2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

PI-ARS: Accelerating Evolution-Learned Visual-Locomotion with Predictive Information Representations

Kuang-Huei Lee*[1] Ofir Nachum*[1] Tingnan Zhang[1] Sergio Guadarrama[1] Jie Tan[1] Wenhao Yu[1]

***Abstract*— Evolution Strategy (ES) algorithms have shown promising results in training complex robotic control policies due to their massive parallelism capability, simple implementation, effective parameter-space exploration, and fast training time. However, a key limitation of ES is its scalability to large capacity models, including modern neural network architectures. In this work, we develop Predictive Information Augmented Random Search (PI-ARS) to mitigate this limitation by leveraging recent advancements in representation learning to reduce the parameter search space for ES. Namely, PI-ARS combines a gradient-based representation learning technique, Predictive Information (PI), with a gradient-free ES algorithm, Augmented Random Search (ARS), to train policies that can process complex robot sensory inputs and handle highly nonlinear robot dynamics. We evaluate PI-ARS on a set of challenging visual-locomotion tasks where a quadruped robot needs to walk on uneven stepping stones, quincuncial piles, and moving platforms, as well as to complete an indoor navigation task. Across all tasks, PI-ARS demonstrates significantly better learning efficiency and performance compared to the ARS baseline. We further validate our algorithm by demonstrating that the learned policies can successfully transfer to a real quadruped robot, for example, achieving a 100% success rate on the real-world stepping stone environment, dramatically improving prior results achieving 40% success.**

I. Introduction

Evolution Strategy (ES) optimization techniques have received increasing interest in recent years within the robotics and deep reinforcement learning (DRL) communities [1], [2], [3], [4], [5], [6], [7]. ES algorithms have been shown to be competitive alternatives to commonly used gradient-based DRL algorithms such as PPO [8] and SAC [9], while also enjoying the benefits of massive parallelism, simple implementation, effective parameter-space exploration, and faster training time [1].

Despite the promising progress of ES algorithms, they nevertheless exhibit key limitations when compared to gradient-based DRL algorithms. Namely, unlike gradient-based methods, ES methods scale poorly to high-dimensional search spaces, commonly encountered when using high-capacity modern neural network architectures [10], [1]. An important and exemplary task is visual-locomotion [3], in which a legged robot relies on its vision input to decide where to precisely place its feet to navigate uneven terrains. Due to the rich and diverse sensor observations as well as complex robot dynamics, learning such a task requires the use of deep convolutional neural networks (CNNs) with a large number of learnable parameters, thus exacerbating the sample-complexity of ES methods.

In this paper, we develop Predictive Information Augmented Random Search (PI-ARS) to relieve this key bottleneck of ES algorithms. Our key insight is to leverage the power of gradient-based and gradient-free learning together by modularizing the learning agent into two components: (1) an encoder network mapping high-dimensional and diverse observation inputs to a concise fixed-length vector *representation*, and (2) a smaller policy network that maps the compressed representations to actions. For (1), we leverage the power of gradient-based learning, and use a self-supervised objective based on maximizing the *predictive information* (PI) of the output representation, inspired by previous work in representation learning [11], [12], [13], [14], [15], [16]. Meanwhile for (2), we leverage the simplicity and parallelizability of the ES optimization method Augmented Random Search (ARS) [2]. By decoupling representation learning from policy optimization in this way, we avoid scalability issues while fully leveraging the advantages of ES methods.

We evaluate our proposed PI-ARS algorithm on a variety of visual-locomotion tasks, both in simulation and on a quadruped robot. Among these tasks, the robot is evaluated on its ability to walk on uneven stepping stones, quincuncial piles, and moving platforms, as well as to complete an indoor navigation task (Figure 2). Through extensive experimentation in simulation, we find that PI-ARS significantly outperforms the baselines (ARS [3], SAC [9], PI-SAC [13]), both in training speed and final performance. We further validate the results by deploying the learned policies on a real quadruped robot. Using the same physical setup as prior work [3], PI-ARS learns more robust policies that can consistently finish the entire course of stepping stones, achieving 100% success over 10 real-robot trials, compared to 40% success rate of prior work. We observe similarly successful robustness to real-world transfer for the indoor navigation policy.

In summary, the contributions of this paper are the following:

1) We propose a new PI-ARS algorithm that combines the advantages of gradient-based self-supervised representation learning and gradient-free policy learning, thus solving a key bottleneck of ES algorithms.
2) We apply PI-ARS in visual-locomotion tasks, which significantly improve the state-of-the-art [3] both in simulation and in the real world.

*Equal contribution
[1]All authors are with Google Research,
1600 Amphitheatre Parkway Mountain View, CA 94043, United States
{leekh, ofirnachum, tingnan, sguada, jietan, magicmelon}@google.com

The supplementary video is available at kuanghuei.github.io/piars

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981952

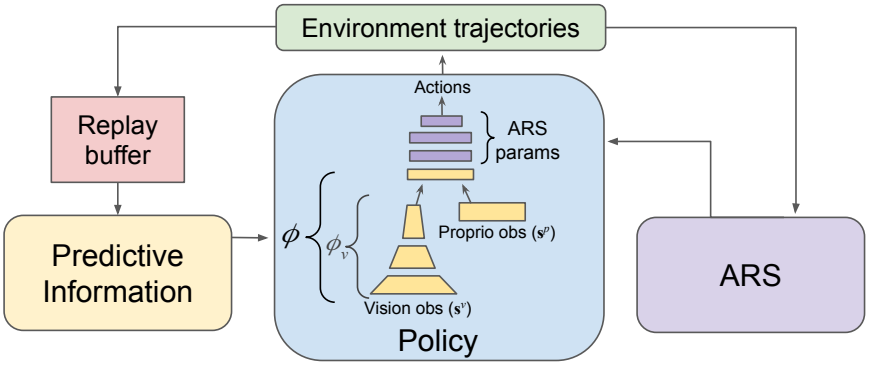


Fig. 1. In PI-ARS, an observation encoder ϕ is used to provide a compressed observation representation. This way, ARS learning is focused on a much smaller neural network than if the whole policy (mapping robot observations to actions) were learned end-to-end. As ARS uses sampled trajectories from the policy to update its parameters, PI-ARS uses the same trajectories to maintain a replay buffer for training ϕ via predictive information (PI), which includes auxiliary-learned networks as described in Section III-B.

II. Related Work

A. Evolution Strategy for RL

There have been numerous works that demonstrate the effectiveness of applying Evolution Strategy (ES) algorithms to continuous control problems [17], [1], [2]. For example, Tan et al. applied Neural Evolution of Augmenting Topologies (NEAT) to optimize a character to perform bicycle stunts [17]. Within the field of deep reinforcement learning, Salimans et al. first demonstrated that ES algorithm can be applied to train successful neural-network policies for the OpenAI Gym control tasks [1]. Mania et al. introduced Augmented Random Search (ARS), a simple yet effective ES algorithm that further improves the learning efficiency for robotic control tasks [2].

Compared to gradient-based RL algorithms, ES algorithms can handle non-differentiable dynamics and objective functions, explore effectively with sparse or delayed rewards, and are naturally parallelizable. As such, researchers have applied ES algorithms in a variety of applications such as legged locomotion [3], [18], power grid control [7], and mixed autonomy traffic [19]. However, as ES algorithms do not leverage backpropagation, they suffer from low sample efficiency and may not scale well to complex high-dimensional problems [1]. As a result, applying ES to learn vision-based robotic control policies is rarely explored [3].

B. Predictive Representations for RL

Our work relies on learning representations that are predictive of future events. Prior work has shown benefit from having good models of the past and future states [20], [21], [22]. More recently, using these principles to guide state representation learning methods has been demonstrated to yield favorable performance both in practice [13], [15] and in theory [23], [24]. A natural approach to learning such representations is using generative models to explicitly predict observations [12], [25], which could be challenging and expensive for high-dimensional tasks. Alternatively, using variational forms [26] of the predictive information [27], commonly leading to contrastive objectives, makes learning such representations more tractable [13], [15], [11]. In this work, we take the contrastive approach to learn predictive representations of the observed states, upon which we learn an optimal policy with augmented random search (ARS) [2], an ES method. To the best of our knowledge, the proposed learning system is the first to take such a combined approach, and the first to apply predictive information representations on visual locomotion tasks with legged robots.

C. Visual-Locomotion for Legged Robots

Visual-locomotion is an important research direction that has received much attention in the robotics research community [28], [29], [30], [31], [32], [33], [34], [35], [36]. Directly training a visual-locomotion controller is challenging due to the high dimensional visual input and the highly nonlinear dynamics of legged robots [3], [37]. Many existing methods manually and explicitly decouple the problem into more manageable components including perception [31], [36], motion planning [32], [38], and whole body control [39], [40], [41]. In this work, we consider direct learning of visual-locomotion controllers for quadruped robots as the test-bed for our proposed learning algorithm and demonstrate that by combining gradient-free ES and gradient-based representation learning techniques, we can enable more effective and efficient learning of visual-locomotion controllers.

III. Method

In this section we describe our method, PI-ARS, which combines representation learning based on predictive information (PI), and Augmented Random Search (ARS) [2]. See Figure 1 for a diagram of the algorithm and Algorithm 1 for a pseudocode.

A. Problem Formulation and Notations

PI-ARS solves a sequential decision-making environment in which at each timestep t the agent is presented with an *observation* $\mathbf{s}_t$. In visual-locomotion, this observation typically includes a visual input $\mathbf{s}_t^v$ (e.g., depth camera images) as well as proprioceptive states $\mathbf{s}_t^p$. The agent's *policy* determines its behavior, by providing a mapping from

observations $\mathbf{s}_t$ to *actions* $\mathbf{a}_t$. After application of action $\mathbf{a}_t$, the agent receives a reward r_t and a new observation $\mathbf{s}_{t+1}$. This process is repeated until the agent is terminated, either due to a timeout or encountering a terminal condition (e.g., the robot falls). The agent's *return* is computed as the sum of rewards over an entire episode, and the agent's goal is to maximize the expected value of the return.

B. Predictive Information

A good observation encoder for policy learning must provide representations that are both *compressive* – so that ARS learning is focused on much fewer parameters than learning from raw observations would entail – and *task-relevant* – so that ARS has access to all features necessary for learning optimal behavior. To this end, we propose to learn an encoder ϕ to maximize predictive information (PI). In general, PI refers to the mutual information between past and future $I(\text{past}; \text{future})$ [27]. In our setting involving environment-produced sub-trajectories $\tau = (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{a}_{t+1}, r_{t+1}, \ldots, \mathbf{a}_{t+k-1}, r_{t+k-1}, \mathbf{s}_{t+k})$, past corresponds to $(\mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1})$ and future refers to the both the per-step rewards $(r_t, \ldots, r_{t+k-1})$ and the ultimate visual observation $\mathbf{s}^v_{t+k}$[1]; i.e., $\mathbf{s}_{t+j} = (\mathbf{s}^v_{t+j}, \mathbf{s}^p_{t+j})$.

We use ϕ to map both $\mathbf{s}_t$ and $\mathbf{s}^v_{t+k}$ to a lower dimensional representation. Namely, as shown in Figure 1, the observation encoder ϕ contains a vision encoder ϕ_v that maps visual observations $\mathbf{s}^v_t$ to a 128-d representation. This representation is subsequently concatenated with proprioceptive states, and the concatenation is projected to be the output of ϕ, which is also 128-d. We thus use the entire encoder ϕ to encode $\mathbf{s}_t$, while use only the vision encoder ϕ_v to encode the future $\mathbf{s}^v_{t+k}$. By learning ϕ to maximize the mutual information between these past and future, we ensure that ϕ encodes the necessary information in $\mathbf{s}_t$ to predict the future. Notably, previous work has shown that representations that are predictive of the future are also provably beneficial for solving the task [23].

To learn ϕ, we use a combination of two objectives, the first corresponding to reward prediction and the second to $\mathbf{s}^v_{t+k}$ prediction.[2] For the former, we simply predict rewards $\hat{r}_t, \ldots, \hat{r}_{t+k-1}$ using an RNN over inputs $(\phi(\mathbf{s}_t), \mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1})$. At time $t+j$, the RNN cell takes a latent state and an action and outputs a reward prediction $\hat{r}_{t+j}$ and the next latent state. The reward loss is defined as

$$L_r = \mathbb{E}_\tau \left[\sum_{j=0}^{k-1} (\hat{r}_{t+j} - r_{t+j})^2 \right]. \tag{1}$$

This corresponds to maximizing $I(\text{past}; \text{future rewards})$ with a generative model [42].

For the prediction of $\mathbf{s}^v_{t+k}$, rather than a generative model, as used for $\hat{r}_{t+j}$, which would present challenges for high-dimensional image observations, we leverage InfoNCE, a contrastive variational bound on mutual information [11], [26], [13]. We use an auxiliary learned, scalar-valued function f to estimate the mutual information using access to samples of sub-trajectories $\tau = (\mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1}, \mathbf{s}_{t+k})$. Specifically, we conveniently exclude the course of actions and choose the following form:

$$I(\text{past}; \text{future obs}) \geq \tilde{I}(\text{past}; \text{future obs}) = \mathbb{E}_\tau \Big[f(\phi(\mathbf{s}_t), \phi_v(\mathbf{s}^v_{t+k})) - \log \mathbb{E}_{\tilde{\mathbf{s}}_{t+k}} [\exp\{f(\phi(\mathbf{s}_t), \phi_v(\tilde{\mathbf{s}}^v_{t+k}))\}] \Big], \tag{2}$$

where $\tilde{\mathbf{s}}^v_{t+k}$ is from an observation randomly sampled independent of τ. Our objective is then to maximize this variational form with respect to both ϕ and f.

To parameterize f, we use an MLP to map $\phi(\mathbf{s}_t)$ to $\mathbf{z}^{\text{past}}_t$, a 128-d vector. Meanwhile, we map $\phi_v(\mathbf{s}^v_{t+k})$ to $\mathbf{z}^{\text{future}}_{t+k}$, a 128-d vector representation of the future, using another MLP. The function f is then computed as a scalar dot-product of the two vectors.

We train both the reward objective and the variational objective using batch samples from a replay buffer of sub-trajectories collected by ARS. To approximate $\mathbb{E}_{\tilde{\mathbf{s}}_{t+k}}$ in Equation (2), we use samples of $\tilde{\mathbf{s}}^v_{t+k}$ within the same batch of sub-trajectories. The full objective is optimized using the Adam stochastic gradient descent optimizer [43] and the gradient is calculated using back-propagation. Full implementation details are included in the Appendix.

Algorithm 1 Pseudocode for PI-ARS.

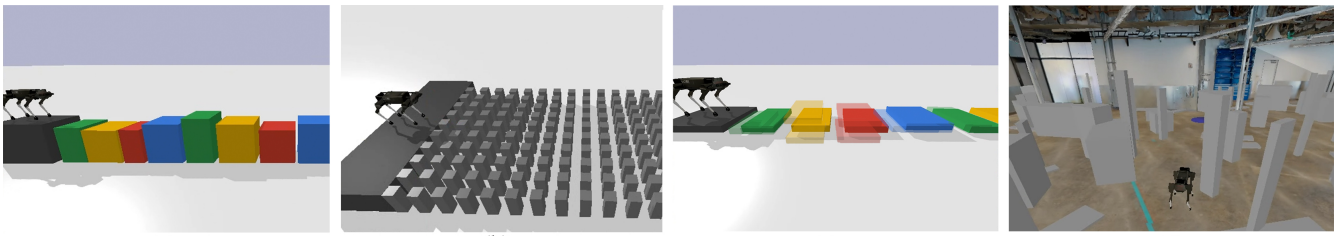
Initialize encoder ϕ, auxiliary networks (RNN, f), and optimizer $\text{Opt}_{\phi,\text{aux}}$.
Initialize ARS parameters θ and optimizer Opt_θ.
Initialize replay buffer $\mathcal{B}$.
for $T = 1, \ldots$ **do**
  #### ARS ####
  Sample $\{\sigma_i\}_{i=1}^N$ from normal with scale δ.
  Collect environment trajectories for policies $(\phi, \theta \pm \sigma_i)$.
  Compute returns $R_{i,\pm}$ for each policy.
  Compute gradient of M best-performing directions: $\hat{g} = \frac{\delta}{M} \sum_{i=1}^M (R_{i,+} - R_{i,-}) \sigma_i$.
  Update θ w.r.t. gradient $\hat{g}$ and Opt_θ.
  Add trajectories to $\mathcal{B}$.
  #### PI ####
  Sample batch of sub-trajectories $\{\tau_i\}_{i=1}^B$ from $\mathcal{B}$.
  for $i = 1, \ldots, B$ **do**
    $\hat{L}_i = -\tilde{I}(\tau_i) + L_r(\tau_i)$ Equations (1), (2)
  end for
  Compute total loss $\hat{L} = \sum_{i=1}^B \hat{L}_i$.
  Update ϕ and aux. networks w.r.t. loss $\hat{L}$ and $\text{Opt}_{\phi,\text{aux}}$.
end for

C. PI-ARS

The encoder ϕ learned by PI maps a high-dimensional observation to a concise 128-d representation, upon which we use ARS to train a more compact policy network as follows. At each iteration of ARS, the algorithm samples

[1]This empirical choice of future works well in our setting.

[2]We note that in our early experiments, we found learning ϕ using reward prediction alone leads to insufficient representations for solving the tasks.



(a) (b) (c) (d)

Fig. 2. The environments that we used to benchmark the PI-ARS learning system: (a) uneven stepping stones, (b) quincuncial piles, (c) moving platforms (the afterimage is for indicating that these platforms are moving), and (d) indoor navigation.

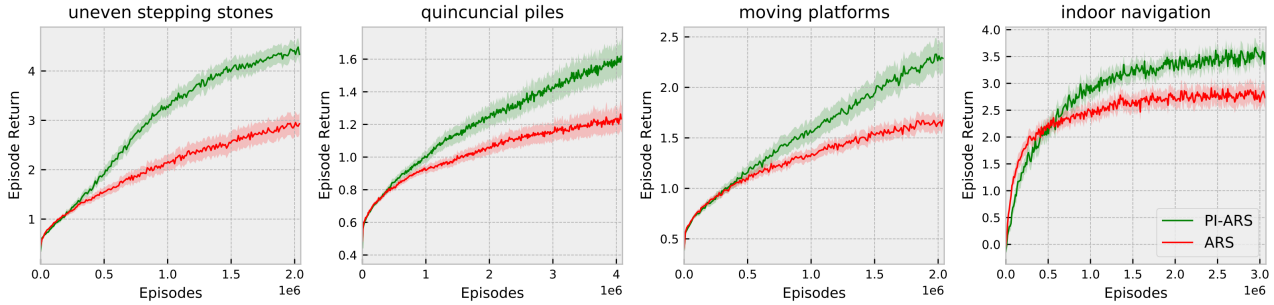


Fig. 3. Simulation results. We compare the performance of PI-ARS to ARS during training on four challenging simulation environments. PI-ARS consistently and significantly outperforms ARS.

N perturbations $[\sigma_1, \ldots, \sigma_N]$ of the policy weights θ using a standard normal distribution with scale σ. The algorithm evaluates the policy returns at $\theta + \sigma_i$ and $\theta - \sigma_i$. ARS then computes an estimation of the policy gradient by aggregating the returns from the best-performing perturbation directions:

$$\hat{g} = \frac{\delta}{M} \sum_{i=1}^{M} (R_{i,+} - R_{i,-})\sigma_i, \tag{3}$$

where δ is the update coefficient, M is the number of top-performing perturbations to be considered, and $R_{i,+}, R_{i,-}$ denote the total return of the policy at perturbations $\theta \pm \sigma_i$. We refer the readers to [2] for additional details.

We iterate between updating the representation network ϕ with the PI objective and updating the policy with ARS. To maximize data re-use, we store the sampled trajectories from perturbed policies evaluated by ARS in a replay buffer used for the PI learning pipeline.

IV. Experiments

We aim to answer the following questions in our experiments:

- Is our proposed algorithm, PI-ARS, able to learn vision-based policies that solve challenging visual-locomotion tasks?
- Does PI-ARS achieve better performance than alternative methods that do not apply representation learning?
- Are our learned policies applicable to real robots?

A. Visual-Locomotion Tasks

To answer the above questions, we design a variety of challenging visual locomotion tasks. Figure 2 shows the suite of environments that we evaluate on. More details of each environment can be found in Section B.

a) Uneven stepping stones: In this task, the robot is tasked to walk over a series of randomly placed stepping stones separated by gaps, and elevation of the stepping stones changes dramatically (Figure 2 (a)).

b) Quincuncial piles: This is an extension to uneven stepping stones, where we reduce the contact surface area and arrange stones in both forward and lateral directions (Figure 2(b)).

c) Moving platforms: We construct a set of stepping stones and allow each piece to periodically move either horizontally and vertically at a random speed (Figure 2(c)).

d) Indoor navigation with obstacles: In this task, we evaluate the performance of PI-ARS controlling the robot to navigate in a cluttered indoor environment (Figure 2 (d). Specifically, we randomly place boxes on the floor of a scanned indoor environment and command the robot to walk to a target position.

B. Experiment Setup

We use the Unitree Laikago quadruped robot [44], which weighs 24kg and has 12 actuated joints, with two depth cameras installed: one Intel D435 in the front for a wider field of view and one Intel L515 on the belly for better close-range depth quality. We create a corresponding simulated Laikago robot in the PyBullet physics simulator [45] with physical properties from hardware spec and simulated cameras that matches the camera intrinsics and extrinsics from the real cameras. The observation, action, and reward designs are detailed as follows.

1) Observation Space: We design the observation space in our visual-locomotion tasks following prior work by Yu et al. [3]. In particular, our observation space consists of two parts: $\mathbf{s} = (\mathbf{s}^v, \mathbf{s}^p)$, where $\mathbf{s}^v$ are the two 32×24 images from depth sensors, and $\mathbf{s}^p$ include all the proprioceptive states (and controller states). In our experiments, $\mathbf{s}^p = (\mathbf{q}_s, \dot{\mathbf{p}}, \dot{\Phi}, \dot{\Theta}, \mathbf{r}_{1...4}, c_{1...4}, \phi_{1...4}, \mathbf{a}_{prev})$ includes the CoM height, roll, and pitch $\mathbf{q}_s = (p_z, \Phi, \Theta)$, the estimated CoM velocity $\dot{\mathbf{p}}$, the gyroscope readings $\dot{\Phi}, \dot{\Theta}$, the robot's feet positions $\mathbf{r}_{1...4}$ in the base frame, the feet contact states $c_{1...4}$, the phase of each leg in its respective gait cycle ϕ, and the previous action.

For the indoor navigation task, we additionally include the relative goal vector $\mathbf{n} = \bar{\mathbf{o}} - \mathbf{p}$ as part of the observation, where $\bar{\mathbf{o}}$ is the target location and $\mathbf{p}$ is the robot's position.

2) Action Space: We follow the prior work [3] to use a hierarchical design for the visual-locomotion controller with a trainable high-level vision policy π_θ that maps visual and proprioceptive input to a high-level motion command, and an MPC-based low-level motion controller that executes the high-level motion command with trajectory optimization. The high-level motion command, i.e. the action space for the RL problem, is defined as: $(\mathbf{q}_s^d, \dot{\mathbf{q}}^d, \mathbf{r}_{1...4}^d, \mathbf{h}_{1...4})$, where $\mathbf{q}_s^d$ and $\dot{\mathbf{q}}^d$ are the desired CoM pose, velocity, $\mathbf{r}_i^d$ is ith foot's target landing position (r_{xi}, r_{yi}, r_{zi}), and $\mathbf{h}_i$ is the peak height of ith foot's swing trajectory.

3) Reward Function: For training a policy to walk on different terrains, we use the following reward function:

$$R(\mathbf{s}, \mathbf{a}) = clip(\dot{p}_x, -\dot{p}_x^{max}, \dot{p}_x^{max}) - w|\Psi|, \tag{4}$$

where $\dot{p}_x$ is the CoM velocity in the forward direction, and Ψ the base yaw angle. The first term rewards the robot to move forward with a maximum speed controlled by $\dot{p}_x^{max}$, the second term encourages the robot to walk straightly.

For the indoor navigation task, we use the delta geodesic (path) distance to the goal as our reward:

$$R(\mathbf{s}, \mathbf{a}, \bar{\mathbf{o}}) = d_g^t - d_g^{t-1}, \tag{5}$$

where d_g^t is the geodesic distance between the robot and the target location at time t.

4) Early termination.: A training episode is terminated if: 1) the robot loses balance (CoM height p_z below 0.15 m, pitch $|\Theta| > 1$ rad, or roll $|\Phi| > 0.3$ rad in our experiments), or 2) the robot reaches an invalid joint configuration, e.g. knee bending backwards.

C. Learning in Simulation

In this subsection, we discuss the results of PI-ARS learned on simulated visual-locomotion tasks and compare to a state-of-the-art ARS approach to robotic visual-locomotion [2], [3] (Figure 3). Among other baseline approaches that we have tried include SAC [9] and PI-SAC [13] but both algorithms failed to make any non-negligible learning progress for the tasks we consider despite extensive hyperparameter tuning, and so we omit these algorithms from the results. For fair comparison, all algorithms utilize the same MPC-based locomotion controller and learn policies in the high-level command space described in Section IV-B.2. All policies utilize the same network architecture; i.e., the policy learned by the baseline ARS method is composed of the same set of convolution and feed-forward layers used for ϕ in PI-ARS.

We train PI-ARS and ARS policies using a distributed implementation. For all PI-ARS and ARS experiments, we perform $N = 1024$ perturbations per ARS iteration and use the top 50% performers ($M = 512$) to update the policy network head. This choice, determined through a grid search, empirically works the best for both PI-ARS and ARS in our implementation. Further increase of N (and thus computation cost) does not significantly improve the performance. The algorithm is run until convergence with a maximum of 4000 training iterations, resulting in a maximum of $2,048,000$ simulation episodes per trial. We perform 30 trials of training PI-ARS/ARS with uniformly sampled σ and δ values ($\sigma \sim [0.005, 0.05]$, $\delta \sim [0.005, 0.05]$) and report the mean and standard error of returns against number of training episodes for each task.

As we demonstrate in the supplementary video, PI-ARS is able to learn vision-based policies that successfully solve these challenging visual-locomotion tasks. Figure 3 shows that on all tasks, PI-ARS gives significantly better returns and sample-efficiency than the ARS baseline. For example, on uneven stepping stones, the mean return after 2,000,000 episodes of training improves by 48.01%, from 2.93 to 4.34. This empirically demonstrates the effectiveness of learning ARS policies upon compressed, gradient-learned representations instead of end-to-end. On the other hand, observing that SAC fails to learn, we hypothesize that the advantages of ARS such as parameter-space exploration and stability are critical to these complex visual-locomotion tasks. Furthermore, adding predictive information to SAC, i.e. PI-SAC, does not improve learning, suggesting that even with an effective representation learner, without a powerful policy solver, a learning algorithm is not able to sufficiently tackle these visual-locomotion tasks.

D. Validation on Real Robot

We deploy the visual-locomotion policy trained in simulation on a Laikago robot to perform two visual-locomotion tasks: 1) walking over real-world stepping stones (Figure 4), and 2) navigating in an indoor environment with obstacles (Figure 5).

To overcome the sim-to-real gap, we adopt the same procedure as done by Yu et al. [3]. For the visual gap, during training, we first apply random noise to the simulated depth images to mimic the real-world depth noises. Then we apply a Navier-Stokes-based in-painting operation [46] with radius of 1 to fill the missing pixels, followed by down-sampling to $32x24$ (with OpenCV's INTER_AREA interpolation method for resizing [47]). On the real hardware, we obtain $640x480$ raw depth images from both L-515 and D435 cameras and perform the same in-painting and down-sampling. To mitigate the dynamics gap, we apply dynamics randomization during training.

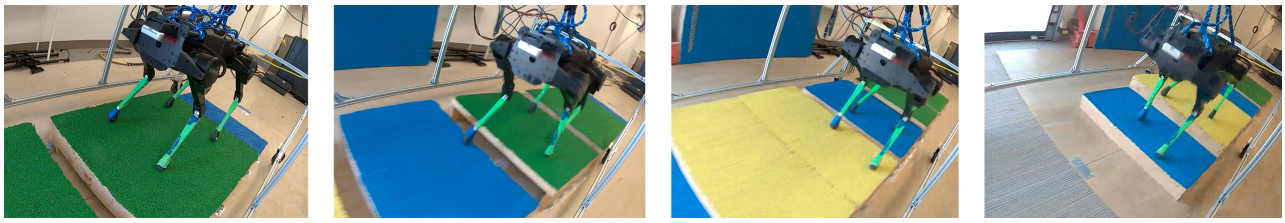


Fig. 4. PI-ARS policy solving a challenging real-world visual-locomotion task involving a series of four step stones separated by gaps. PI-ARS successfully completes this terrain, avoiding all gaps, 100% of the time measured over 10 trials.

Fig. 5. PI-ARS policy learns to navigate in a cluttered real-world indoor environment.

Videos of our real-world experiments can be found in the supplementary material.

a) Stepping Stones: For the stepping stones task, we created a physical setup consisting of four stones separated by three gaps between $[0.12, 0.18]$m (Figure 4). The PI-ARS policy is learned in simulation with an easier version of uneven stepping stones where stone heights change less significantly. Our PI-ARS policy was able to achieve 100% success rate on the stepping stone environment with 10 trials. In contrast, the ARS baseline [3] with the same training and evaluation setting achieved 40% success rate for reaching the last stone with all four legs and often failed at the last gap.

b) Indoor Navigation: For evaluating the navigation task in the real world, we design a route in an indoor environment with obstacles (Figure 5). The robot needs to navigate to the target location while avoiding the obstacles. To enable the robot to better avoid the obstacles, we rotate the front camera of the robot such that it can see $\sim$3 meters ahead of the robot. We also track the robot base position using a motion capture system, which is needed to compute the relative goal vector $\mathbf{n}$. As shown in the supplementary video, our PI-ARS policy is able to successfully navigate to the designated target location. For the setting shown in Figure 5, our policy successfully discovered a 'shortcut' in between two obstacles and was able to go through. We do note there is collision with the obstacle's arm. This is because the robot was trained with simulated obstacles with box shapes only; further training with more diverse obstacles would likely mitigate this problem.

Overall, these experiments validate that PI-ARS is capable of learning policies that can transfer to real robots.

V. Conclusion

We present a new learning method, PI-ARS, and apply it to the visual-locomotion problem. PI-ARS combines gradient-based representation learning with gradient-free policy optimization to leverage the advantages of both. PI-ARS enjoys the simplicity and scalability of gradient-free methods, and it relieves a key bottleneck of ES algorithms on high-dimensional problems by simultaneously learning a low-dimensional representation that reduces the search space. We evaluate our method on a set of challenging visual-locomotion tasks, including navigating through uneven stepping stones, quincuncial piles, moving platforms, and cluttered indoor environments, among which PI-ARS significantly outperforms the state-of-the-art. Furthermore, we validate the policy learned by PI-ARS on a real quadruped robot. It enables the robot to walk over randomly-placed stepping stones and navigating in an indoor space with obstacles. In the future, we plan to test PI-ARS in outdoor visual-locomotion tasks, which presents more diverse and interesting terrains for the robot to overcome.

Acknowledgments

We thank Noah Brown, Gus Kouretas, and Thinh Nguyen for helping set up the real-world stepping stones and address robot hardware issues.

References

[1] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[2] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.

[3] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *5th Annual Conference on Robot Learning*, 2021.

[4] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan, "Rapidly adaptable legged robots via evolutionary meta-learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3769–3776.

[5] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2950–2957, 2020.

[6] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[7] R. Huang, Y. Chen, T. Yin, X. Li, A. Li, J. Tan, W. Yu, Y. Liu, and Q. Huang, "Accelerated deep reinforcement learning based load shedding for emergency voltage control," *arXiv preprint arXiv:2006.12667*, 2020.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[9] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[10] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.
[11] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
[12] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.
[13] K.-H. Lee, I. Fischer, A. Liu, Y. Guo, H. Lee, J. Canny, and S. Guadarrama, "Predictive information accelerates learning in rl," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 890–11 901, 2020.
[14] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," *arXiv preprint arXiv:2004.04136*, 2020.
[15] M. Yang and O. Nachum, "Representation matters: Offline pretraining for sequential decision making," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 784–11 794.
[16] X. Chen, S. Toyer, C. Wild, S. Emmons, I. Fischer, K.-H. Lee, N. Alex, S. H. Wang, P. Luo, S. Russell, *et al.*, "An empirical investigation of representation learning for imitation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
[17] J. Tan, Y. Gu, C. K. Liu, and G. Turk, "Learning bicycle stunts," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 50:1–50:12, 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601121
[18] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7551–7557.
[19] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Conference on robot learning*. PMLR, 2018, pp. 399–409.
[20] J. Schmidhuber, "Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments," Institut für Informatik, Technische Universität München, Tech. Rep., 1990.
[21] ——, "Reinforcement learning in markovian and non-markovian environments," in *Advances in Neural Information Processing Systems*, 1991, pp. 500–506.
[22] ——, "On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models," *arXiv preprint arXiv:1511.09249*, 2015.
[23] O. Nachum and M. Yang, "Provable representation learning for imitation with contrastive fourier features," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
[24] M. Yang, S. Levine, and O. Nachum, "Trail: Near-optimal imitation learning with suboptimal data," *arXiv preprint arXiv:2110.14770*, 2021.
[25] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.
[26] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker, "On variational bounds of mutual information," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5171–5180.
[27] W. Bialek and N. Tishby, "Predictive information," *arXiv preprint cond-mat/9902341*, 1999.
[28] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1096–1103.
[29] O. A. V. Magana, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and continuous foothold adaptation for dynamic locomotion through CNNs," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.
[30] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2436–2442.
[31] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5761–5768.
[32] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.
[33] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," *arXiv preprint arXiv:2011.00032*, 2020.
[34] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *arXiv preprint arXiv:2012.03094*, 2020.
[35] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.
[36] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2464–2470.
[37] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. Kim, and P. Agrawal, "Learning to jump from pixels," *arXiv preprint arXiv:2110.15344*, 2021.
[38] H. W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *2015 Robotics: Science and Systems Conference, RSS 2015*. MIT Press Journals, 2015.
[39] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
[40] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.
[41] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4102–4109.
[42] I. Fischer, "The conditional entropy bottleneck," *Entropy*, vol. 22, no. 9, p. 999, 2020.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[44] "Unitree Robotics." [Online]. Available: http://www.unitree.cc/
[45] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," 2017.
[46] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.
[47] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

APPENDIX

A. PI-ARS Implementation

We describe implementation details of PI-ARS as follows.

1) Network Architecture for ϕ: In the visual-locomotion tasks we consider, an observation $\mathbf{s}_t$ contains a visual observation $\mathbf{s}_t^v$ and proprioceptive states $\mathbf{s}_t^p$. The visual observation $\mathbf{s}_t^v$ contains one depth image from the front camera and one from the rear (described in Section IV-B). Accordingly, our vision encoder ϕ_v contains two identical CNNs that independently map the front and rear images to two 64-d vectors and concatenates them into a 128-d representation output $\mathbf{z}_t^v$. Each CNN consists of 2 convolution layers with 3×3 kernels, 8 channels, stride size 1, followed by a 64-d linear projection, where each convolution and projection

is followed by a relu activation. $\mathbf{z}_t^v$ is then concatenated with proprioceptive states $\mathbf{s}_t^p$ and linearly projected with tanh activation to yield a 128-d representation output $\mathbf{z}_t$. These together give us the observation encoder ϕ, in which $\mathbf{z}_t = \phi(\mathbf{s}_t^v, \mathbf{s}_t^p) = \phi(\mathbf{s}_t)$.

2) Network Architecture for Auxiliary Functions: The auxiliary-learned function f maps $\mathbf{z}_t$ to a unit-length 128-d vector $\mathbf{z}_t^{\text{past}}$, which corresponds to the past, via a 2-layer MLP (64 units, 128 units) followed by l-2 normalization.

For the future observation, f only considers the visual observation $\mathbf{s}_{t+k}^v$ and ignores $\mathbf{s}_{t+k}^p$:

$$\mathbf{z}_{t+k}^{\text{future}} = h(\text{stopgrad}(\phi_v(\mathbf{s}_{t+k}^v))) \tag{6}$$

where h is another 2-layer MLP (64 units, 128 units), and stopgrad refers to the stop gradient operation. The output of f is a dot-product of $\mathbf{z}_t^{past}$ and $\mathbf{z}_{t+k}^{future}$.

For predicting the future rewards, we recurrently apply an auxiliary RNN cell g to encode a latent state and an action (i.e. the past) and output a reward prediction and the next latent state at each time step:

$$\hat{r}_t, \mathbf{z}'_{t+1} = g(\mathbf{z}_t, \mathbf{a}_t), \hat{r}_{t+1}, \mathbf{z}'_{t+2} = g(\mathbf{z}'_{t+1}, \mathbf{a}_{t+1}), \ldots, \hat{r}_{t+k-1}, \mathbf{z}'_{t+k} = g(\mathbf{z}'_{t+k-1}, \mathbf{a}_{t+k-1}).$$

g is a 3-layer MLP (128 units each with tanh activations), and a 128-to-1 linear layer branch is attached to the second layer of g to output reward predictions $\hat{r}_t, \ldots, \hat{r}_{t+k-1}$, one at each step. The initial latent state is $\mathbf{z}_t$.

3) Policy Network Head: The ARS-learned policy network head is a simple 3-layer MLP (64, 32, and $\dim(\mathcal{A})$ units, where $\mathcal{A}$ is the action space) with tanh activation that takes $\mathbf{z}_t$ and proprioceptive states $\mathbf{s}_t^p$ as input, and outputs an action. The output of the policy network is tanh-squashed to $[-1, 1]$ and subsequently re-scaled to the environment action bounds (Table I).

4) PI-ARS Training: In PI-ARS, we alternate between 1 step for ARS and 2 steps for PI. Each PI step uses a batch of 512 k-step trajectories from the replay buffer and performs gradient steps with a learning rate of 10^{-4}. We set $k = 5$ for all tasks but indoor navigation. For indoor navigation, we use $k = 30$ as the task nature requires a longer planning horizon. We also apply observation normalization, using running means and standard deviations.

5) ARS, SAC, PI-SAC Implementations: To ensure fair comparison for non-representation learning methods (ARS, SAC), we use the an identical policy network as used for PI-ARS, which combines a base encoder ϕ and a policy network head. Thus, all algorithms have access to the same capacity policy network. The critic in SAC and PI-SAC share the same base encoder ϕ with the policy (actor) network and we stop gradients from the policy head to the base encoder following [13]. We also apply the same observation normalization used for PI-ARS to these baseline methods.

B. Visual-Locomotion Task details

Here we describe additional details regarding the visual-locomotion tasks we used in our work. For the uneven stepping stone, quincuncial piles, and moving platform tasks we follow the design in prior work [3].

TABLE I
ACTION SPACE RANGES

action	lower bound	upper bound
Target local foothold	(-0.05, -0.05, -0.03)m	(0.1, 0.05, 0.03)m
Target peak swing height	0.05m	0.1m
Desired CoM height	0.42m	0.47m
Desired base roll	-0.1	0.1
Desired base pitch	-0.2	0.2
Desired base twist speed	-0.2	0.2

a) Uneven stepping stones: In this task we evaluate the ability for the policy to traverse stepping stones with varied heights. The widths, and lengths of stepping stones are sampled from $[0.55, 0.7]$, and $[0.5, 0.8]$ meters. The height offsets of neighboring stones are uniformly sampled in $[0.13, 0.2]m$, and a gap of $[0.05, 0.1]m$ is added between the stones. To successfully traverse this environment the agent needs to identify and avoid the gaps between stones and land the swing leg to the appropriate height for the next stone.

b) Quincuncial piles: In this task, we create a two-dimensional stepping stones to evaluate the robot's behavior in avoiding gaps in both forward and lateral direction. Specifically, each stone has an area of $0.15 \times 0.15m^2$ with a standard deviation of $0.015m$ in height, and is separated by $[0.13, 0.17]$ m from each other in both x and y directions. At the beginning of each episode, we also randomly rotate entire stone grid in $[0.1, 0.1]$ rad.

c) Moving platforms: Our proposed framework can also be applied to handle dynamic objects. In this example, we construct an environment with random stepping stones and allow each piece to move dynamically. Each platform follows a periodic movement whose magnitude and frequency are randomly sampled in $[0.10, 0.15]m$ and $[0.4, 1.0]Hz$, respectively. Also, we randomly pick half of the platforms to move horizontally and the rest vertically. This task requires the control policy to infer both the position and velocity of the platforms and thus presents a more challenging representation learning problem. To enable the model to infer velocity information, we stack a history of three recent images as input to the policy for this task, which slightly changes the observation spec.

d) Indoor navigation: For the indoor navigation task, we use a scan of the building interior with size about 25×15m. We randomly place 50 boxes of different sizes in the scanned scene to model obstacles during navigation. Among the 50 obstacles, 20% are sampled with length in $[1.6, 2]$m, width in $[0.7, 1]$m, and height in $[0.4, 1.2]$m, which are to mimic bigger obstacles like sofa. 40% are sampled with length and width in $[0.5, 0.8]$m and height in $[0.4, 1.4]$m, which represent smaller obstacles like chairs. The rest 40% are sampled with length and width in $[0.1, 0.4]$m and height in $[0.8, 2.0]$m, which correspond to taller objects like pillars.