applied sciences

MDPI

Article

Occlusion Avoidance in a Simulated Environment Using Reinforcement Learning

Márton Szemenyei and Mátyás Szántó *

Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
* Correspondence: mszanto@iit.bme.hu

Abstract: Neural network-based solutions have revolutionized the field of computer vision by achieving outstanding performance in a number of applications. Yet, while these deep learning models outclass previous methods, they still have significant shortcomings relating to generalization and robustness to input disturbances, such as occlusion. Most existing methods that tackle this latter problem use passive neural network architectures that are unable to act on and, thus, influence the observed scene. In this paper, we argue that an active observer agent may be able to achieve superior performance by changing the parameters of the scene, thus, avoiding occlusion by moving to a different position in the scene. To demonstrate this, a reinforcement learning environment is introduced that implements OpenAI Gym's interface, and allows the creation of synthetic scenes with realistic occlusion. The environment is implemented using differentiable rendering, allowing us to perform direct gradient-based optimization of the camera position. Moreover, two additional methods are also presented, one utilizing self-supervised learning to predict occlusion segments, and optimal camera positions, while the other learns to avoid occlusion using Reinforcement Learning. We present comparative experiments of the proposed methods to demonstrate their efficiency. It was shown, via Bayesian *t*-tests, that the neural network-based methods credibly outperformed the gradient-based avoidance strategy by avoiding occlusion with an average of 5.0 fewer steps in multi-object scenes.

Keywords: computer vision; object detection; differentiable rendering; self-supervised learning; neural networks; reinforcement learning

check for updates

Citation: Szemenyei, M.; Szántó, M Occlusion Avoidance in a Simulated Environment Using Reinforcement Learning. *Appl. Sci.* **2023**, *13*, 3090. https://doi.org/10.3390/app13053090

Academic Editor: João M. F. Rodrigues

Received: 30 January 2023
Revised: 20 February 2023
Accepted: 22 February 2023
Published: 27 February 2023

1. Introduction

Object recognition is one of the fundamental problems in the field of machine vision, with deep neural networks achieving state-of-the-art performance in the last decade. These advances led to successes in numerous applications, such as autonomous vehicles, surveillance and mapping tasks, and industrial object detection. Nonetheless, in all the aforementioned fields, the reliability of these algorithms is paramount, and further research is necessary to increase generalization and robustness to various disturbances to ensure the applicability of these methods in real-world situations such as robotics applications or autonomous vehicle sensor systems.

The generalization capabilities of deep neural networks have been a major focus of recent research, resulting in substantial knowledge on the effect of training methods, efficient architectures or data augmentation strategies. Nonetheless, the robustness to various input disturbances has not been explored to the same degree. We argue that, due to the tendency of neural networks to overfit to data, these effects can affect performance significantly.

Due to the data augmentation strategies frequently used during the training of neural networks, some simple disturbances are relatively well-handled by these methods. These usually include various geometric and color transformations and random noise. However, some more complex environmental disturbances, such as rain, unique lighting conditions or fog are rarely investigated. The focus of our paper is partial occlusion, as it is one of the most significant and common sources of disturbance [1,2].

Naturally, the reason for the relative absence of these factors from deep learning training procedures is their cost; Acquiring an accurately labelled dataset containing the aforementioned natural disturbances would result in considerable difficulty. In order to resolve this issue, we propose using a high-quality synthetic dataset, generated using state-of-the-art rendering techniques, as that allows us to create high-quality, automatically labelled datasets containing ground truth object silhouettes and occluded areas with minimal costs.

Furthermore, most current object detection algorithms are passive methods, meaning that they use a single image to predict the location and classes of object in the scene, while they cannot perform actions to influence their environment. This limitation renders these methods unable to resolve some of these disturbances. Yet, humans are not brain-in-the-jar systems, as we are able to engage with our environment, creating what is called a Perception Action Loop (PAL). This allows us to combine information from multiple observations that were taken deliberately to maximize our understanding of the scene.

In this paper, a novel method is presented that is capable of identifying occlusion, and efficiently avoiding it by moving the camera to a new position and orientation. In order to achieve this, an OpenAI Gym-based reinforcement learning environment is developed that contains multiple objects in occlusion, and provides an RGB-D image for the agent. The environment also uses differentiable rendering to allow us to compute the derivative of the occlusion area with respect to the camera position (or other scene parameters). We show that, by using this gradient, the occlusion can be avoided using simple gradient-based optimization. Previous methods, e.g., Refs. [3–5], rely on an end-to-end neural network-based solution for occlusion detection, which do not allow for such an optimization-based occlusion-minimization.

We also extend our work [6] to propose two neural-network based methods that are able to resolve occlusion without relying on gradients extracted from the simulated environment. The first of such methods uses self-supervised learning to predict the occlusion map in the image, while it also predicts the gradient direction, using only an RGB-D image as input. The second method uses end-to-end reinforcement learning to avoid occlusion in a minimal amount of steps, using the same input. To demonstrate the effectiveness of the proposed learning methods, the Bayesian *t*-test was used on 50 random runs of the environment to show that these methods credibly outperformed the gradient method, as well as the random movement policy.

To summarize, our main contributions are the following:

1. A novel, open-source reinforcement learning environment is developed using PyTorch3D that allows the rendering of images that contain occluded objects. The environment uses differential rendering to provide researchers with a differentiable reward function measuring the amount of occlusion in the scene.
2. The usability of this environment is demonstrated by performing Stochastic Gradient Descent (SGD) optimization of the camera pose using this differential reward to successfully resolve occlusion in scenes with multiple objects.
3. A novel solution is presented for occlusion avoidance using only RGB-D images as input. This method is trained to predict occlusion maps and gradient directions using self-supervised learning by generating example scenes in the environment.
4. Furthermore, a reinforcement learning (RL) agent is also trained to successfully avoid occlusion in the environment between several objects. This agent also uses RGB-D images as input, and is trained to avoid occlusion in the minimal amount of steps.
5. Finally, the effectiveness of the different methods is evaluated and compared via a series of experiments. We show the superiority of the RL-based method using the Bayesian *t*-test.

In Section 2, the most relevant related works are reviewed, including methods for object detection under occlusion, as well as active perception methods. Then, in Section 3 the environment is presented, detailing the action and observation spaces, as well as the reward function. In Section 4 gradient-based optimization is presented, while in Section 5 neural network-based methods are described. Section 6 details the results of experimental evaluation, including the Bayesian *t*-test, while Section 7 summarizes our work, and details planned further research.

2. Previous Work

In this section, research connected to the topic of this paper is presented. First, works on object detection under partial occlusion are explored, then, in the latter part of this section, a selection of works that realize parts of a full perception–prediction–action loop are discussed in detail.

2.1. Object Detection in Occluded Scenes

Object detection is one of the fundamental problems in computer vision. In recent years, significant advances have been made in this field. Deep neural networks have revolutionized this field, starting with region-based methods [7–9]. Then, single-shot detectors, such as SSD or YOLO, provided significantly more efficient detectors [10–12], with their more advanced variants [13–15] achieving SOTA results on standard object detection tasks. Transformer-based methods [16] have also made advances recently, while, finally, anchor-free methods, such as CenterNet [17], aim to solve the problems of post-processing steps, such as Non-Maximum Suppression (NMS).

Still, mostf these detection methods do not directly address the problem of detecting input disturbances, especially partial occlusion, in a supervised manner. Instead, most works focus on improving their results when occlusion is present without explicitly detecting it. This is partially due to the fact that, to the best of our knowledge, no large-scale datasets exist, where occlusion labels (e.g., masks defining the exact overlap area between two object silhouettes) are available, making direct supervised learning of occlusion maps infeasible. Notably, two datasets that contain partially occluded objects have been released recently. These sets of images with occluded objects and the adhering ground truth masks, describing the regions of the individual objects, are entitled the Occluded COCO [18] and OVIS datasets [19]. However, these datasets do not contain occlusion masks, i.e., the exact overlap area between two object silhouettes. This property limits the usability of both datasets for machine learning-based occlusion map detection.

Object detection methods that aim to deal with partial occlusion need to rely on some form of self-supervised learning, meaning that the labels have to be generated automatically [20]. The most straightforward way of doing this is to generate synthetic occlusion in natural images, and train neural network-based methods to make correct predictions in spite of these artificial occlusions. Examples of this approach include Follmann et al. [3], where a semantic segmentation method is trained to predict the full object segments correctly, despite parts of the image being masked out. Zhan et al. [21] presented a generative approach. They trained a Generate Adversarial Network (GAN) to accurately restore masked-out parts of an image, thus, successfully restoring information lost by occlusion.

While the performance of these methods is superb, unfortunately, the use of synthetic occlusions introduces a critical weakness. Since the distribution of these masked-out image parts is vastly different from the distribution of a natural image, detecting the occluded parts becomes significantly easier, and, thus, the generalization capabilities of these methods in regard to natural occlusions are questionable.

This is not the case for Compositional Neural Networks (CompNets) [4,5], which are one of the most notable methods of object detection under partial occlusion. This method employs unsupervised learning to learn the distribution of occluding parts and distinguish these from the distributions of parts containing relevant objects. This allows the method to be trained in natural images containing natural occlusions, thus, avoiding the aforementioned problems.

While unsupervised learning of partial occlusion is impressive, we argue, however, that using supervised learning from annotated data might lead to superior performance. In a previous work, Pásztor [22] used a simulation created via the Unity game engine to render realistic images with natural occlusion. They also used the game engine to extract weak occlusion labels for training. The intersections of the object bounding boxes were labeled as occlusion. They then trained a semantic segmentation network to predict the occlusion masks directly from RGB images, and found—quite surprisingly—that the network had a tendency to learn the true occlusion mask, despite being only provided with the inaccurate mask.

2.2. *Active Detection Methods*

While the object detection methods detailed above have achieved considerable success, they also share a limitation. All these methods envision object detection as a passive task, where the method is presented in a single frame (or in some cases multiple frames from a video), and responds with a single prediction. Even in multi-view systems, the collection of observations is independent from the detection method. In reality, however, most natural object detection systems (i.e., people and animals) integrate the task of data collection and object detection, and the needs of the latter influence the former, as these agents are able to change the parameters of the observed scene. This way, these autonomous agents can derive more robust predictions by implementing the so-called Perception–Action Cycle (PAC).

However, to implement such an active observation procedure, every agent needs to combine two essential skills. First, the agent needs to understand and correctly predict the effect of its own actions on the environment. This can be achieved by employing some form of unsupervised or—more likely due to the nature of the problem—self-supervised learning. Notably, this relatively novel type of machine learning has been a major focus of research recently, and is already being used to predict realistic possible futures in frame sequences [23].

Second, the agent needs to be able to perform a sequence of actions that bring it to a desired state in a complex, interactive environment. This is usually accomplished via Reinforcement Learning (RL) [24,25], in which the agent is trained to maximize a reward function by performing action in the environment. This is usually a rather difficult problem due to difficulties, like the credit assignment problem and the exploration-exploitation dilemma. In summary, for the algorithm to be able to perform active object detection, it needs to combine all three forms of learning: reinforcement learning for determining actions, self-supervised for predicting their effects, and supervised for the object detection itself.

In the literature, there are a number of notable methods that combine the above types of learning in the same algorithm. For instance, Nair et al. [26] published a solution that combines reinforcement and self-supervised learning, where the agent imagines goals for itself, and derives an improved understanding of its environment by trying to achieve them. Similarly, curiosity-based reinforcement learning methods [27–29] include forward stepping prediction goals, while they also encourage agents to reach states, where their prediction is flawed, thereby facilitating the exploration of novel states. Finally, World Models [30] combines perception methods with prediction and action modules, coming relatively close to implementing a perception–action loop, although the goal of the method is different.

Notably, several methods exist in the literature that use frames from multiple views to perform object detection [31]. The skill of combining multiple information sources is a necessary component of active perception. It is worth mentioning that, in a previous work [32,33], we implemented a reinforcement learning environment including multiple cooperating agents that each had noisy partial observations. In this scenario, the communicating agents were rewarded for being able to identify scene objects accurately, and, therefore, they developed policies that took certain actions just to improve the agent's vision of the scene.

3. The RL Environment

In this section, the reinforcement learning environment is presented in detail, including its observation and action spaces, and the reward function. The environment itself is based on differentiable rendering, to create the scene and observations, Meta's PyTorch3D [34] framework is used. We chose to use PyTorch and PyTorch3D for this project mainly due to the frameworks' versatility and ease of use, as well as the enormous ecosystem of open-source projects, tools and knowledge transfer systems available. This library enables machine learning on 3D data, extending the popular PyTorch framework [35]. The framework provides advanced mesh storage and batching methods, essential for 3D computer vision research. Furthermore, PyTorch3D also includes several differentiable renderers, which enabled us not only to calculate a reward function, based on the number of occluded pixels, but also to compute the derivative of this reward with regard to the camera pose.

A single instance of the environment renders a scene containing three different objects, selected randomly from the ShapeNet dataset [36]. The objects are placed in different locations along the z axis. The first object is placed in the origin, the others are displaced in the negative direction by default values of $1/2$ and 2. The camera is placed 4 units away on the negative z axis. This creates a base setup that guarantees occlusion between objects in the scene.

To avoid trivial solutions, and to allow more disturbances, the latter two objects—i.e., the ones not placed in the origin—are offset from the z axis horizontally. The size of the displacement is randomly chosen from a standard normal distribution, such that the camera is always able to observe some occlusion between the objects. The horizontal offset of the two objects are equal in magnitude but are opposite to each other—i.e., if the object closer to the camera is offset to the right, then the second object is moved to the left. Figure 1 shows a few typical environment setups.

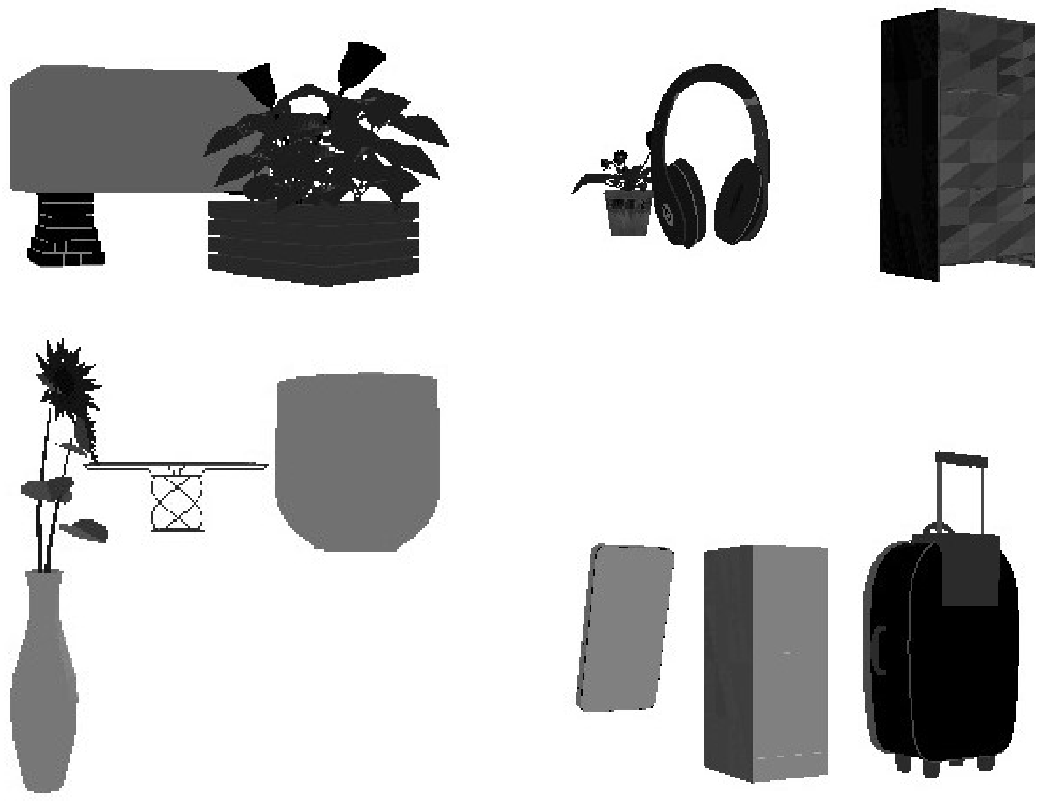


Figure 1. A few example setups of the environment. The camera position has been moved from the initial position for better visibility of the objects.

3.1. Action Space

As the goal of the environment is to train the agent to take a more advantageous position that ensures better visibility of the objects, the actions themselves are movements of the virtual camera used for rendering. Now, it is worth noting that if we intend to minimize the amount of occlusion measured in the image (using pixels or any other unit), then several trivial solutions present themselves. The agent could simply turn the camera away from the scene, thus ensuring no occlusion is seen by not seeing anything at all. Alternatively, the agent could just move the camera infinitely far away from the scene, thus reducing the visible objects to a size smaller than a single pixel, again minimizing the occlusion of the scene.

Thankfully, both trivial solutions are easily avoided by restricting the camera position to a spherical surface with a fixed radius, while ensuring that the camera is always in a look-at orientation. For this reason, the two actions the agent can take are to move the camera along this sphere using the azimuth (φ) and elevation (θ) angles. Since PyTorch3D expects the camera position to be given in the Cartesian coordinate system, the three parameters (φ, θ, and the radius ρ) have to be converted, and the look-at orientation has to be computed using PyTorch3D's built-in functions. The spherical-to-Cartesian conversion is calculated as follows:

$$x = \rho sin(\varphi)cos(\theta), \quad (1)$$

$$y = \rho sin(\varphi)sin(\theta), \quad (2)$$

$$z = \rho cos(\varphi). \quad (3)$$

The initial extrinsic camera parameters in the spherical coordinate system are given by the equations below:

$$\theta = 0^\circ, \quad (4)$$

$$\rho = 4, \quad (5)$$

$$\varphi \sim U(-\pi/32;\ \pi/32). \quad (6)$$

3.2. Observation Space

As stated previously, we intended to create an environment that allows end-to-end learning from images. However, having depth information would presumably improve the agent's performance, so rendering RGB-D images of the scene would be desirable. Due to the wide availability of commercial RGB-D cameras, this environment is not unrealistic. Depth images can be simply acquired from the Z buffers of the renderer. It is worth noting, however, that depth images generated this way are somewhat "too perfect", given that real depth cameras usually have gaps at object boundaries due to the parallax effect.

Once the new camera position is calculated, it is turned into Cartesian coordinates, and the look-at orientation is computed. Then, the camera pose is converted into a standard homogeneous format $[\mathbf{R}\ T]$, and, then, a Hard Flat Shader is used to render the RGB image. By default, the RGB and depth images are rendered at a 512×512 resolution, while other render settings are set to PyTorch3D's defaults.

During our experiments, we noticed that all of PyTorch3D's renderers tended to incorrectly calculate the maximum amount of triangle faces to use per bin, resulting in a final render with parts of certain more detailed objects missing. We solved this issue by automatically calculating a much higher limit for this variable, based on the number of triangles in the scene. However, this resulted in considerably higher render times in scenes that contained more complex geometries.

3.3. Reward Function

The final element of the environment is the reward function that is used for training the reinforcement learning agent. Here, our goal was to ensure that the agent found a pose that avoided all occlusion in as few steps as possible. To ensure this, in every step when the occlusion is not resolved, we give a small penalty (-0.2) to the agent. Once the occlusion has been successfully avoided, the agent receives a reward of 5.

However, we felt that giving only sparse rewards at the end of the optimization would result in subpar performance compared to using a dense reward system. By giving direct feedback to the RL agent after every step, the training procedure can establish the relationship between actions and rewards much more quickly. To do this, we compare the amount of occlusion before and after the action is taken, and add this to the reward.

However, this addition presents a new problem. The amount of occlusion is measured in pixels and has a variance that is several orders of magnitude higher than the fixed rewards introduced earlier. Moreover, the amount of occlusion varies significantly due to the size of the objects in the environment, which is out of the agent's control. To solve this problem, we normalize the size of this component with the size of the objects in the given scene. For our test. we investigated the following two normalization schemes:

1. Union-based normalization: the occlusion-based reward is divided by the total area of the object silhouettes in the first frame.
2. Intersection-based normalization: the occlusion-based reward is divided by the area of the occlusion in the first frame.

To summarize, the reward function is defined as:

$$R = L_{Occl}^{t-1} - L_{Occl}^{t} + P(L_{Occl}^{t}), \tag{7}$$

$$P(L_{Occl}^{t}) = \begin{cases} 5, & \text{if } L_{Occl}^{t} = 0 \\ -0.2, & \text{otherwise} \end{cases} \tag{8}$$

where L_{Occl}^{t} is the amount of occluded pixels in timestep t, and P is the additional penalty term.

4. Optimization

As described above, one of the main objectives of our work was the creation and training of an agent capable of avoiding occlusions and successfully observing all objects in the simulated environment. Since this was a rather complex task to be accomplished by an autonomous agent purely relying on reinforcement learning, we developed a model-based learning method, which enables the agent to predict the alterations of the occlusion map following an action, i.e., a camera movement. This method allows the agent to predict the consequences of its actions and plan ahead—thus, achieving much faster convergence.

We found, however, that a superb solution was also achievable. The differential rendering capability provided by PyTorch3D enables the differentiation of the occlusion map with regard to the camera movement. This attribute enables the direct prediction of the locally optimal camera movement using the RGB-D images. Nevertheless, we needed to investigate whether such a loss/reward function was viable.

To this end, we defined an optimization method that uses Gradient Descent (GD) to achieve zero occlusion—that is, we move the camera to a pose where the amount of occluded pixels becomes 0. As mentioned above, we wanted to prevent degenerate solutions, and, therefore, the camera positions were restricted to a spherical surface. The surface was defined by its non-variable radius. Hence, the only parameters that the agent was able to change during optimization were the elevation (θ) and azimuth (φ) angles. The solutions yielded by this optimization could still be optimal, as moving radially, i.e., directly towards, or outwards from, the center of the sphere does not yield the complete disappearance of occlusions.

The loss function was defined as follows:

1. The differential rendering engine accessible via the PyTorch3D-based environment enabled the rendering of the silhouettes for all the individual objects in the scene.
2. The full occlusion map was created by an element-wise multiplication of these images.
3. The sum of the pixels of the full occlusion map was calculated yielding the fully differentiable loss function.

Alternatively, the loss can be expressed in equation form as:

$$L_{Occl} = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} \sum Mask_i \bigodot Mask_j, \tag{9}$$

where N is the number of objects, $Mask_i$ is the silhouette of the Ith object, and $\bigodot$ is element-wise multiplication.

Note that, in the cases where one of the objects fully contained the other object, i.e., the latter was not visible to the agent, the derivative of the loss function evaluated to 0 and no additional information was earned. To forego this problem, we introduced a number of large-kernel smoothing operations on the individual silhouette images. This caused them to be translated into a form where the pixel values corresponded to their distances from the original object silhouettes. Using this modification, we were able to ensure smooth gradients. However, this problem was solved in PyTorch3D, since it already produces grayscale silhouette renders.

The extrinsic camera parameters, i.e., elevation and azimuth angles, were then optimized using the occlusion map and a learning rate of 10^{-6}. During optimization a common error scenario occurred when the output gradient of the render engine was NaN. We analyzed the occurrence of such errors, but found no evident pattern and concluded that the phenomenon resulted from within the PyTorch3D implementation of the engine. In order to avoid further complications resulting from NaN gradients, we introduced a workaround, which was that, whenever this occurred, the optimization was rerun after randomly perturbing the camera position.

5. Architecture and Training

In this section, the architecture and training procedure are presented. The goal of our neural network was to implement a strategy for occlusion avoidance, which would reduce the problem of occluded object detection to an occlusion-free scenario. This step is necessary, since the direct gradient-based optimization method is only utilizable in the original, synthetic environment. However, a CNN-based method does not require environment-calculated gradients, and only requires the input images for its functionality. This allows the below-described approach to be usable in real-world environments, given the right configuration. The Convolutional Neural Network (CNN) presented in this section was also trained to use the RGB-D images for predicting the occlusion map and gradient directions, as we predicted that these would aid the network in learning useful representations of the input.

5.1. Model Architecture

Since our goal with the CNN-based method was to predict the outputs for the images rendered in the environment, we designed a multi-head structure that, once properly trained using multi-task learning, would be capable of producing the occlusion masks, as well as the prediction for the environment gradients showing the optimal camera movement derived via differential rendering. The three parts of the architecture are:

1. a shared convolutional encoder,
2. a convolutional decoder, responsible for predicting the occlusion maps, and
3. a dense head predicting the gradient directions.

Our architecture is designed as a U-Net-like [37] structure, containing both an encoder and decoder subsystem Both the encoder and decoder subsystems are designed as U-Net-like structures. The resulting architecture makes use of 5 downscaling and upscaling blocks, each with 2 consecutive, 3 × 3 convolutional layers, followed by ReLU and BatchNorm. For downscaling and upscaling, strided and transposed convolution blocks were used, respectively. The initial channel count of 8 was doubled on every downscaling step, and skip connections between the encoder and decoder networks were utilized at every level. We designed and trained numerous different variations of these parts. We analyzed the impact of having residual connections in our convolutional blocks between the two parts in the R-UNet variant, and then introduced depth-wise separable convolution blocks in the SR-UNet version. We also tested the effect of atrous convolutions using the SAR-UNet variant. The proposed architecture is visualized in Figure 2.

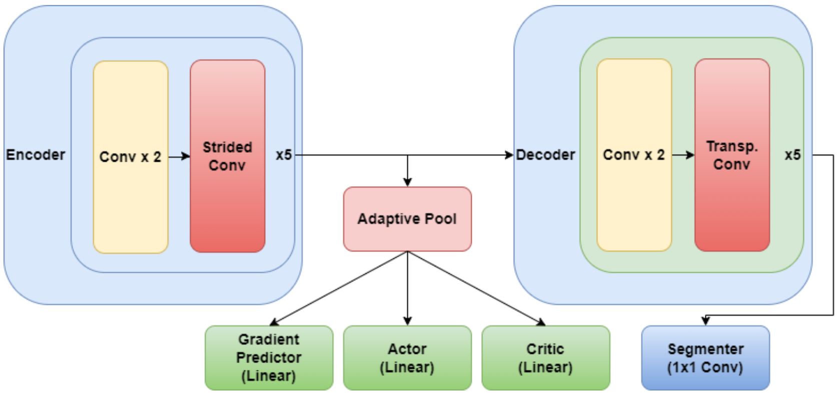


Figure 2. The neural network architecture. All variants used 3 × 3 convolution kernels, in the case of the atrous variant, dilation was set to 2. In the S variants, the 3 × 3 convolution was separated to a sequence of 3 × 1, 1 × 3 and 1 × 1 convolutions.

5.2. Pre-Training

We created a labeled dataset using the above-defined environment for performing pre-training. For the database, all the ShapeNet-provided object categories were used. A total of 20 images for 6500 random scenes were generated, which gave us a total of 130,000 256 × 256 RGB-D images with the corresponding gradient directions and occlusion masks. Moreover, the camera positions for each image were saved. The algorithm used for dataset generation is shown in Algorithm 1.

Algorithm 1 Dataset generation

```
Input:
# of scenes: $N_s$
# of images per scene: $N_i$
Output:
Dataset: $D$
 1: for $j$ in range($N_s$) do
 2:     Initialize Camera
 3:     for Object in range(3) do
 4:         Initialize Object $\rightarrow$ Objects
 5:     for $k$ in range($N_i$) do
 6:         Render Camera Image $\rightarrow I_{j,k}$
 7:         Calculate Occlusion Mask $\rightarrow M_{j,k}$
 8:         Calculate Camera Gradient via SGD $\rightarrow G_{j,k}$
 9:         Zip Data as $[I_{j,k}, M_{j,k}, G_{j,k}] \rightarrow D_{j,k}$
10:         Move Camera Position Randomly
11:     Zip Data $D_{j,k}$ over $k \rightarrow D_j$
12: Zip Data $D_j$ over $j \rightarrow D$
13: return $D$
```

Backface culling and HardFlatShader were used, respectively, for rasterizing and shading the 3D models. We then split the dataset randomly using an 80–20% ratio. We did so while keeping all 20 images resulting from the individual scenes in the same split—that is either training or validation. The resulting task was, therefore, more complex than if the dataset split jad been carried out along the individual images.

The resulting dataset provided images with occluded objects as well as occlusion maps for three objects. Moreover, it provided the environment gradient for the optimal step at each environment state, i.e., relative camera and object positions. In contrast with previously published occlusion datasets [18,19], this array of data at each given datapoint allowed the training of a PAL agent.

We evaluated the model following every epoch during the training time, using the validation split, and assured that the best performing models were saved. We augmented the dataset using Color Jitter and randomized horizontal flipping, and trained the model using an AdamW optimizer with Cosine Annealing Learning Rate Schedule. For the two heads of the network we used different loss functions. The occlusion mask prediction was evaluated using the Soft Dice Coefficient (SDC), whereas the gradient direction was optimized with MSE loss between the ground truth and predicted values. In practice, the sine and cosine values of the gradient direction were used for calculating the error. To summarize, the loss function used for pre-training is the following:

$$L_{SD} = 1 - \frac{2\ f_\theta(Img) \odot M_O}{f_\theta(Img)^2 + M_O^2}, \tag{10}$$

$$L_{MSE} = \|g_\Psi(Img) - [sin(\alpha),\ \ cos(\alpha)]\|^2, \tag{11}$$

$$L = \beta_1 L_{SD} + \beta_2 L_{MSE}, \tag{12}$$

where f and g are the segmenter and gradient predictor networks parameterized by θ and Ψ, respectively. Img is the input image, while α and M_O are the ground truth gradient direction and occlusion mask. $\odot$ is an element-wise multiplication, and β_1 and β_2 are the relative weights of the two loss terms. Table 1 shows the hyperparameters we used during training.

Table 1. The hyperparameters used for pre-training.

Parameter	**Epochs**	**LR**	**Decay**	**Batch**	β_1	β_2
Value	50	10^{-3}	10^{-5}	128	1	1

5.3. Training

The reinforcement learning agent was based on this pre-trained model, as it used the CNN as its backbone. We trained action and value heads of the RL agent using Proximal Policy Optimization [38]. Two variations were trained: one with union-based and one with intersection-based normalization. Table 2 shows the hyperparameters we used for training the reinforcement-learning agents.

Table 2. The hyperparameters used for training the RL-based agent. N denotes the number of environment steps, PPO made one update for K epochs every N_u steps. ϵ denotes the parameter of the PPO loss function, σ_a is the variance of the continuous action, while γ is the discount rate.

Parameter	N	N_u	LR_{Act}	LR_{Crit}	σ_a	ϵ	K	γ
Value	20,000	200	3×10^{-4}	10^{-3}	0.6	0.2	80	0.99

The training was performed using the following loss function:

$$L^{act}_\Theta = \mathbb{E}_t[min(r_t(\Theta)A_t,\ \ clip(r_t(\Theta), 1-\epsilon, 1+\epsilon)A_t], \tag{13}$$

$$r_t(\Theta) = \frac{\pi_\Theta(a_t|s_t)}{\pi_{\Theta_{Old}}(a_t|s_t)}, \tag{14}$$

$$L = LR_{act} L_\Theta^{act} + LR_{Crit}(V_\Theta(S_t) - V_{target})^2 \tag{15}$$

where A_t is the advantage function, ϵ is the PPO hyperparameter, π_Θ and V_Θ are the actor and critic networks, respectively, while V_{target} is the target value function derived from the reward R (defined in Equation (7)).

Figure 3 shows the number of steps taken to reach the unoccluded state and the rewards received by the agents. Note that the training was successful, since both RL agents were able to learn the solution of the given problem and avoid occlusion in a relatively low amount of steps.

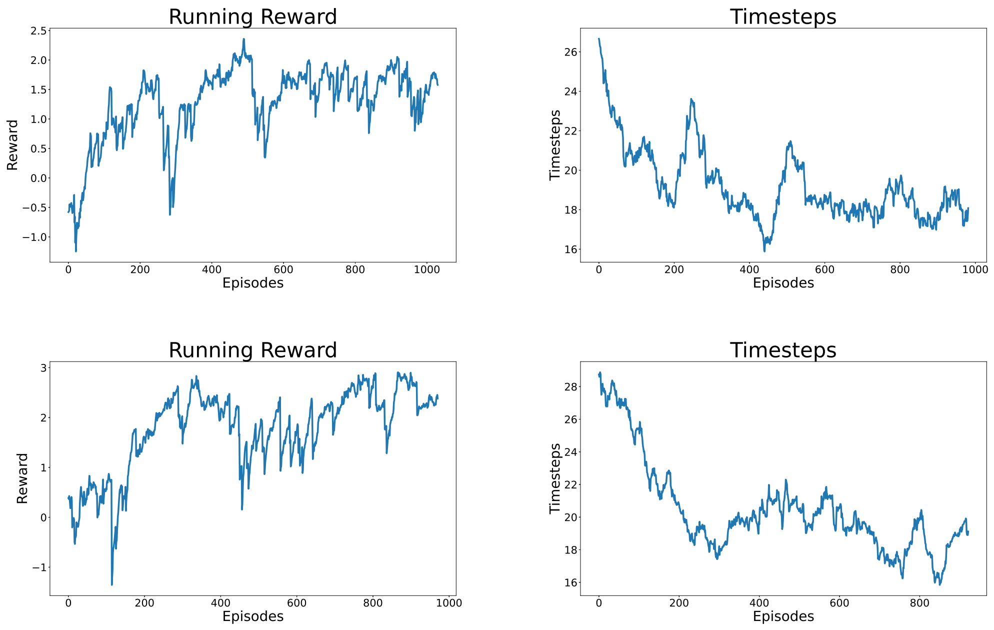


Figure 3. Results of training the union-normalized (**top**) and the intersection-normalized (**bottom**) agents. The figure shows the rolling average rewards (**left**) and the total number of timesteps taken (**right**) during occlusion-avoidance.

6. Experimental Results

In this section, the results of our experiments are presented. First, we discuss the results of pre-training, and, then, we compare the results of the fully neural approaches with the direct, environment-supplied gradient-based occlusion avoidance. Our experiments were performed using an Intel i5 8400U CPU, an NVIDIA Titan XP GPU, and 16 GB DDR4 RAM, running Ubuntu 20.24.

6.1. Occlusion and Gradient Estimation

The results of the training procedures introduced in Section 5 are shown in Table 3. The best loss values of the training are presented, i.e., the dice-loss (denoted with Loss) and the MSE loss of the multi-task occlusion and gradient estimator heads. In addition, the pixel-wise accuracy and the Intersection over Union (IoU) metrics are also given for the occlusion predictions. The results of the four previously introduced network variants are compared in the table. Moreover, during one of our experiments, the smooth L1 loss was used (with $\beta = 0.01$) to train the gradient direction estimator head.

Based on the results shown in Table 3, we concluded that the SAR-UNet-L1 architecture slightly outperformed the other 4 network variants on the metrics related to the occlusion mask prediction task, and strongly surpassed them on the gradient direction estimation task. We strongly suspected that this increased accuracy was caused by slowly vanishing the smooth L1 loss around zero that allowed the persistence of useful gradients around small loss values. We found that separable convolutions caused the generalization capabilities of the network to improve, while only slightly deterring the performance of the network. As a result we decided to keep these modifications as implemented for the later experiments. Although a 60% IoU measure is generally not considered great, we emphasize that, in this task, the agent often had to predict the occlusion mask for fully occluded objects—a completely impossible task. For this reason, we accepted the training results, since the the neural network often had little to no clue about the environment and, therefore, was met with a task incorporating increased complexity as a result of fully occluded objects.

Table 3. Results of the pre-training on the architectures investigated. We display the dice loss, pixel-wise accuracy and Intersection over Union (IoU) measures for the occlusion segmentation, and the Mean Squared Error (MSE) for the gradient prediction task. Best results are emphasized in bold.

Model	MSE	Loss	Pixel Acc	IoU
Base UNet	0.038	0.607	99.65	61.09%
R-UNet	0.038	0.606	99.65	61.47%
SR-UNet	0.038	0.61	99.64	60.34%
SAR-UNet	0.038	0.609	99.66	61.87%
SAR-UNet-L1	**0.005**	**0.604**	**99.67**	**62.37%**

6.2. *Occlusion Avoidance*

With the neural network-based agent, we performed numerous experiments using the environment described in Section 3. For the reproducibility of the experiments, we designed two simple test scenarios in which the object categories were fixed. First, we tested capabilities of the agents using 2 identical chairs, and, then, using 2 identical tables in the environment. Figure 4 shows the initial environments for these test cases.

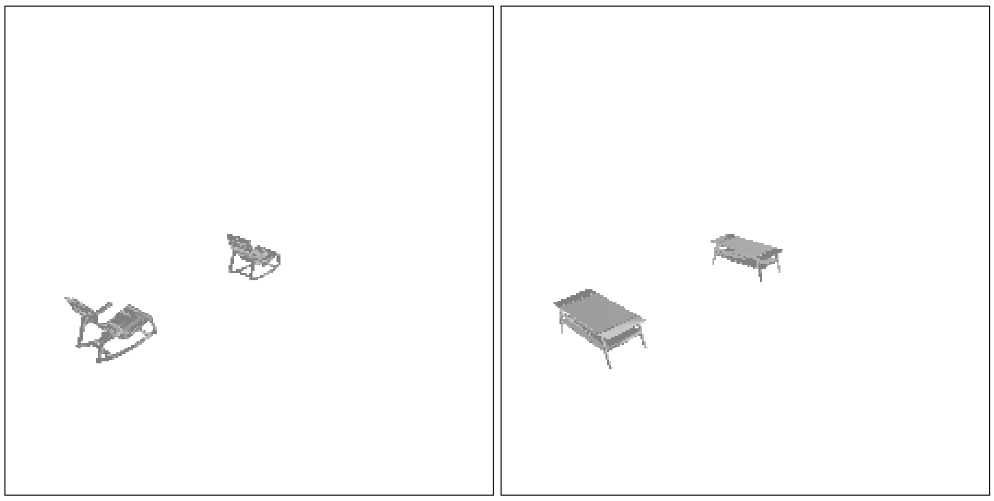


Figure 4. Environments for test scenarios. Two chairs (**left**) and two tables (**right**).

6.2.1. Optimization Using Environment Gradients

The test cases were repeated numerous times, where we varied the initial azimuth (φ) value between $[-\pi/32; \pi/32)$ with a step size of $d\varphi = \pi/208$. The results shown in Figure 5 visualize the main advantage of the pre-trained method. With objects characterized by complex geometries, such as the chairs and the tables used, the environment-supplied gradient-based method took a considerably higher amount of steps to reach an unoccluded state, especially if the initial azimuth value was set to be low, i.e., between $-2°$ and $2°$. We argue that this was caused by the environment gradients moving the camera towards local minima, which were more common in cases of objects containing structural holes and beams. This phenomenon can be seen from the variance of the steps taken in Figure 5. During the two chairs test case, when the initial azimuth value was set to 0, the direct gradient-based agent took 2377 steps to find an unoccluded state, whereas completing the task for two tables only took 163 steps. Note that, in the figure, the maximum number of steps was limited to 2000 to enhance visibility.

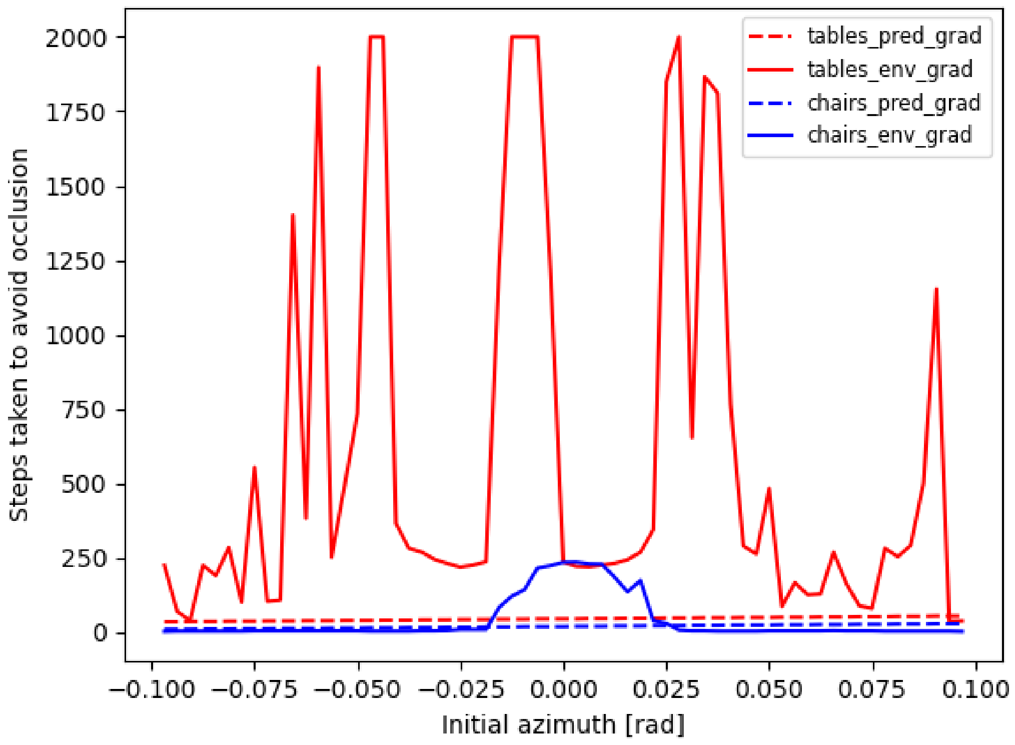


Figure 5. Two chairs and two tables test cases: the number of steps taken by the agents to reach an unoccluded state from various initial azimuth (φ) values.

We found that the methods usually yielded somewhat symmetrical results for negative and positive initial φ values. This symmetry can also be observed in Figure 5. We argue that this was caused by the symmetrical structure of the used table and chair objects. The slight differences were most likely caused by the slight imperfections of the surfaces of the models, that caused the silhouette renders used for gradient calculation to become unsymmetrical. These imperfections, however, can also be expected from real-life, physical objects. Therefore, we think that this feature of the ShapeNet dataset had a positive effect on the generalization capabilities of our solution. The previously mentioned problem of the render engine causing the sporadic appearance of NaN gradients could also lead to an increased number of steps taken by the agents.

6.2.2. Comparative Analysis

To compare the newly developed reinforcement learning agent with our prior solutions, we performed experiments using five different agents:

1. a random agent moving in random directions;
2. an environment agent that stepped in the environment gradient direction;
3. a pre-trained agent that moved towards the predicted gradient direction—as introduced in Section 5.2;
4. two reinforcement learning agents—as introduced in Section 5.3.

The agents were only able to perform normalized length ($l = 1$) steps. For these experiments, we set up the environment to contain three random objects, but the initial azimuth values were now always set to $\varphi = 0$. We ran each setting of the environment with the five aforementioned agents for 50 randomly initialized environments.

The outcomes of the experiments are displayed in Table 4. From the shown results, it can be observed that the reinforcement learning agents were, in fact, capable of arriving at an unoccluded state in a reliably lower amount of steps than all the other agents used. The superiority of the robustness of this latter method could be observed in both the mean value and the variance of the number of steps taken by the agents. The intersection normalized RL agent outperformed all the other methods in our test scenario; however, the reasons contributing to this result are subject to further tests and examination.

Table 4. Effectiveness of occlusion avoidance: The table shows the average number of steps the strategies took to reach the unoccluded state, and the variance. Best results are emphasized in bold.

Method	Random	Env. Grad	Pred. Grad	Agent-UN	Agent-IN
Mean steps	31.5	20.8	18.5	15.9	**15.8**
Std	13.9	10.2	8.2	4.1	**3.8**

We also performed a paired sample Bayesian *t*-test [39] using the BEST library to provide statistical proof of our method's efficacy. The test used 50 paired samples performed with the aforementioned four methods. For simplicity's sake, we only used the Intersection-normalized version of the RL agent, as the two versions were very close in performance. The summarized results of the tests are presented in Table 5, while the full results are shown in Figure A1 in the Appendix A.

Table 5. Bayesian *t*-test comparing the four methods of occlusion avoidance. We display the bounds of the 95% HDI between the two methods, as well as the difference of means and the confidence that the second of the compared methods reached the unoccluded state *significantly* faster that the first.

Baseline	Random	Random	Random	Env. Grad	Env. Grad	Pred. Grad
Method	Env. Grad	Pred. Grad	Agent-IN	Pred. Grad	Agent-IN	Agent-IN
HDI low	6.42	9.15	11.69	−0.21	2.11	0.05
HDI high	14.97	17	19.36	2.43	6.74	1.63
Diff of Means	10.73	12.98	15.58	1.03	4.34	0.84
Confidence	100	100	100	94.6	100	98.47

The results demonstrated that all three proposed methods credibly surpassed the random policy with significant effect sizes. In practice, this meant that our methods reached the unoccluded state in significantly fewer steps. It was also apparent that the two neural network-based methods achieved higher results than the method using the environment gradients. However, the predicted-gradient method only surpassed the former with a confidence of 94.6%, barely falling short of the commonly accepted 95% threshold. The RL agent-based solution, however, credibly outperformed all other solutions, albeit the average difference between the two best performing methods was less than a single step. Still, this difference resulted in a confidence of 98.47%, which, we argue, was possible due to the significantly lower variance of the RL agent's performance.

7. Discussion

In this paper, a novel method was presented that successfully implemented occlusion avoidance as a perception–action–loop. The method was trained using a combination of self-supervised and reinforcement learning techniques, and was able to efficiently optimize the camera position in an OpenAI Gym-compatible virtual environment. The environment was created using PyTorch3D's differentiable renderer and provided three random occluded objects from the ShapeNet dataset. In total, the following four agents were presented: the first used the gradients calculated analytically using differentiable rendering; the second employed a random policy; the third used self-supervised learning to predict the optimal direction of movement; the fourth employed reinforcement learning to avoid occlusion in a minimal number of steps. The first and second agents were used as baselines for comparison. The third and fourth methods used the RGB-D images as input.

Our experiments demonstrated that all agents performed well in the environment; however, the agents that employed learning managed to significantly outperform all others. The agent using reinforcement learning proved to be the most reliable, as demonstrated by its credibly superior performance according to the Bayesian *t*-test performed during our investigations. The RL-based agent also had the smallest variance in the number of steps taken to avoid occlusion, further cementing it as the highest performing solution.

Naturally, the methods come with some limitations, namely that they have been trained and tested in synthetic environments, where both the color and depth images used as inputs were devoid of any errors. This likely meant that, in order to use this agent in a real environment, sim-to-real transfer would have to be done. Furthermore, the background used in the environment would also need to be updated to include a more realistic background, with walls, floor and ceiling, as the existence of these in a real setting may confuse an agent trained on fully synthetic data. Our datasets and the code used have been made available at https://github.com/MILAB-IIT-CV/OcclusionEnv (accessed on 27 January 2023).

Author Contributions: Conceptualization, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); methodology, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); software, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); validation, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); formal analysis, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); investigation, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); resources, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); data curation, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); writing—original draft preparation, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); writing—review and editing, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); visualization, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); supervision, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); project administration, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó); funding acquisition, M.S. (Márton Szemenyei) and M.S. (Mátyás Szántó). All authors have read and agreed to the published version of the manuscript.

Funding: Support was provided by the the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory, and Márton Szemenyei was supported by the UNKP-22-4 New National Excellence Program of the Ministry for Innovation and Technology.

Institutional Review Board Statement: Not applicable.

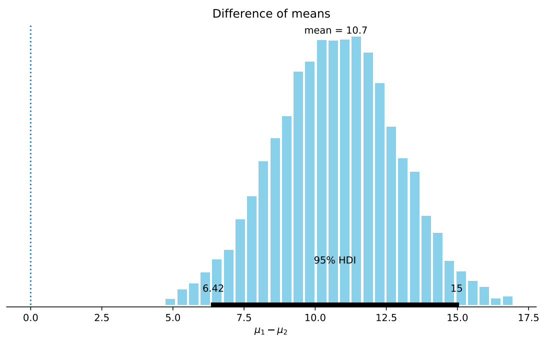
Informed Consent Statement: Not applicable.

Data Availability Statement: All software elements and the brief explanation of the usage thereof, as well as the dataset, are made available by the authors on https://github.com/MILAB-IIT-CV/OcclusionEnv (accessed on 27 January 2023).

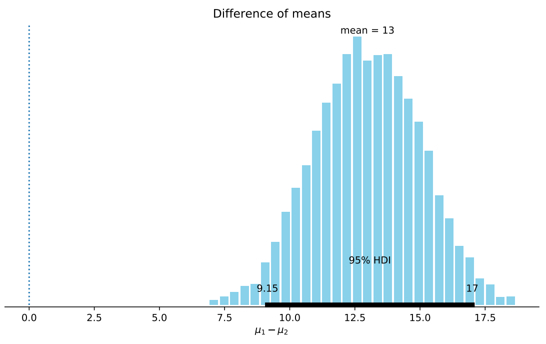
Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

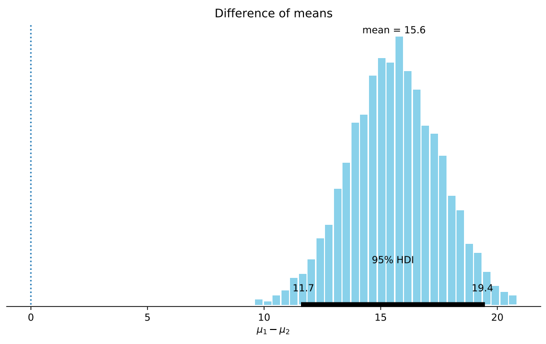
Here, the detailed results of the Bayesian *t*-test are presented.



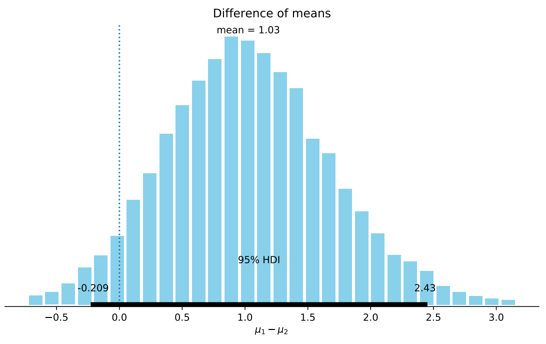
(**a**) Random Policy vs. Environment Gradients.



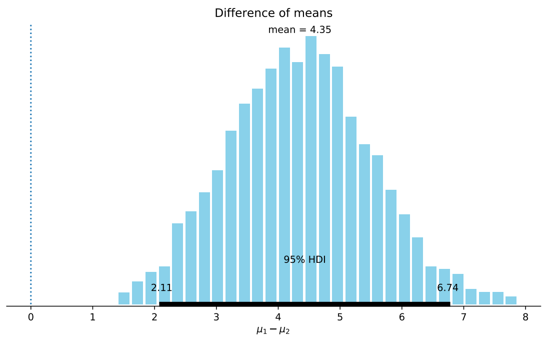
(**b**) Random Policy vs. Predicted Gradients.



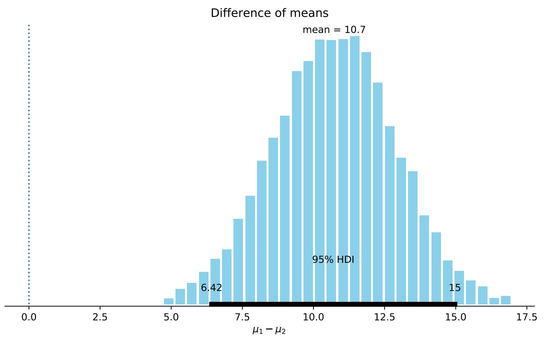
(**c**) Random Policy vs. the RL agent.



(**d**) Environment vs. Predicted Gradients.



(**e**) Environment Gradients vs. the RL agent.



(**f**) Predicted Gradients vs. the RL agent.

Figure A1. Results of the Bayesian *t*-test.

References

1. Chen, P.; Liu, W.; Dai, P.; Liu, J.; Ye, Q.; Xu, M.; Chen, Q.; Ji, R. Occlude Them All: Occlusion-Aware Attention Network for Occluded Person Re-ID. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 11813–11822. [CrossRef]
2. Liu, D.; Arai, S.; Xu, Y.; Tokuda, F.; Kosuge, K. 6D pose estimation of occlusion-free objects for robotic Bin-Picking using PPF-MEAM with 2D images (occlusion-free PPF-MEAM). *IEEE Access* **2021**, *9*, 50857–50871. [CrossRef]
3. Follmann, P.; König, R.; Härtinger, P.; Klostermann, M. Learning to See the Invisible: End-to-End Trainable Amodal Instance Segmentation. *arXiv* **2018**, arXiv:1804.08864. [CrossRef]
4. Kortylewski, A.; He, J.; Liu, Q.; Yuille, A. Compositional Convolutional Neural Networks: A Deep Architecture with Innate Robustness to Partial Occlusion. *arXiv* **2020**, arXiv:2003.04490. [CrossRef]
5. Kortylewski, A.; Liu, Q.; Wang, A.; Sun, Y.; Yuille, A. Compositional Convolutional Neural Networks: A Robust and Interpretable Model for Object Recognition Under Occlusion. *Int. J. Comput. Vis.* **2020**, *129*, 736–760. [CrossRef]

6. Szanto, M.; Szemenyei, M. Self-Supervised Occlusion Detection and Avoidance using Differentiable Rendering. In Proceedings of the 2022 International Symposium on Measurement and Control in Robotics (ISMCR), Houston, TX, USA, 28–30 September 2022. [CrossRef]
7. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv* **2013**, arXiv:1311.2524. [CrossRef]
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2015**, arXiv:1506.01497. [CrossRef]
9. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017. [CrossRef]
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2015**, arXiv:1506.02640. [CrossRef]
11. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
12. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016*; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
13. Li, C.; Li, L.; Geng, Y.; Jiang, H.; Cheng, M.; Zhang, B.; Ke, Z.; Xu, X.; Chu, X. YOLOv6 v3.0: A Full-Scale Reloading. *arXiv* **2023**, arXiv:2301.05586. [CrossRef]
14. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696. [CrossRef]
15. Xu, S.; Wang, X.; Lv, W.; Chang, Q.; Cui, C.; Deng, K.; Wang, G.; Dang, Q.; Wei, S.; Du, Y.; et al. PP-YOLOE: An evolved version of YOLO. *arXiv* **2022**, arXiv:2203.16250. [CrossRef]
16. Zong, Z.; Song, G.; Liu, Y. DETRs with Collaborative Hybrid Assignments Training. *arXiv* **2022**, arXiv:2211.12860. [CrossRef]
17. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. *arXiv* **2019**, arXiv:1904.08189. [CrossRef]
18. Zhan, G.; Xie, W.; Zisserman, A. A Tri-Layer Plugin to Improve Occluded Detection. *arXiv* **2022**, arXiv:2210.10046. [CrossRef]
19. Qi, J.; Gao, Y.; Hu, Y.; Wang, X.; Liu, X.; Bai, X.; Belongie, S.; Yuille, A.; Torr, P.H.S.; Bai, S. Occluded Video Instance Segmentation: A Benchmark. *arXiv* **2021**, arXiv:2102.01558. [CrossRef]
20. Saleh, K.; Szénási, S.; Vámossy, Z. Occlusion Handling in Generic Object Detection: A Review. In Proceedings of the 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 21–23 January 2021. [CrossRef]
21. Zhan, X.; Pan, X.; Dai, B.; Liu, Z.; Lin, D.; Loy, C.C. Self-Supervised Scene De-occlusion. *arXiv* **2020**, arXiv:2004.02788. [CrossRef]
22. Pásztor, G. Robust Object Detection in Simulated Industrial Environments. Master's Thesis, Budapest University of Technology and Economics: Budapest, Hungary, 2021.
23. Wei, D.; Lim, J.; Zisserman, A.; Freeman, W.T. Learning and Using the Arrow of Time. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018. [CrossRef]
24. Baker, B.; Kanitscheider, I.; Markov, T.M.; Wu, Y.; Powell, G.; McGrew, B.; Mordatch, I. Emergent Tool Use From Multi-Agent Autocurricula. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
25. Pathak, D.; Gandhi, D.; Gupta, A. Self-Supervised Exploration via Disagreement. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; MIT Press: Cambridge, MA, USA, 2019, Volume 97, pp. 5062–5071.
26. Nair, A.; Pong, V.; Dalal, M.; Bahl, S.; Lin, S.; Levine, S. Visual Reinforcement Learning with Imagined Goals. *arXiv* **2018**, arXiv:1807.04742. [CrossRef]
27. Pathak, D.; Agrawal, P.; Efros, A.A.; Darrell, T. Curiosity-driven Exploration by Self-supervised Prediction. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; MIT Press: Cambridge, MA, USA, 2017, Volume 70, pp. 2778–2787.
28. Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.J.; Darrell, T.; Efros, A.A. Large-Scale Study of Curiosity-Driven Learning. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
29. Reizinger, P.; Szemenyei, M. Attention-Based Curiosity-Driven Exploration in Deep Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, 4–8 May 2020; pp. 3542–3546. [CrossRef]
30. Ha, D.; Schmidhuber, J. Recurrent World Models Facilitate Policy Evolution. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montreal, QC, Canada, 3–8 December 2018; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 2455–2467.
31. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3D Object Detection Network for Autonomous Driving. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
32. Szemenyei, M.; Reizinger, P. Attention-Based Curiosity in Multi-Agent Reinforcement Learning Environments. In Proceedings of the 2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO), Athens, Greece, 8–10 December 2019. [CrossRef]

33. Szemenyei, M.; Estivill-Castro, V. Fully neural object detection solutions for robot soccer. *Neural Comput. Appl.* **2021**, *34*, 21419–21432. [CrossRef]
34. Ravi, N.; Reizenstein, J.; Novotny, D.; Gordon, T.; Lo, W.Y.; Johnson, J.; Gkioxari, G. Accelerating 3D Deep Learning with PyTorch3D. *arXiv* **2020**, arXiv:2007.08501. [CrossRef]
35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
36. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. *ShapeNet: An Information-Rich 3D Model Repository*; Technical Report; Stanford University: Stanford, CA, USA; Princeton University: Princeton, NJ, USA; Toyota Technological Institute at Chicago: Chicago, IL, USA, 2015. [CrossRef]
37. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:1505.04597. [CrossRef]
38. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
39. Kruschke, J.K. Bayesian estimation supersedes the t test. *J. Exp. Psychol. Gen.* **2013**, *142*, 573–603. [CrossRef] [PubMed]