technologies

Article

A Simulated Environment for Robot Vision Experiments †

Christos Sevastopoulos [1,*], Stasinos Konstantopoulos [2], Keshav Balaji [1], Mohammad Zaki Zadeh [1] and Fillia Makedon [1]

[1] Department of Computer Science and Computer Engineering, University of Texas at Arlington, Arlington, TX 76019, USA; Keshav.Balaji@mavs.uta.edu (K.B.); Mohammad.Zakizadehgharie@mavs.uta.edu (M.Z.Z.); Makedon@uta.edu (F.M.)
[2] Institute of Informatics and Telecommunications, National Centre for Scientific Research 'Demokritos', 15341 Agia Paraskevi, Greece; konstant@iit.demokritos.gr
* Correspondence: Christos.Sevastopoulos@mavs.uta.edu
† This paper is an extended version of our paper published in PETRA 2021, doi:10.1145/3453892.3462214.

Abstract: Training on simulation data has proven invaluable in applying machine learning in robotics. However, when looking at robot vision in particular, simulated images cannot be directly used no matter how realistic the image rendering is, as many physical parameters (temperature, humidity, wear-and-tear in time) vary and affect texture and lighting in ways that cannot be encoded in the simulation. In this article we propose a different approach for extracting value from simulated environments: although neither of the trained models can be used nor are any evaluation scores expected to be the same on simulated and physical data, the conclusions drawn from simulated experiments might be valid. If this is the case, then simulated environments can be used in early-stage experimentation with different network architectures and features. This will expedite the early development phase before moving to (harder to conduct) physical experiments in order to evaluate the most promising approaches. In order to test this idea we created two simulated environments for the Unity engine, acquired simulated visual datasets, and used them to reproduce experiments originally carried out in a physical environment. The comparison of the conclusions drawn in the physical and the simulated experiments is promising regarding the validity of our approach.

Keywords: robot perception; machine learning; traversability estimation

Citation: Sevastopoulos, C.; Konstantopoulos, S.; Balaji, K.; Zaki Zadeh, M.; Makedon, F. A Simulated Environment for Robot Vision Experiments. *Technologies* **2022**, *10*, 7. https://doi.org/10.3390/technologies10010007

Academic Editor: Tomohiro Fukuda

Received: 30 November 2021
Accepted: 5 January 2022
Published: 12 January 2022

1. Introduction

Deep neural networks in general, and *deep vision* in particular, have become not only a viable machine learning methodology but, in fact, an extremely successful one due to the recent availability of large-scale datasets and the computational power to process them. Robot vision has undoubtedly also received a major boost, but robot vision is more than computer vision that happens to be applied to a robotics application, because robots are able to supervise themselves by posteriorly discovering labels for earlier input via their interaction with their environment. This important aspect of robot vision cannot be served by static digital collections. However, the related research depends on conducting physical experiments, which are more time- and effort-consuming, which makes it difficult to accumulate the data volumes needed to fully exploit advances in deep learning.

On the other hand, training on *simulation* data has proven invaluable in applying machine learning in robotics, mostly dexterous control and navigation. As noted by Zhao et al. [1] in their recent survey, simulated images cannot be directly used to train vision systems no matter how realistic the image rendering is, as many physical parameters (temperature, humidity, and wear-and-tear in time) vary and affect texture and lighting in ways that cannot be encoded in the simulation. What they note as a promising research path is the simulated variability of the visual parameters, pushing the model to be general enough to also include real-world data.

In this article we present a concept that combines these two ideas: robots are able to improve their vision models by interacting with the environment and, at the same time, can expedite this process by assuming a performant initial model trained in a simulated environment. We first briefly discuss *traversability estimation and novelty detection* to establish that these robot vision tasks are prime examples of tasks where a robot can improve through its interaction with the environment (Section 2). We then present the two simulated environments we developed, designed to support experiments on these tasks (Section 3). Finally, we present a preliminary experiment designed to build confidence in the validity of experiments conducted in these environments (Section 4) and conclude (Section 5).

2. Background and Motivation

Traversability estimation [2] is a key technology in field robotics, as it allows robots to extract from sensory input the occupancy and cost information that is typically used by obstacle avoidance and navigation algorithms. Traversability estimation can be *appearance-based*, extracting from the visual signal terrain features such as roughness, slope, discontinuity and hardness [3] or terrain classes such as soil, grass, asphalt, and vegetation [4]; or it can be *geometry-based* extracting terrain features such as roughness, slope, and discontinuity from *digital elevation maps (DEM)* obtained from stereoscopic or depth sensors [5].

Even the more detailed appearance-based methods, however, only have broad classes of terrain (such as 'grass' or 'vegetation') and the features they extract cannot make the distinction between vegetation or foliage that a robotic platform can safely push away and harder obstacles that the platform should circumvent. Applying deep learning, as opposed to conventional machine learning and AI methods, has proven very effective in general engineering and robotics tasks [6,7]. Specifically for traversability estimation, methods that use deep vision to directly estimate traversability from raw images [8] can be trained to make the distinctions that are appropriate for each platform, but require training data specifically acquired and annotated for each platform. Our own earlier experiments [9] validated the accuracy of labels autonomously collected by a physical platform that annotated obstacles as compliant or hard obstacles by observing its IMU while trying to push them. Naturally, data collected in this manner was too sparse to train a deep vision system, but was sufficient for improving the accuracy of a general-purpose deep vision system by training the final classification layers into a task-specific binary classifier.

At the same time, using simulated data to fine-tune deep vision systems is not a new concept, both generally [10] and specifically in off-road navigation: Sharma et al. [11] used the MSU Autonomous Vehicle Simulator (MAVS) [12] to fine-tune the DeconvNet semantic segmentation network. MAVS provides realistic visual data that Sharma et al. proved adequate for improving DeconvNet accuracy when validated on real data, but does not integrate well with ROS software stacks. This restricts the simulated robot to data collected from predefined routes that the robot cannot affect. As the authors note, an oversimple or too random dataset can cause negative transfer, it is therefore important to be able to acquire data based on the robot's realistic operational scenarios.

This can be achieved using simulators that are better integrated with robotics software, such as the Gazebo/ODE robot and physics simulator [13]. Chavez-Garcia et al. [14] used Gazebo/ODE to train a network that directly maps range data to traversability estimation. However, range data will only learn geometry-based traversability and is unable to distinguish between compliant and hard obstacles. Since Gazebo does not generate realistic visual data, it does not support transfer learning of deep vision networks.

A similar picture can be seen in the anomaly detection for inspection tasks. Simulated experiments are mostly based on anomalies that can be detected in the 3D data [15,16]. Having said that, a recent breakthrough in applying deep vision for anomaly detection uses a semi-simulated approach: Zavrtanik et al. [17] trained the network using simulated visual anomalies on physical images, and showed that it is more efficient to learn the extent of deviation rather than to model the visual appearance of normality or anomaly. Similarly, Defard et al. [18] used a pre-trained convolutional neural network (CNN) and multivariate

Gaussian distributions to obtain a probabilistic representation of the normal class from simulated data. Defard et al. used real images in their experiments, but their findings strongly corroborate the findings by Zavrtanik et al. on the efficiency of approaching anomaly detection as a one-class learning task where data is used to find the extent of deviation from the normal distribution that constitutes an anomaly.

The above motivated the development of the simulated environments presented here. Specifically:

- We chose traversability estimation as an appealing task due to the fact that interesting properties of the environment (most prominently obstacle compliance) cannot be estimated geometrically from the range data that is available in Gazebo; while at the same time the ability to integrate with autonomous navigation is important so that the robot can supervise itself by attempting to traverse obstacles;
- We chose a vineyard spraying scenario as a characteristic of an application with compliant obstacles (foliage);
- We chose anomaly detection as an interesting fact because, again, some anomalies can only be identified visually, and there are strong indications in [19] that simulated data can contribute to the training of relevant deep vision systems;
- We chose an indoors industrial environment scenario as a means to complement the field robotics spraying scenario and additionally, owing to the fact that an industrial environment's visual features can relatively exhibit a uniform distribution in some extent, where discolorations and other visual anomalies are expected to be indicative of functional anomalies. Meaning, it would make more sense to implement the extent of deviation methods on a home or commercial setting, where there is very little visual uniformity in the first place.

Based on the aforementioned points, we developed the two simulated environments presented in this article as a means for experimenting with alternative deep network architectures in dynamic re-training scenarios where the robot learns as it interacts with the environment. By using the simulated environment during the development of both the self-supervision method and the architecture of the deep vision network, we aim to construct a relatively robust state of development before deployment on physical platforms and environments, where trial-and-error cycles require substantially more effort and time.

3. Simulated Environments

We developed our environments in Unity, a cross-platform 3D game engine integrated with a variety of plugins for both simulating physics and photorealistic image synthesis (See https://unity.com, accessed on 1 January 2022). In our experimental setup we used the Nvidia PhysX 3.3 physics engine (See https://docs.nvidia.com/gameworks/#gameworkslibrary/physx/physx.htm, accessed on 1 January 2022) to simulate the effect of collisions between the platform and various elements of the environment. Some of these elements are designed to easily yield without harming the platform (grass and thinner foliage) while others are rigid obstacles that should be avoided (rocks and tree trunks).

Unity renders both visual and range data from this environment, which can be used as simulated sensors for the Robot OS (ROS). In order to achieve this integration, it is necessary to convert between the ROS message format used by ROS components to exchange information and the JSON data interchange format [20] used by Unity. For this we used the Rosbridge suite (see http://wiki.ros.org/rosbridge_suite, accessed on 1 January 2022) to translate between ROS messages and JSON strings at the ROS side and the ROS# library (see https://github.com/siemens/ros-sharp, accessed on 1 January 2022) at the Unity side. In our setup ROS is running on an Ubuntu Virtual Machine and we initiated a Rosbridge server that provides a WebSocket transport layer between Unity and the ROS software stack, so that the latter subscribes to ROS topics such as pose, odometry, camera, etc., published by ROS#.

In our scenario, we simulate the autonomous motion of an All-Terrain Vehicle (ATV) navigating through the fields of a vineyard while encountering both non-traversable hard

obstacles and traversable soft obstacles. Different foliage, for example, will or will not be possible to push out of the way depending on the platform's mechanical properties. In other words, what would appear as an obstacle to standard ROS navigation using Point Cloud data might or might not be traversable. The robot can experiment with the simulated environment to acquire traversability ground-truth datapoints from the physics simulator.

3.1. Warehouse Environment

The industrial environment of a warehouse will be used for anomaly detection experiments for inspection tasks. It offers many capabilities for increasing its complexity. For instance, objects of different colour might exhibit different physical properties and thus traversability estimation should rely on successful detection. Dynamic obstacles, such as walking humans or other robots, can pose threats in determining traversable areas and safe paths to proceed. Our design (Figure 1), solely includes static objects (racks, barrels, tables, and fire-extinguishers, etc.) as a way to examine the fundamental properties of a safe indoor navigation.

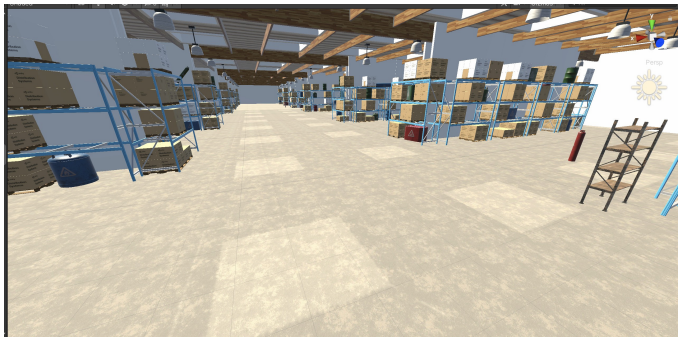

Figure 1. Scene of the indoor environment (warehouse).

3.2. Vineyard Environment

The vineyard environment will be used for traversability estimation experiments for spraying and inspection tasks. In particular, the simulated terrain displays several properties that are of interest for traversability estimation experiments (steepness, rocks, mud, foliage, tree leaves, trunks, and branches). Among these, in the experiments presented here we turn our attention to foliage (Figure 2). Specifically, similarly looking foliage might display radically different compliance and might or might not be traversable by a specific platform. This makes classifying foliage as traversable or not a task that is both challenging and very relevant, as being over-cautious can make a robot used for vineyard spraying and inspection very inefficient.

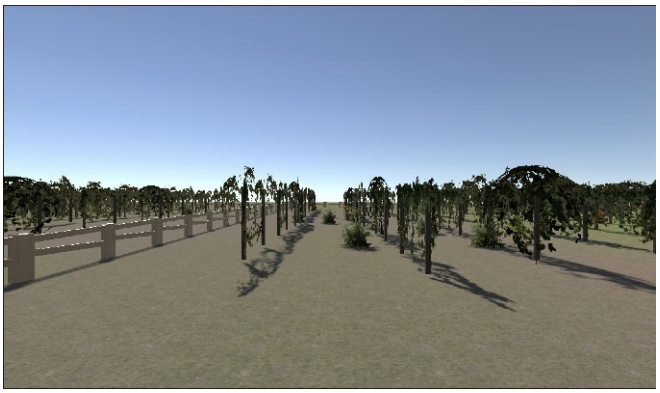

Figure 2. Scene of the outdoor environment (vineyard), showing compliant foliage and hard obstacles (tree trunks).

3.3. Dataset Collection

Both environments have been designed in a way that each comprises of two arenas, a positive and a negative one. What differentiates each arena is the amount of hindrance that objects exert on the robot's path.

We manually defined the robot's motion waypoints in areas that the robot should traverse to. Being able to go forward and turn at rates we specify, its motion resembles the one of a vines' spraying robot for the vineyard scenario and a patrol robot for the warehouse. Specifically, the waypoints' XYZ coordinates are assigned manually at specific locations we select and thus the Unity script that controls the robot's movement is responsible for driving the robot through each waypoint. In the case of navigating an obstacle-free path, the robot navigates at a constant velocity and captures images, at a constant rate, which are automatically collected and labelled as positive and thus the positive dataset is constructed. On the other hand, we manually place obstacles (foliage, barrels, etc.) at random locations close to the assigned waypoints, rendering them as potential threats for the robot to decrease its speed as long as it senses their presence in its vicinity. To be more precise, certain obstacles were placed in a manner that do not overlap with the waypoints and hence the robot is safe to proceed but it is still able to identify their existence. Additionally, some objects were intentionally placed at a specific location that obstructs the robot's path in order to generate negative datapoints (Figure 3).

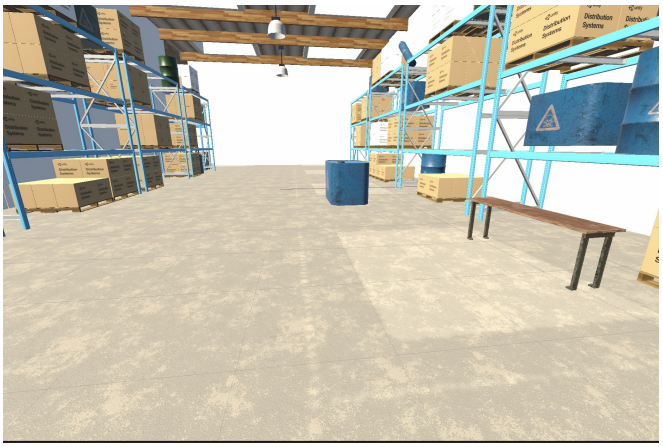

Figure 3. The blue object (tank) interferes directly with the trajectory's waypoints in contrast to the brown table.

We spawned the robot at random locations, and collected images that were annotated based on whether the robot was navigating the positive or the negative arena. In parallel, we automated a front-view screenshot collection, for both the positive and negative environments, that stores the images directly in a folder as png images and therefore expedites our building of positive and negative datasets.

4. Simulated Experiment

We conducted an experiment in order to evaluate the validity of conclusions reached from the simulated environment. In this experiment, we reproduced the experimental section of a prior publication where deep learning was used to learn a traversability estimation model. In the original experiments data were collected from a physical indoor environment and the experiments were used to evaluate different variations in the proposed method.

Should we reach the same conclusions using our simulated environment, then we can claim that our environment can be convincingly used in early-stage investigations of deep learning robot perception methods. This will facilitate the development phase before moving to (harder to conduct) physical experiments in order to evaluate the most promising approaches.

4.1. Neural Network Architecture

Our experiment replicates the network architecture proposed by Hirose et al. [21,22], namely a Deep Convolutional Generative Adversarial Network (DCGAN) ensemble of two adversarial modules, i.e., a generator (Gen) and a discriminator (Dis), along with a third network, the inverse generator (Gen^{-1}). The inverse generator is trained on the real image dataset and outputs the value of the latent vector z that is fed as an input to the Generator (Figure 4). All three networks (Gen, Dis, Gen^{-1}), are built as standard CNN which are trained simultaneously in an unsupervised manner. Three tensors R, D, and F are extracted from the discriminator and used as features for an additional classifier layer that decides if the scene encountered by the robot is GO or NOT GO. Specifically:

- R corresponds to the Residual Loss and is expressed as the pixel-wise absolute difference between real and generated images;
- D corresponds to the Discriminator Loss and is expressed as the absolute difference between the last convolutional layer features of the discriminator applied to the real data, and the same layer features applied to the generated images;
- F corresponds to the Discriminator feature expressed as the last convolutional layer features of the discriminator applied to the real data.

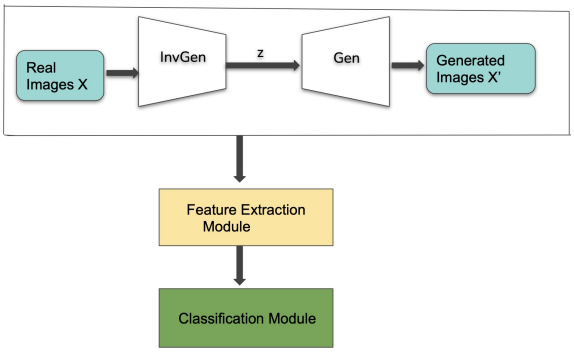

Figure 4. Overview of the pipeline.

These tensors are reshaped into vectors and concatenated into the classifier input. The classifier is a fully convolutional (FC) layer trained in a supervised way on a small amount of both positive and negative annotated images. The FC layer is trained to output the final GO or NOT GO decision (Figure 5).

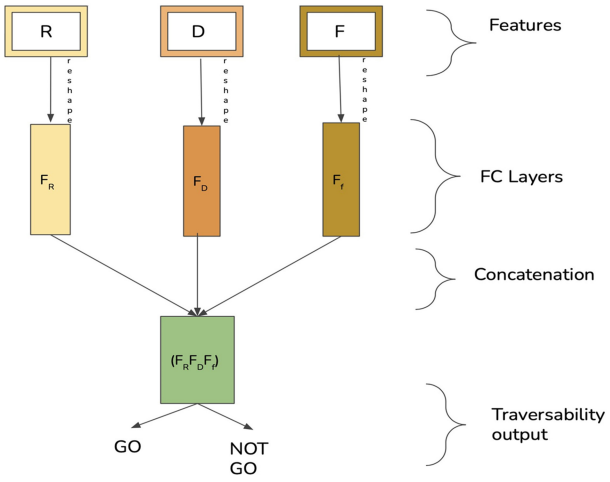

Figure 5. The Residual Loss (R), Discriminator Loss (D), and Discriminator Feature (F) tensors are reshaped and concatenated before being used as inputs for the fully convolutional classifier.

4.2. Training

We collected images through the robot autonomous navigation in Unity and randomly split the runs between training and testing. For both the indoor (Figure 6) and outdoor (Figure 7) scenario the resulting dataset, collected in a self-supervised manner, consists of approximately 17,000 positive training images. On top of that, an additional 1% of manually labelled negative images, along with an equal number of positive ones arbitrarily selected from the large dataset, was used in order to train and test the FC layer. We conducted our experiments on an Intel(R) Core(TM) i7-5820K CPU @ 3.30 GHz, using Nvidia Cuda 11.4 GPU. Using the Pytorch Framework and Python 3.8.10, we trained for a total of 60 epochs using a learning rate of 0.00001, Adam optimizer with beta1 = 0.85.

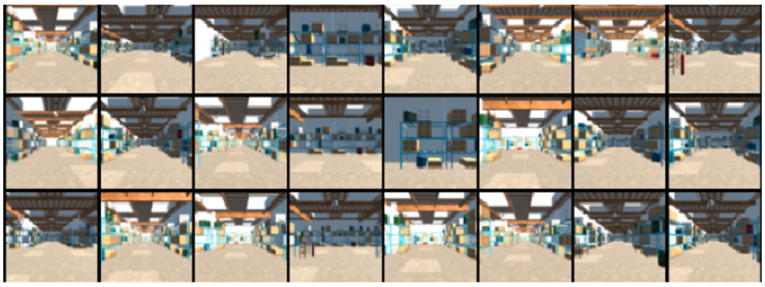

Figure 6. Training images (warehouse).

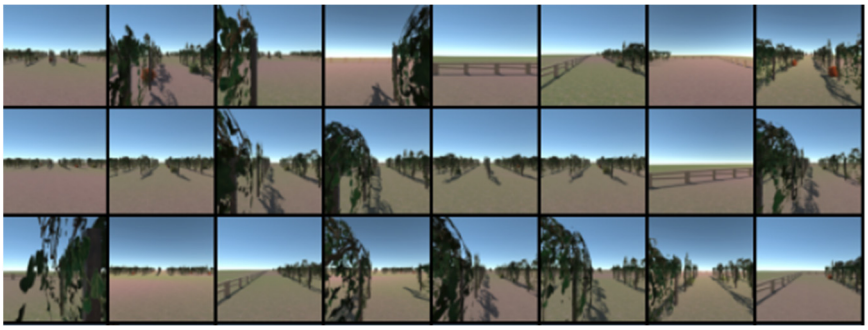

Figure 7. Training images (vineyard).

4.3. Results and Discussion

We evaluated the environment's suitability by assessing each network on the test datasets. Specifically, in order to measure the performance of each network, we used accuracy, recall, precision, and F1 score as metrics (Table 1).

The following points support our claim that the same conclusions are reached about the method from the simulated and the physical experiments:

- The F feature is the one that performs best by itself, and R is the feature that performs worst, although the differences are more pronounced in the simulated experiments than in the physical experiments. The simulated experiments rank the features correctly, but provide an exaggerated impression of their performance relative to each other;
- Among the two-features models, those that include F perform better than the one that does not. This observation conforms to the physical experiment, although, similarly to above, the differences are more distinct in the simulated experiments;
- In the RDF model, recall is higher than precision, pointing to the direction of making the model more specific in order to improve accuracy. This observation is accordant with the physical experiment;
- Regarding the individual features, both the simulated and the physical experiments show that D and F need to be more specific.

On the other hand, the following points run counter to our above-mentioned claim:

- In the vineyard scenario, dropping the D feature increases the levels of performance, although in the physical data all features provide some information and the RDF model is the most performant one;
- Among the two-features models, those that exclude R perform better than the one that does not. This observation is not consistent with the physical experiment, where the experiment that excludes D performs equally well as the one that excludes R;
- Regarding the precision and recall of the R feature, the physical experiments show that R needs to be generalized to increase recall, but the simulated experiments provide mixed results.

Table 1. Results on the test dataset using networks of different variations for the simulated environments.

Features	Accuracy [%]	Recall [%]	Precision [%]	F1 Score
Vineyard Environment				
R + D + F	90	97	85	91
R + D	75	76	74	75
D + F	84	86	85	84
R + F	96	95	95	95
R	51	53	76	36
D	85	85	84	86
F	95	92	91	93
Warehouse environment				
R + D + F	94	94	90	92
R + D	79	80	81	79
D + F	88	91	87	90
R + F	90	90	90	92
R	53	51	41	34
D	80	80	78	79
F	93	93	91	92
Physical experiment, as reported by Hirose et al. [21]				
R + D + F	94.25	95.75	92.96	94.33
R + D	91.63	94.00	89.74	91.81
D + F	93.00	96.50	90.19	93.24
R + F	93.13	95.00	91.56	93.25
R	85.38	83.50	86.75	85.10
D	91.63	94.50	89.36	91.86
F	92.25	95.50	89.67	92.49

Overall, we notice that our results are in relative accordance with the results obtained using the physical platform and the highest accuracy was achieved using the ensemble of RDF features for the warehouse environments which is more abundant in features than the vineyard. Inevitably, some limitations are faced due to the conventionalized nature of the simulated environment. Enriching the environment with more elements of physics and stochasticity as well as integration with a physical platform constitute the main avenues of our future research.

5. Conclusions and Future Work

We created two simulated indoor and outdoor Unity environments. The Unity/ROS framework presented here combines ROS integration with the generation of realistic visual streams, allowing simulated robots to acquire realistic visual data while moving in the environment in dynamically computed routes dictated by their operational requirements. We used this environment to train deep vision systems that classify objects detected in point cloud data. In the outdoor environment experiment foliage is classified as a 'compliant' vs. 'lethal' obstacle for the purpose of traversability estimation. In the indoor environment experiment objects are classified as 'expected' vs. 'novel'.

Through these experiments we aimed to test our main hypothesis that the simulated environments provide a solid groundwork for robot vision problems. Specifically, we reproduced an experiment originally conducted in a physical environment [21] in order to establish whether the conclusions reached by analysing the results of the simulated experiments are the same as the conclusions drawn by analysing the results of the physical experiments. We found that our approach is valid for the purpose of feature selection, as the relative ranking of the models using different mixtures of features is consistent between the simulated and the physical experiments.

This conclusion allows us to include our simulated environments in our future research plans as a testbed before carrying out physical experiments. Besides expediting the research processes, this plan will also allow us to verify the environments on more experiments and build more confidence on the validity of conclusions drawn from simulated experiments. On this front, besides enriching the environment with more elements, we will also compare the results obtained using Unity with results obtained using other simulation environments, more commonly used in robotics research. Although Unity is unquestionably one of the most realistic simulators available when it comes to synthesising visual signal, it will be relevant to quantify and understand the effect of the level or image realism offered by Unity.

Author Contributions: Conceptualization, C.S.; Methodology, C.S.; Software, K.B. and M.Z.Z.; Supervision, S.K. and F.M.; Writing—review & editing, C.S. and S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780265 (http://www.esmera-project.eu, accessed on 1 January 2022) through cascade funding to the *iRTA* project (https://irta.agenso.gr, accessed on 1 January 2022).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used for these experiments are available at https://doi.org/10.5281/zenodo.5732281 (Vineyard) and https://doi.org/10.5281/zenodo.5731646 (Warehouse). The Unity environments used to acquire these datasets are available at https://github.com/ChristosSev/Vineyard_G and https://github.com/ChristosSev/Warehouse_G. All URLs accessed on 1 January 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020. [CrossRef]
2. Langer, D.; Rosenblatt, J.; Hebert, M. A behavior-based system for off-road navigation. *IEEE Trans. Robot. Autom.* **1994**, *10*, 776–783. [CrossRef]
3. Howard, A.; Seraji, H.; Tunstel, E. A rule-based fuzzy traversability index for mobile robot navigation. In Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001), Seoul, Korea, 21–26 May 2001; Volume 3, pp. 3067–3071.
4. Angelova, A.; Matthies, L.; Helmick, D.; Perona, P. Fast terrain classification using variable-length representation for autonomous navigation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, USA, 17–22 June 2007.
5. Wermelinger, M.; Fankhauser, P.; Diethelm, R.; Krüsi, P.; Siegwart, R.; Hutter, M. Navigation Planning for Legged Robots in Challenging Terrain. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, Korea, 9–14 October 2016.
6. Beruvides, G.; Quiza, R.; Rivas, M.; Castaño, F.; Haber, R.E. Online detection of run out in microdrilling of tungsten and titanium alloys. *Int. J. Adv. Manuf. Technol.* **2014**, *74*, 1567–1575. [CrossRef]
7. Mohammed, W.M.; Nejman, M.; Castaño, F.; Martinez Lastra, J.L.; Strzelczak, S.; Villalonga, A. Training an Under-actuated Gripper for Grasping Shallow Objects Using Reinforcement Learning. In Proceedings of the 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS 2020), Tampere, Finland, 10–12 June 2020. [CrossRef]
8. Adhikari, S.P.; Yang, C.; Slot, K.; Kim, H. Accurate Natural Trail Detection Using a Combination of a Deep Neural Network and Dynamic Programming. *Sensors* **2018**, *18*, 178. [CrossRef] [PubMed]

9. Sevastopoulos, C.; Oikonomou, K.M.; Konstantopoulos, S. Improving Traversability Estimation through Autonomous Robot Experimentation. In Proceedings of the 12th International Conference on Computer Vision System (ICVS 2019), Thessaloniki, Greece, 23–25 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11754.
10. Bousmalis, K.; Irpan, A.; Wohlhart, P.; Bai, Y.; Kelcey, M.; Kalakrishnan, M.; Downs, L.; Ibarz, J.; Pastor, P.; Konolige, K.; et al. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA 2018), Brisbane, QLD, Australia, 21–25 May 2018.
11. Sharma, S.; Ball, J.E.; Tang, B.; Carruth, D.W.; Doude, M.; Islam, M.A. Semantic Segmentation with Transfer Learning for Off-Road Autonomous Driving. *Sensors* **2019**, *19*, 2577. [CrossRef] [PubMed]
12. Hudson, C.R.; Goodin, C.; Doude, M.; Carruth, D.W. Analysis of Dual LIDAR Placement for Off-Road Autonomy Using MAVS. In Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Kosice, Slovakia, 23–25 August 2018.
13. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), Sendai, Japan, 28 September–2 October 2004. [CrossRef]
14. Chavez-Garcia, R.O.; Guzzi, J.; Gambardella, L.M.; Giusti, A. Learning Ground Traversability from Simulations. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1695–1702. [CrossRef]
15. Aleksi, I.; Kraus, D.; Hocenski, Ž.; Keser, T. Simulated surface anomaly detection in underwater object reconstruction. In Proceedings of the 33rd Conference on Automation in Transportation (KoREMA 2013), Osijek, Croatia, 20–23 November 2013.
16. Martin, R.A.; Blackburn, L.; Pulsipher, J.; Franke, K.; Hedengren, J.D. Potential Benefits of Combining Anomaly Detection with View Planning for UAV Infrastructure Modeling. *Remote Sens.* **2017**, *9*, 434. [CrossRef]
17. Zavrtanik, V.; Kristan, M.; Skoc˘aj, D. DRÆM—A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2021), Online, 10–17 October 2021.
18. Defard, T.; Setkov, A.; Loesch, A.; Audigier, R. PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization. In Proceedings of the Pattern Recognition. ICPR International Workshops and Challenges, Virtual Event, 10–15 January 2021; Volume 12664. [CrossRef]
19. Richter, C.; Roy, N. Safe visual navigation via deep learning and novelty detection. In Proceedings of the Robotics: Science and Systems XIII (RSS 2017), Cambridge, MA, USA, 12–16 July 2017. [CrossRef]
20. Technical Committee TC39. *The JSON Data Interchange Syntax*, 2nd ed.; Technical Report 404; ECMA: Geneva, Switzerland, 2017.
21. Hirose, N.; Sadeghian, A.; Goebel, P.; Savarese, S. To go or not to go? A near unsupervised learning approach for robot navigation. *arXiv* **2017**, arXiv:1709.05439.
22. Hirose, N.; Sadeghian, A.; Vázquez, M.; Goebel, P.; Savarese, S. Gonet: A semi-supervised deep learning approach for traversability estimation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3044–3051.