	1420	765	701	720	497	90	9
17	1340	628	512	408	497	86	6
18	940	402	302	228	490	75	7
19	1190	445	438	425	433	78	8
20	640	163	114	167	113	48	3
21	2410	855	1329	1170	1104	127	9
22	780	161	167	232	230	51	5
23	2990	1470	1366	1764	1509	121	14
24	1480	675	704	665	892	77	8
25	1590	922	1186	1049	828	104	9
26	1320	361	665	536	578	97	9
27	1940	965	977	822	817	134	12
28	950	155	206	276	212	63	5
29	320	91	92	87	87	52	4
30	1120	185	222	183	125	67	4

the rooms of the house randomly, so that the raid, depending on the order followed, can be longer or shorter. This means that raids in the same house can have significant differences in the number of captures.

In order to handled this data, we have added a simple API developed in Phyton. The API accesses the logs included in the dataset to build a rich data structure that can be incorporated into other applications. In this way, we provide an easy and agile solution to extract information from the generated logs, being able to access information such as the

[4] http://wiki.ros.org/gmapping

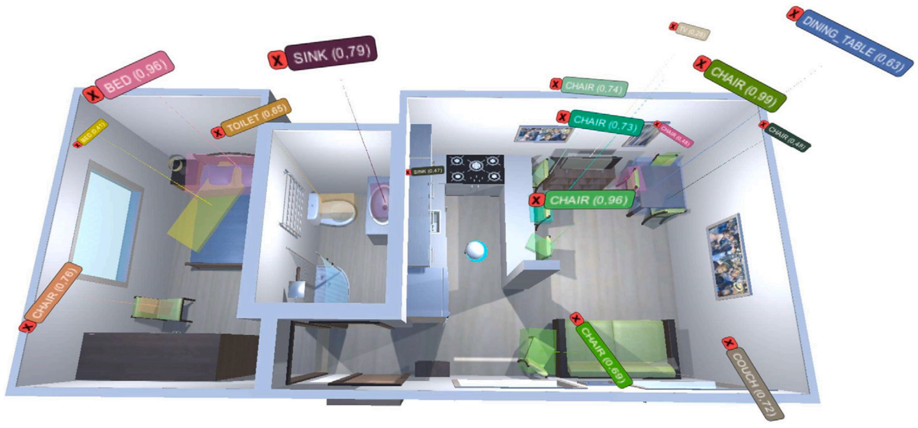


Fig. 9. Image of the semantic map obtained in the first use case in the 20th house. The detected objects are shown with a detection box and a label which shows the type of object and the confidence score of the detection. Each detection has a unique color that is used in both the boxes and the labels.

number of objects in a given image, or to know how many objects or rooms of a certain category exist in a house.

5.4. Use case: Building semantic maps

Semantic maps are representations of the robot workspace enriched with meta-information that provide additional knowledge about the elements that it contains (*e.g.* categories of objects and rooms, functionalities, etc.). Robots can take advantage of these maps to achieve a certain level of understanding of their surroundings. For example, these allow them to tackle tasks at a high level like "go to the bathroom", rather than "go to the x, y coordinates". In a previous work, we proposed *ViMantic* (Fernandez-Chaves, Ruiz-Sarmiento, Petkov, & Gonzalez-Jimenez, 2021b), an architecture that allows the building of semantic maps from the objects detected by robots connected to it. For doing so, the semantic map is represented as a virtual environment where *ViMantic* instantiates labeled bounding boxes that refer to one or a set of object detections. In this way, the virtual environment shows at a glance the position, size and type of the objects detected by the connected robots during their operation. This approach was tested on *Robot@Home* (Ruiz-Sarmiento et al., 2017), a dataset containing data from real houses, using different state-of-the-art neural networks for object detection, producing positive results (see Fig. 9).

In this use case we have relied on *ViMantic* to build a semantic map for each of the thirty virtual houses included in *Robot@VirtualHome*. For this purpose, we have performed a simulation in each house using random appearance settings and the virtual agent described in Section 5.2 using the wanderer behavior tool. A fundamental task to properly populate semantic maps is such of object detection, which has been carried out by means of the state-of-art neural network Detectron2 (Wu, Kirillov, Massa, Lo, & Girshick, 2019), as it reached the highest success rate in our previous works.

The *Environment Manager* allows us to obtain information about the virtual environment such as the type of a virtual object, or a list of objects inside the house. Resorting to this ground truth information, we have automated the computation of relevant statistics such as precision (the number of detection boxes successfully instantiated) and recall (the number of detectable objects that have been successfully detected).

Table 3 shows the results obtained in each simulation. It is worth mentioning that the semantic map building has been a passive task while the robot wandered around the house. This is reflected in the low rate of objects found (Recall), with an average of 40%. This way there is room for improvement, for example by using a dedicated semantic mapping behavior that explores the environment in detail. As for the precision, the average from all houses is 72.7%. However, houses such as 10 or 29 have obtained a lower precision rate (66.7% and 50% respectively) due to the low number of objects detected in them.

It is important to clarify that detection boxes tend to grow and encompass objects of the same type that are close together. A recurring case of this occurs when there are multiple chairs around a table, a single detection box encompasses all the chairs when they are close enough together. In this way, the number of detected objects is greater than the number of detection boxes, for example in house 3 where there were 11 correct detection boxes and a total of 16 objects found. In contrast, it also happens that the CNN used detects the same object multiple times, e.g. in case of partial occlusion. In these cases, two detection boxes can be found for a single object. In this way, the number of detection boxes is greater than the number of detected objects, for example in house 30 where there were 11 detection boxes for 8 detected objects.

5.5. Use case: Appearance-based localization

Appearance-based localization is the task of localizing an image within an environment without relying on local features, solely employing global features and a previously constructed map of image descriptor-pose pairs. One of the main advantages of such approaches, as a result of using whole image descriptors (Arandjelovic, Gronat, Torii, Pajdla, & Sivic, 2016), is their robustness against extreme lighting conditions.

Existing real world, indoor datasets for localization are mainly gathered during robot or camera navigation through an environment, although only a fraction of them include records of the same scene under different lighting conditions, which impedes to evaluate localization under challenging conditions. Images captured from pose grids are an alternative source of information to generate appearance maps, however, their construction in real environments turns to be a time-consuming and expensive process, being only feasible for small areas. On the other hand, existing virtual datasets can efficiently gather grid maps, with the drawback that few of them simulate realistic environments and, to the best of our knowledge, none of them includes the feature of varying the environment appearance and illumination along with navigation capabilities.

The *Robot@VirtualHome* ecosystem allows to acquire images from navigation sequences or grid maps under diverse illumination settings, which we took advantage of to test the impact of illumination on the performance of an state-of-the-art appearance-based localization method. Concretely, this problem was approached with (Jaenal, Moreno, & Gonzalez-Jimenez, 2021), a solution that divides the environment into regions called *Patches of Smooth Appearance Change* (PSACs) that model the local association between pose and the holistic descriptor, specifically NetVLAD (Arandjelovic et al., 2016). We designed an experiment involving several illumination settings for: i) a single grid map obtained with the grid behavior configured with a

Table 3
Results of the semantic map building for each house contained in *Robot@VirtualHome*. The Detections column shows the number of elements detected by the CNN in all frames while Detection boxes are ViMantic's representation of the objects detected in the environment. Detection boxes can correspond to one or more detections.

House	CNN detections	Detection boxes	Detected objects	Precision (%)	Recall (%)
1	288	22	16	68,2	38,1
2	132	11	6	63,6	40,0
3	427	15	16	73,3	45,7
4	395	20	13	50,0	38,2
5	221	13	11	76,9	35,5
6	134	12	13	75,0	46,4
7	159	9	10	77,8	40,0
8	151	10	10	70,0	32,3
9	129	11	8	72,7	34,8
10	51	3	2	66,7	11,1
11	106	7	6	71,4	40,0
12	193	12	10	75,0	50,0
13	304	18	12	66,7	46,2
14	424	19	14	78,9	45,2
15	199	13	8	76,9	40,0
16	141	12	10	83,3	37,0
17	252	19	16	78,9	55,2
18	183	10	8	70,0	36,4
19	198	17	9	70,6	36,0
20	54	6	7	83,3	43,8
21	371	20	16	70,0	45,7
22	59	6	3	83,3	27,3
23	633	28	18	71,4	56,3
24	461	18	14	77,8	66,7
25	878	30	17	66,7	45,9
26	188	14	14	92,9	51,9
27	413	20	22	75,0	48,9
28	81	7	4	71,4	17,4
29	28	2	3	50,0	10,0
30	145	15	8	73,3	40,0
Average	247	14	11	72,7	40,0

Table 4
Comparative median position (ϵ) and rotation (θ) errors reported by the appearance-based localization approach, tested within maps and sequences with diverse radiometric conditions. The parameter l_a shown in the name of each experiment indicates the angle of the virtual sun with respect to the house to simulate the time of day.

	Grid Day ($l_a = 170^o$)		Grid Sunset ($l_a = 179^o$)		Grid Night ($l_a = 200^o$)	
	ϵ	θ	ϵ	θ	ϵ	θ
Route Day ($l_a = 90^o$)	0.23	12.36	0.27	12.58	0.38	19.28
Route Sunset ($l_a = 175^o$)	0.26	16.17	0.26	11.42	0.29	12.08
Route Night ($l_a = 185^o$)	0.30	15.99	0.31	16.23	0.46	22.42
Route Night ($l_a = 195^o$)	0.34	17.61	0.21	10.28	0.34	11.20
Route Day ($l_a = 25^o$)	0.21	7.22	0.26	12.35	0.44	12.64
Route Day ($l_a = 165^o$)	0.27	12.56	023	10.80	0.43	20.48

distance between nodes of 0.5 m and 10 images per node, and (ii) a $\sim$ 25 m long test route obtained following a wanderer behavior (refer to SEC4.3). In addition, we created a *synthetic odometry* for the route, corrupted by zero-mean Gaussian noise with $\sigma_u = (0.06 \text{ m}, 1°)$.

Table 4 shows the compared localization performance for the different radiometric conditions of the sequence and grid map, expressed in terms of median errors in position and orientation after five runs for each scenario. The results demonstrate that, when localizing on maps gathered under brighter luminosity, as Day or Sunset, less position and orientation error is incurred than at night, thus providing a more accurate description of the environment and higher performances in appearance-based localization. Finally, the worst performance reported occurred when localizing under radiometric settings that differ significantly from those of the map, e.g: Day and Night, Sunset and Night, etc.

6. Conclusions

In this paper we introduced *Robot@VirtualHome*, an ecosystem of tools for managing virtual environments and agents developed to boost the research in different service robotics-related tasks with realistic simulations. The adjective "realistic" is due to the unique way in which the 30 virtual houses provided were designed: mimicking the layout of rooms in real houses as well as the position and type of the objects that can be found therein. This enables generalization to the real world, which is essential in robotic simulations where scene semantics play a fundamental role. The proposed ecosystem of tools also comprises a number of valuable features: modification of the lighting of the environment and lamps, change of the paint of floors and walls, setting of the free navigation space by opening and closing doors, or customization of objects' models, among others. Service robot simulation can be performed using customizable agents called virtual robots. These can be controlled by algorithms developed directly in the virtual environment or can be connected to real robots through standardized *WebSocket* communication channels. In addition, virtual robots can include virtual sensory systems that model real sensors such as cameras or laser scanners.

We have also exposed the demand for computational resources during the simulations. On average, the simulations occupy $1G$ of RAM memory and each frame takes 6 ms ($> 60FPS$) to process. To illustrate the possibilities of our proposal, we have built a dataset of images captured in the provided virtual houses under diverse appearance conditions. This dataset contains RGB and depth images, instance segmentation masks, 2D laser scans and occupancy grid maps. Additionally, we have also shown two use cases where *Robot@VirtualHome* is exploited to successfully address robotics-related problems. In the first one, we have employed the provided functionalities and ViMantic (Fernandez-Chaves et al., 2021b) to build semantic maps for each of the virtual houses. In the second, we have introduced an appearance-based location application and tested it in one of the virtual houses.

In the future we plan to further leverage and develop the features of *Robot@VirtualHome*. For example, to include houses with more than one floor as well as new room types such as basements, gardens or attics. In terms of applications, there are many interesting alternatives, for example, using the virtual environment as a workbench to train reinforcement learning systems that actively explore houses (Fernandez-Chaves, Matez-Bandera, Ruiz-Sarmiento, Monroy, Petkov, & Gonzalez-Jimenez, 2021). Another interesting use case is to compare and measure the performance of different object detection networks in different lighting conditions, so that the robot could choose which CNN to use depending on the ambient light at any given time.

CRediT authorship contribution statement

David Fernandez-Chaves: Methodology, Software, Data curation, Investigation, Writing – original draft. **Jose-Raul Ruiz-Sarmiento:** Methodology, Validation, Supervision, Writing – review & editing. **Alberto Jaenal:** Data curation, Writing – original draft. **Nicolai Petkov:** Funding acquisition, Project administration, Supervision, Writing – review & editing. **Javier Gonzalez-Jimenez:** Conceptualization, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the research projects *WISER* (DPI2017-84827-R), funded by the Spanish Government and financed by the European Regional Development's funds (FEDER), *ARPEGGIO* (PID2020-117057GB-I00), funded by the European H2020 program, by the grant number FPU17/04512 and the UG PHD scholarship program from the University of Groningen. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal used for this research. We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

References

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5297–5307).

Bajcsy, R., Aloimonos, Y., & Tsotsos, J. K. (2018). Revisiting active perception. *Autonomous Robots*, *42*(2), 177–196.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., et al. (2016). Deepmind lab. arXiv preprint arXiv:1612.03801.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Buatois, A., Flumian, C., Schultheiss, P., Avarguès-Weber, A., & Giurfa, M. (2018). Transfer of visual learning between a virtual and a real environment in honey bees: the role of active vision. *Frontiers in Behavioral Neuroscience*, *12*, 139.

Burgueño, A. M., Ruiz-Sarmiento, J. R., & Gonzalez-Jimenez, J. (2021). Autonomous docking of mobile robots by reinforcement learning tackling the sparse reward problem. In *Lecture notes in computer science*: *Vol. 12862*, (pp. 392–403). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-85099-9_32.

Cabon, Y., Murray, N., & Humenberger, M. (2020). Virtual kitti 2. arXiv preprint arXiv:2001.10773.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., et al. (2017). Matterport3d: Learning from rgb-d data in indoor environments. arXiv preprint arXiv:1709.06158.

Cheng, J., Cheng, H., Meng, M. Q.-H., & Zhang, H. (2018). Autonomous navigation by mobile robots in human environments: A survey. In *2018 IEEE international conference on robotics and biomimetics (ROBIO)* (pp. 1981–1986). http://dx.doi.org/10.1109/ROBIO.2018.8665075.

Fernandez-Chaves, D., Matez-Bandera, J. L., Ruiz-Sarmiento, J. R., Monroy, J., Petkov, N., & Gonzalez-Jimenez, J. (2021). Exploiting spatio-temporal coherence for video object detection in robotics. In *International Conference on Computer Analysis of Images and Patterns* (pp. 186–196). Springer.

Fernandez-Chaves, D., Ruiz-Sarmiento, J., Petkov, N., & Gonzalez-Jimenez, J. (2019). Integration of cnn into a robotic architecture to build semantic maps of indoor environments. In *International work-conference on artificial neural networks* (pp. 313–324). Springer.

Fernandez-Chaves, D., Ruiz-Sarmiento, J., Petkov, N., & Gonzalez-Jimenez, J. (2020). From object detection to room categorization in robotics. In *Proceedings of the 3rd international conference on applications of intelligent systems* (pp. 1–6).

Fernandez-Chaves, D., Ruiz-Sarmiento, J., Petkov, N., & Gonzalez-Jimenez, J. (2021b). Vimantic, a distributed robotic architecture for semantic mapping in indoor environments. *Knowledge-Based Systems*, *232*, Article 107440. http://dx.doi.org/10.1016/j.knosys.2021.107440.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, *32*(11), 1231–1237.

Gerkey, B., Vaughan, R. T., & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. *Vol. 1*, In *Proceedings of the 11th international conference on advanced robotics* (pp. 317–323). Citeseer.

González-Jiménez, J., Galindo, C., & Ruiz-Sarmiento, J. (2012). Technical improvements of the Giraff telepresence robot based on users' evaluation. In *2012 IEEE RO-MAN: The 21st IEEE international symposium on robot and human interactive communication* (pp. 827–832). IEEE.

Haskins, A. J., Mentch, J., Botch, T. L., & Robertson, C. E. (2020). Active vision in immersive, 360 real-world environments. *Scientific Reports*, *10*(1), 1–11.

Hu, Y., & Meng, W. (2016). RosUnitySim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning. *SIMULATION*, *92*(10), 931–944.

Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *Technical Report 07-49*, Amherst: University of Massachusetts.

Jaenal, A., Moreno, F.-A., & Gonzalez-Jimenez, J. (2021). Appearance-based sequential robot localization using a patchwise approximation of a descriptor manifold. *Sensors*, *21*(7), 2483.

Josifovski, J., Kerzel, M., Pregizer, C., Posniak, L., & Wermter, S. (2018). Object detection and pose estimation based on convolutional neural networks trained with synthetic data. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 6269–6276). IEEE.

Juliani, A., Berges, V.-P., Vckay, E., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

Kang, K., Belkhale, S., Kahn, G., Abbeel, P., & Levine, S. (2019). Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)* (pp. 6008–6014). IEEE.

Koenig, N., & Howard, A. (2004a). Design and use paradigms for gazebo, an open-source multi-robot simulator. *Vol. 3*, In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)* (pp. 2149–2154). IEEE.

Koenig, N., & Howard, A. (2004b). Design and use paradigms for Gazebo, an open-source multi-robot simulator. *Vol. 3*, In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS) (IEEE Cat. No.04CH37566)* (pp. 2149–2154). http://dx.doi.org/10.1109/IROS.2004.1389727.

Li, Q., Zhu, J., Liu, J., Cao, R., Fu, H., Garibaldi, J. M., et al. (2020). 3D map-guided single indoor image localization refinement. *ISPRS Journal of Photogrammetry and Remote Sensing*, *161*, 13–26.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft COCO: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Linowes, J. (2015). *Unity virtual reality projects*. Packt Publishing Ltd.

Martins, G. S., Ferreira, J. F., Portugal, D., & Couceiro, M. S. (2019). Modsem: Towards semantic mapping with distributed robots. In K. Althoefer, J. Konstantinova, & K. Zhang (Eds.), *Towards autonomous robotic systems* (pp. 131–142). Cham: Springer International Publishing.

Miller, A. H., Feng, W., Fisch, A., Lu, J., Batra, D., Bordes, A., et al. (2017). Parlai: A dialog research software platform. arXiv preprint arXiv:1705.06476.

Naseer, M., Khan, S., & Porikli, F. (2018). Indoor scene understanding in 2.5/3d for autonomous agents: A survey. *IEEE Access*, *7*, 1859–1887.

Navarro, F., Fdez, J., Garzón, M., Roldán, J. J., & Barrientos, A. (2018). Integrating 3D reconstruction and virtual reality: A new approach for immersive teleoperation. *Vol. 694*, In *Advances in intelligent systems and computing* (pp. 606–616). Springer Verlag, http://dx.doi.org/10.1007/978-3-319-70836-2_50.

Pan, X., You, Y., Wang, Z., & Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. arXiv preprint arXiv:1704.03952.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., et al. (2018). VirtualHome: SImulating household activities via programs. Cs.Cv, arXiv:1806.07011v1.

Qiu, W., & Yuille, A. (2016). Unrealcv: Connecting computer vision to unreal engine. In *European conference on computer vision* (pp. 909–916). Springer.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009). Ros: an open-source robot operating system. *Vol. 3*, In *ICRA workshop on open source software* (3.2), (p. 5). Kobe, Japan.

Roldán, J. J., Peña-Tapia, E., Garzón-Ramos, D., de León, J., Garzón, M., del Cerro, J., et al. (2019). Multi-robot systems, virtual reality and ROS: Developing a new generation of operator interfaces. *Vol. 778*, In *Studies in computational intelligence* (pp. 29–64). Springer Verlag, http://dx.doi.org/10.1007/978-3-319-91590-6_2.

Ruiz-Sarmiento, J. R., Galindo, C., & González-Jiménez, J. (2015). Exploiting semantic knowledge for robot object recognition. *Knowledge-Based Systems*, *86*, 131–142.

Ruiz-Sarmiento, J. R., Galindo, C., & González-Jiménez, J. (2017). Robot@home, a robotic dataset for semantic mapping of home environments. *International Journal of Robotics Research*, *36*(2), 131–141. http://dx.doi.org/10.1177/0278364917695640.

Ruiz-Sarmiento, J.-R., Galindo, C., Monroy, J., Moreno, F.-A., & Gonzalez-Jimenez, J. (2019). Ontology-based conditional random fields for object recognition. *Knowledge-Based Systems*, *168*, 100–108.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., & Funkhouser, T. (2017). Semantic scene completion from a single depth image. In *Proceedings of 30th IEEE conference on computer vision and pattern recognition*.

Straub, D., & Rothkopf, C. A. (2021). Looking for image statistics: Active vision with avatars in a naturalistic virtual environment. *Frontiers in Psychology*, *12*, 431.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., et al. (2019). The replica dataset: A digital replica of indoor spaces. arXiv arXiv:1906.05797.

Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., et al. (2016). Torchcraft: a library for machine learning research on real-time strategy games. arXiv preprint arXiv:1611.00625.

Taira, H., Rocco, I., Sedlar, J., Okutomi, M., Sivic, J., Pajdla, T., et al. (2019). Is this the right place? Geometric-semantic pose verification for indoor visual localization. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*.

Tian, Y., Gong, Q., Shang, W., Wu, Y., & Zitnick, C. L. (2017). Elf: An extensive, lightweight and flexible research platform for real-time strategy games. arXiv preprint arXiv:1707.01067.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033). IEEE.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., et al. (2017). Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2.

Wu, Y., Wu, Y., Gkioxari, G., & Tian, Y. (2018). Building generalizable agents with a realistic and rich 3D environment. ICLR, arXiv:1801.02209v1.

Xiao, J., Owens, A., & Torralba, A. (2013). SUN3D: A database of big spaces reconstructed using SfM and object labels. In *Computer vision (ICCV), 2013 IEEE international conference on* (pp. 1625–1632).

Zhao, W., Queralta, J. P., & Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)* (pp. 737–744). IEEE.

Zheng, J., Zhang, J., Li, J., Tang, R., Gao, S., & Zhou, Z. (2020). Structured3D: A large photo-realistic dataset for structured 3D modeling. Cs.Cv, arXiv:1908.00222v3.

Zuñiga-Noël, D., Ruiz-Sarmiento, J.-R., & Gonzalez-Jimenez, J. (2019). Intrinsic calibration of depth cameras for mobile robots using a radial laser scanner. In *International conference on computer analysis of images and patterns* (pp. 659–671). Springer.