Generating Scenarios from High-Level Specifications for Object Rearrangement Tasks

Van Waveren, Sanne; Pek, Christian; Leite, Iolanda; Tumova, Jana; Kragic, Danica

DOI
10.1109/IROS55552.2023.10341369

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2023

Citation (APA)
Van Waveren, S., Pek, C., Leite, I., Tumova, J., & Kragic, D. (2023). Generating Scenarios from High-Level Specifications for Object Rearrangement Tasks. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2023* (pp. 11420-11427). (IEEE International Conference on Intelligent Robots and Systems). IEEE. https://doi.org/10.1109/IROS55552.2023.10341369

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341369

Generating Scenarios from High-Level Specifications for Object Rearrangement Tasks

Sanne van Waveren$^{\star 1}$, Christian Pek$^{\star 2}$, Iolanda Leite[3], Jana Tumova[3], and Danica Kragic[3]

Abstract— **Rearranging objects is an essential skill for robots. To quickly teach robots new rearrangements tasks, we would like to generate training scenarios from high-level specifications that define the relative placement of objects for the task at hand. Ideally, to guide the robot's learning we also want to be able to rank these scenarios according to their difficulty. Prior work has shown how generating diverse scenario from specifications and providing the robot with easy-to-difficult samples can improve the learning. Yet, existing scenario generation methods typically cannot generate diverse scenarios while controlling their difficulty. We address this challenge by conditioning generative models on spatial logic specifications to generate spatially-structured scenarios that meet the specification and desired difficulty level. Our experiments showed that generative models are more effective and data-efficient than rejection sampling and that the spatially-structured scenarios can drastically improve training of downstream tasks by orders of magnitude.**

I. Introduction

Many robotic tasks might require a robot to learn how to rearrange objects according to a task specification that defines the relative placement of objects. For instance, for the table setting task in Fig. 1 we might specify that *the knife needs to be placed right of the plate and the plate on the placemat*. To learn and validate policies for such tasks, we ideally would like to generate diverse sets of scenarios from such high-level specifications and control how difficult the scenarios are to solve for the robot.

Several approaches have been proposed to generate scenarios. Often, new scenarios are created by applying random perturbations to initial configurations [1]. While straightforward, the generated scenarios do not necessarily adhere to task specifications and often require filtering. Procedural scenario generation approaches aim at generating scenarios according to task specifications. For instance, we can initiate a single environment from a PDDL specification [2], which requires us to specify the exact object placement. Another work generates scenarios from specifications that define relational constraints between objects [3]. While this approach can generate multiple scenarios per specification, it places objects sequentially without considering the environment's geometry which might result in invalid scenarios. Instead, [4], [5] use example scenarios to learn the distribution of object placements with generative models. While these approaches naturally capture scenario variations, they rely on human-curated training datasets and might generate scenarios that cannot be solved or are too difficult to solve for the robot.

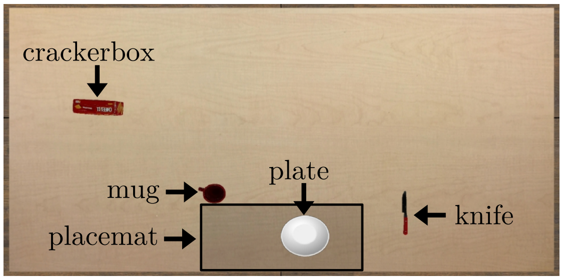


Fig. 1. Example spatial configuration for objects in a table setting task.

Curriculum learning approaches pace the robot's learning by training on scenarios with increasing difficulty [6]. Several works have learned to generate scenarios with various difficulty levels. For example, one can train an RL agent to generate scenarios with manually increasing difficulty [7] or using the training performance of a second agent trying to solve the scenario [8]. However, the agent might not be able to generate scenarios for each desired difficulty level. Alternatively, work has generated scenarios with varying difficulty by increasing the distance between initial and predefined goal states, which is known to increase the amount of exploration needed for the robot to solve the task [9], [10]. While these approaches successfully generate scenarios by controlling the difficulty, they require predefined goal states and cannot be generated from high-level specifications.

In this work, we address the challenge of generating diverse scenarios from high-level specifications while controlling the difficulty of generated scenarios. Specifically, we leverage a spatial logic to specify desired object configurations in Sec. III-A and generate a wide variety of scenarios using generative models in Sec. III-B. Our spatial logic provides us with a continuous measure of task difficulty by evaluating how close the scenarios meet the specification. In Sec. IV, we demonstrate that training with our spatially-structured scenarios can significantly improve the learning of downstream tasks and we investigate the scenario quality and diversity of our generative models on various specifications.

II. Related Work

We briefly discuss specification languages to define high-level tasks and review relevant works on procedural scene

*Authors contributed equally. This work was performed at KTH.

[1] School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA, sanne@gatech.edu

[2] Department of Cognitive Robotics, Delft University of Technology, Delft, the Netherlands, c.pek@tudelft.nl

[3] Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden, {iolanda,tumova,dani}@kth.se

This work was partially funded by the Swedish Foundation for Strategic Research (SSF FFL18-0199) and by the HORIZON-CL4-2021-HUMAN-01 ELSA project. The authors are affiliated with Digital Futures.

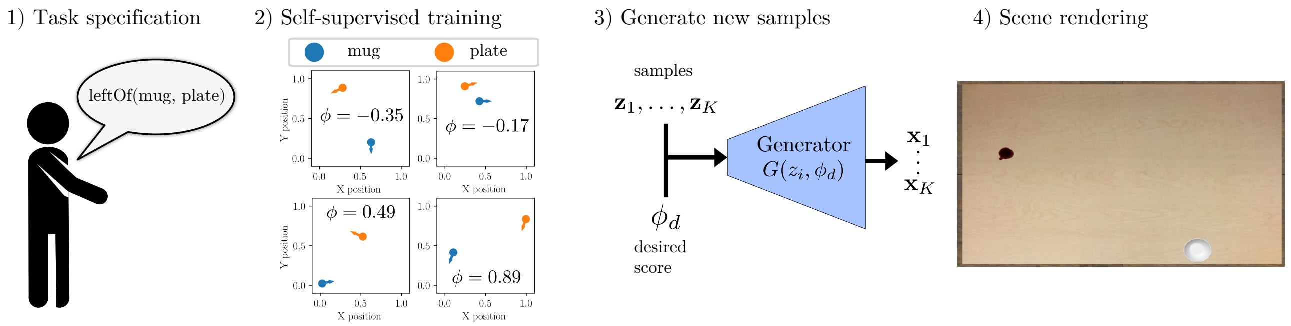


Fig. 2. Overview of framework to automatically generate scenarios from high-level task specifications with desired spatial configurations.

generation and using generative models for data generation.

a) Task specifications: Task specifications are part of planning frameworks, such as STRIPS [29], ADL [30], and PDDL [31, 32] to express various robotics tasks, e.g., through goal conditions and predicates. Temporal logics have become increasingly popular to specify high-level tasks in robot-agnostic fashions. Linear Temporal Logics (LTL) [11], [12] and Metric Interval Temporal Logics (MITL) [13] provide temporal operators and Boolean propositions, e.g., to specify that an object is *eventually* placed at the desired location. These logics usually only provide binary evaluations: a scenario either satisfies or violates a specification. Signal Temporal Logics (STL) [14], [15] consider continuous domains and add quantitative semantics, i.e., a continuous real value that measures the extent to which a scenario satisfies or violates a specification [16]. These semantics can be used to guide planners to satisfy specifications over time [17], [18]. We use an STL-inspired spatial logic to specify and generate scenarios with various difficulty levels. Our quantitative semantics are defined over object distances which allow us to control a scenario's difficulty, e.g., scenarios that are closer to satisfying the task are usually easier to solve because they require more exploration than scenarios that are further from satisfying the task.

b) Procedural scenario generation: Procedural scenario generation has been a popular topic in the game community to create new levels with varying difficulties [19]–[21]. Computer graphics approaches generate new scenarios through rearranging objects [22] and perturbations [23]. In robotics, new scenarios are important for training and validation [24]–[26], e.g., to identify failure cases in shared autonomy manipulation tasks [27] or to pace the learning process [6], [28]–[32]. These scenarios can be generated through adaptive parameterization [33], [34], reinforcement learning [7], or generative networks [5], [8]. Prior work has generated scenarios directly from given specifications, e.g., from PDDL [2], probabilistic specifications [3], however, they require one to specify the exact object placement. Similar to our work, [5] and [4] generate scenarios using generative models, however, they rely on human-curated data. Instead, we automatically generate training data from a high-level specification, e.g., *the mug should be left of the plate*. We then train generative models to learn the training data's distribution.

c) Generative models: Generative models have shown success to approximate data distributions and generate diverse data, e.g., for grasps [35], [36], to predict motions of objects [37] and humans [38], [39], and to understand scenes [4], [40]. Several generative models have been proposed: Diffusion [41], [42] and flow-based models [43], [44] are powerful but require large training and generation times. Variational Autoencoders (vAEs) [45], [46] focus on learning encodings as distributions through encoder and decoder networks, whereas Generative Adversarial Networks (GANs) [47] aim at generating accurate data through the competing generator and a discriminator networks. Adversarial Autoencoders (AAEs) [48], [49] combine the advantages of AEs and GANs by forcing the latent embedding to follow chosen distributions. We choose GANs and AAEs to generate diverse scenarios due to their efficient training and generation times and their ability to model complex data distributions.

III. Spatially-Structured Data Generation

We want to generate a set of scenarios from a logical specification that defines the desired spatial relations between objects in the scenarios. To control the scenarios' difficulty, we need a metric that indicates how close the scenarios meet the specification. Within desired bounds of the metric, we want to generate as diverse scenarios as possible.

To create specifications, we summarize a spatial logic in Sec. III-A. This spatial logic defines quantitative semantics allowing us to compute the extent to which a scenario satisfies a given specification $\mathcal{R}$, referred to as the satisfaction value ϕ. For instance, the specification may define the task specification $\mathcal{R}_{\text{task}}$, e.g., the mug should be left of the plate. In Sec. III-B, we describe how we condition generative models on ϕ to generate scenarios corresponding to a desired satisfaction value ϕ_d. Fig. 2 illustrates our scenario generation pipeline. First, we provide scenario specifications by combining objects (e.g., mug), spatial relations (e.g., leftOf), and Boolean operators. We then generate a training dataset, compute the satisfaction value ϕ for each sample, and train a generative model conditioned on ϕ. Finally, we create a set of scenarios by randomly sampling the generative model's latent space $\mathbf{z}$ for various $\phi_d \in [\phi_{\text{min}}, \phi_{\text{max}}]$.

A. Spatial relations and their quantitative semantics

We consider scenarios with $N \in \mathbb{N}_+$ objects and describe their pose $p_i, i \in \{1, \dots, N\}$, by its center position and orientation in space, i.e., $p_i \in SE(j), j \in \{2, 3\}$, where SE is the special Euclidean group [50, Sec. 4.2]. In the case of $SE(2)$, each pose corresponds to $p = (p_x, p_y, p_\theta)^T \in \mathbb{R}^3$, where p_x, p_y and p_θ describe the position and orientation in the plane, respectively. The scenario configuration $\mathbf{x} = (p_1, \dots, p_N)^T$ describes the poses of the N objects.

Spatial relations between objects can be described through distances. Let us therefore introduce the distance between two objects, p_i and p_j, as $\mathrm{dist}(p_i, p_j) = ||p_i - p_j||_2$. Further, $\mathrm{proj}_\mathrm{a} : p \mapsto p_\mathrm{a}$ projects p to the given component $\mathrm{a} \in \{x, y, \theta\}$. We define spatial relations between two objects, p_i and p_j according to a given world-coordinate system:

Definition 1 (Spatial relations)**.** *The projection-based spatial relations* leftOf, rightOf, belowOf, aboveOf *are defined as:*

$$\begin{aligned}\mathrm{leftOf}(p_i, p_j) &:= \mathrm{proj}_x(p_i) \leq \mathrm{proj}_x(p_j),\\ \mathrm{rightOf}(p_i, p_j) &:= \neg\, \mathrm{leftOf}(p_i, p_j),\\ \mathrm{belowOf}(p_i, p_j) &:= \mathrm{proj}_y(p_i) \leq \mathrm{proj}_y(p_j),\\ \mathrm{aboveOf}(p_i, p_j) &:= \neg\, \mathrm{belowOf}(p_i, p_j),\end{aligned}$$

where $\neg$ denotes the Boolean negation. The distance-based spatial relations closeTo, farFrom *are defined as:*

$$\begin{aligned}\mathrm{closeTo}(p_i, p_j) &:= \mathrm{dist}(p_i, p_j) \leq \epsilon_1,\\ \mathrm{farFrom}(p_i, p_j) &:= \mathrm{dist}(p_i, p_j) \geq \epsilon_2,\end{aligned}$$

where ϵ_1, ϵ_2 are user-defined parameters. We define the angle relation angle *as:*

$$\mathrm{angle}(p_i, \Theta) := \Theta - \epsilon_3 \leq \mathrm{proj}_\theta(p_i) \leq \Theta + \epsilon_3,$$

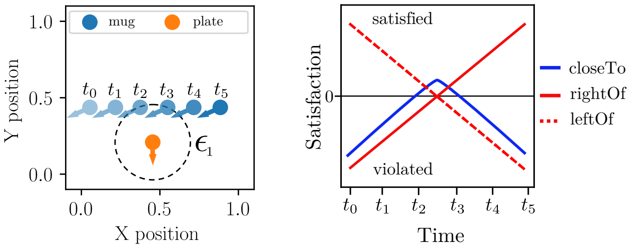
where ϵ_3 is a user-defined parameter to allow small numerical deviation from the desired angle Θ. Note that for brevity we do not check whether angles are outside of $[0, 2\pi)$.

By combining the spatial relations with Boolean operators, we can compose complex scenario specifications $\mathcal{R}$ using the following composition grammar: $\mathcal{R} := \top \,|\, \mathrm{rel}_1(p) \,|\, \mathrm{rel}_2(p_i, p_j) \,|\, \neg\mathcal{R} \,|\, \mathcal{R}_i \wedge \mathcal{R}_j \,|\, \mathcal{R}_i \vee \mathcal{R}_j$, where $\top$ is the Boolean *true* statement and rel_1 and rel_2 are unary and binary spatial relations, respectively.

We compute a scenario's difficulty as the extent to which the scenario satisfies the task specification. Inspired by STL [16], we exploit that our spatial relations are based on distances and define quantitative semantics to compute how much a scenario $\mathbf{x}$ satisfies or violates a specification $\mathcal{R}$:

Definition 2 (Quantitative semantics of spatial relations)**.** *We define the quantitative semantics $\phi(\mathbf{x}, \mathcal{R}) \in \mathbb{R}$ of a given specification $\mathcal{R}$ for a scenario $\mathbf{x}$ as:*

$$\begin{aligned}\phi(\mathbf{x}, \top) &:= \infty\\ \phi(\mathbf{x}, \neg\mathcal{R}) &:= -\,\phi(\mathbf{x}, \mathcal{R}),\\ \phi(\mathbf{x}, \mathcal{R}_1 \wedge \mathcal{R}_2) &:= \min\big(\,\phi(\mathbf{x}, \mathcal{R}_1), \phi(\mathbf{x}, \mathcal{R}_2)\big),\\ \phi(\mathbf{x}, \mathcal{R}_1 \vee \mathcal{R}_2) &:= \max\big(\,\phi(\mathbf{x}, \mathcal{R}_1), \phi(\mathbf{x}, \mathcal{R}_2)\big).\end{aligned}$$



(a) Scenario

(b) Satisfaction values

Fig. 3. (a) scenario where mug moves from left to right while the plate remains static. (b) satisfaction values of three spatial relations over time.

We define ϕ for single spatial relations as:

$$\begin{aligned}\phi(\mathbf{x}, \mathrm{leftOf}(p_i, p_j)) &= \mathrm{proj}_x(p_j) - \mathrm{proj}_x(p_i),\\ \phi(\mathbf{x}, \mathrm{closeTo}(p_i, p_j)) &= \epsilon_1 - \mathrm{dist}(p_i, p_j),\\ \phi(\mathbf{x}, \mathrm{angle}(p_i, \Theta)) &= \min\big(\,\mathrm{proj}_\theta(p_i) - \Theta + \epsilon_3,\\ &\qquad \Theta + \epsilon_3 - \mathrm{proj}_\theta(p_i)\big).\end{aligned}$$

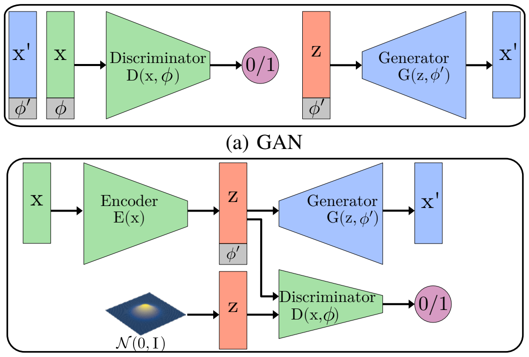
The quantitative semantics for the other spatial relations are computed analogously by rearranging their conditions to be satisfied when greater than or equal to zero.

Fig. 3a illustrates a scenario in which we move a mug from left (t_0) to right (t_5) around a static plate. Fig. 3b shows the computed satisfaction values for the specifications $\{\mathrm{closeTo}, \mathrm{rightOf}, \mathrm{leftOf}\}(\mathrm{mug}, \mathrm{plate})$. The distance between objects encodes how much the relation rightOf is satisfied: While the mug is placed left of the plate (e.g., at t_0), its X-coordinate is smaller than that of the plate, the resulting satisfaction value is negative, i.e., violating. If the mug is right of the plate, its X-coordinate is greater, and the satisfaction value is positive, i.e., satisfying. The leftOf relation is computed as the negative value of rightOf. The $closeTo$ relation is satisfied when the mug is in the plate's ϵ_1-neighborhood. The satisfaction values correspond to the distance of the mug to the ϵ_1-neighborhood.

For violating scenarios, a robot would need to move the objects further the more relations are violated, i.e., scenario t_0 requires more exploration than scenario t_2 when we consider $\mathcal{R} = \mathrm{rightOf}(\mathrm{mug}, \mathrm{plate})$. Thus, when we generate scenarios with lower satisfaction values ϕ, we increase the scenarios' difficulty.

B. Training of conditioned generative models

To train our generative models, we first create a dataset $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}, M \in \mathbb{N}_+$, of random scenario configurations with $N \in \mathbb{N}_+$ objects. The objects are randomly placed and oriented in the workspace by drawing from a probability distribution, e.g., a uniform distribution. In addition to a task specification $\mathcal{R}_\mathrm{task}$, we might also want to consider a safety specification $\mathcal{R}_\mathrm{safe}$ in object rearrangement tasks. We use the semantics in Def. 2 to label each training sample $\mathbf{x}_i$ with its corresponding satisfaction values $\phi_i = \big(\phi(\mathbf{x}_i, \mathcal{R}_\mathrm{task}), \phi(\mathbf{x}_i, \mathcal{R}_\mathrm{safe})\big)$. The created training dataset $\mathcal{X}_\mathrm{train}$ has the form $\mathcal{X}_\mathrm{train} = \{(\mathbf{x}_1, \phi_1), \dots, (\mathbf{x}_M, \phi_M)\}$. With $\mathcal{X}_\mathrm{train}$, we then train conditioned generative models $\mathrm{G} : (\mathbf{z}, \phi_d) \mapsto \mathbf{x}'$ that generate scenarios $\mathbf{x}' = \mathrm{G}(\mathbf{z}, \phi_d)$



(a) GAN

(b) AAE

Fig. 4. GAN (a) and AAE (b) conditioned on the satisfaction value ϕ of a scenario $\mathbf{x}$.

from latent vectors $\mathbf{z} \in \mathbb{R}^L$ and desired satisfaction values ϕ_d such that $\phi(\mathbf{x}', \mathcal{R}) \approx \phi_d$.

1) Generative Adversarial Network: Fig. 4a illustrates the GAN's architecture. The generator $\mathrm{G}(\mathbf{z}, \phi)$ learns to produce realistic scenario configuration that match ϕ_d, while the discriminator $\mathrm{D} : (\mathbf{x}, \phi) \mapsto b \in [0, 1]$, learns to classify whether $\mathbf{x}$ is a real scenario given ϕ. During training, both networks play a zero-sum game to become better at generating/classifying scenarios. We train the generator by randomly sampling latent vectors from $\mathbf{z}$ and labels ϕ' from a normal distribution and generating scenarios as $\mathbf{x}' = \mathrm{G}(\mathbf{z}, \phi')$. We measure the generator's ability to fool the discriminator through the binary cross entropy loss (BCE) between the discriminators predictions and the ground truth $\mathbf{1}$ as: $\ell_G(\mathbf{z}) = \mathrm{BCE}\left(\mathrm{D}(\mathrm{G}(\mathbf{z}, \phi'), \phi'), \mathbf{1}\right)$. The discriminator distinguishes between real and fake scenarios, encoded through the loss function $\ell_D = 0.5\big(\mathrm{BCE}(\mathrm{D}(\mathbf{x}, \phi), \mathbf{1}) + \mathrm{BCE}(\mathbf{x}', \phi'), \mathbf{0}\big)$, where $\mathbf{x}$ is a real scenario from the training set (real) and $\mathbf{x}'$ is a generated scenario (fake), i.e., $\mathbf{x}' = \mathrm{G}(\mathbf{z}, \phi')$.

2) Adversarial Auto Encoder: Fig. 4b illustrates the AAE's architecture. We use a classical AE structure, e.g., as shown in [51, Ch. 14], with the mean squared error as the reconstruction loss. Since our scenario representation is minimal by using objects poses, the AE is overcomplete, i.e., it increases the input data's dimensionality, which is beneficial for learning features. We add an adversarial network to enforce a normal distribution as a prior on the latent representation. The discriminator distinguishes between samples from a true Gaussian and fake samples from the AE's latent representation (see Fig. 4b). We use the BCE loss for the generator (i.e., the AE's encoder) and discriminator.

IV. Evaluation

We first investigate whether spatially-structured datasets with our difficulty measure can speed up the learning of downstream tasks. To that end, Sec. IV-A compares the performance of two curriculum learning strategies that use spatially-structured datasets to regular reinforcement learning in sparse reward settings. Afterwards, we evaluate our generative models: 1) How accurate can conditioned generative models generate scenarios for desired satisfaction values? 2) To what extent can we generate scenarios with a high diversity, i.e., with a wide range of object placements under a specification? 3) How efficient are generative models compared to a classical rejection sampling strategy? We study these questions in table setting (Sec IV-B and Sec. IV-C) and object ordering tasks (see Sec. IV-D).

All experiments are carried out with an Apple M1 processor, 16GB of memory, Pytorch 1.1.0 [52], the Adam optimizer [53], and Pybullet to render scenarios [54]. We use e, b, lr to denote the number of epochs, batches, and learning rate, respectively. We report statistics over 3 random training seeds. The code is publicly available at github.com/sannevw/scenario_generation and a video of our experiments can be found in the supplementary material.

A. Learning with spatially-structured data

We demonstrate that learning with spatially-structured data from our generative models increases learning performance. To that end, we consider two tasks, $\mathcal{R}^A_{\text{task}}$ and $\mathcal{R}^B_{\text{task}}$ (see Tab. I, in a Q-Learning setup [55]. In $\mathcal{R}^A_{\text{task}}$, the agent needs to move the mug on top of the plate, and in $\mathcal{R}^B_{\text{task}}$, the agent needs to move the spoon left and below of the plate. The agent can move objects on the table step-wise left/right/up/down and receives a reward $r = 10$ if the specification is satisfied, $r = -10$ if an object is moved off the table, and $r = 0$ otherwise. This goal-reward representation is highly sparse and difficult to solve [56], [57]. We compare three learning strategies: 1) training with scenarios where objects are placed randomly (baseline), 2) training with spatially-structured scenarios generated by linearly decreasing ϕ over episodes from 0 to $\phi_{\text{min}} < 0$ (curriculum learning), and 3) training with spatially-structured scenarios where we decrease ϕ only when the agent can consistently solve the currently presented scenarios (self-paced curriculum learning).

Fig. 5 shows the required number of episodes for each strategy until the policy converged. We varied the environment size $M \in \{10, 20, \dots, 100\}$, where the table is square with edge length $2M$. For $M = 100$ in task $\mathcal{R}^A_{\text{task}}$, the baseline requires more than 8×10^6 episodes to converge. Our simple curriculum learning strategy already requires 50% less episodes. The self-paced learning strategy is significantly more efficient and requires orders of magnitude less episodes, e.g., 23 times less for $M = 100$. We observe a similar trend for task $\mathcal{R}^B_{\text{task}}$, in which self-paced learning with spatially-structured scenarios is drastically faster. We further noticed that our curriculum learning strategies result in less simulation resets where the agent moves objects off the table. Note that for smaller environment sizes M one can obtain similar results without using generative models by ordering the training data on their satisfaction values and then learn with scenarios with decreasing values.

B. Experiment 1 - Learned scenario distributions

We compare the GAN's and AAE's performance for scenarios with two objects, a plate and a mug. The models are conditioned on a task and safety specification (see Tab. I), where $\mathcal{R}^1_{\text{task}}$ encodes that the mug is placed left and

TABLE I
SCENARIO SPECIFICATIONS OF EACH EXPERIMENT.

Experiment	Specification
Downstream - Sec.IV-A	$\mathcal{R}^{A}_{\text{task}} = \text{closeTo}(\text{mug}, \text{plate}), \epsilon_1 = 0, \mathcal{R}^{B}_{\text{task}} = \text{leftOf}(\text{spoon}, \text{plate}) \wedge \text{belowOf}(\text{spoon}, \text{plate})$
Exp. 1 - Sec. IV-B	$\mathcal{R}^{1}_{\text{task}} = \text{leftOf}(\text{mug}, \text{plate}) \wedge \text{aboveOf}(\text{mug}, \text{plate}), \mathcal{R}^{1}_{\text{safe}} = \neg\,\text{closeTo}(\text{mug}, \text{plate}), \epsilon_1 = 0.25$
Exp. 2 - Sec. IV-C	$\mathcal{R}^{2}_{\text{task}} = \text{onPlacemat}(\text{plate}) \wedge \text{aboveOfPlacemat}(\text{mug}) \wedge \text{rightOfPlacemat}(\text{knife}) \wedge \text{angle}(\text{knife}, \text{NORTH}) \wedge \text{aboveOf}(\text{crackerbox}, \text{plate}) \wedge \text{leftOf}(\text{crackerbox}, \text{plate})$
Exp. 3 - Sec. IV-D	$\mathcal{R}^{3}_{\text{task}} = \left(\bigwedge_{i\in\{1,N-1\}} \text{leftOf}(p_i, p_{i+1})\right) \vee \left(\bigwedge_{i\in\{1,N-1\}} \text{aboveOf}(p_i, p_{i+1})\right) \vee \left(\bigwedge_{i\in\{1,N-1\}} \text{rightOf}(p_i, p_{i+1})\right) \vee \left(\bigwedge_{i\in\{1,N-1\}} \text{belowOf}(p_i, p_{i+1})\right)$

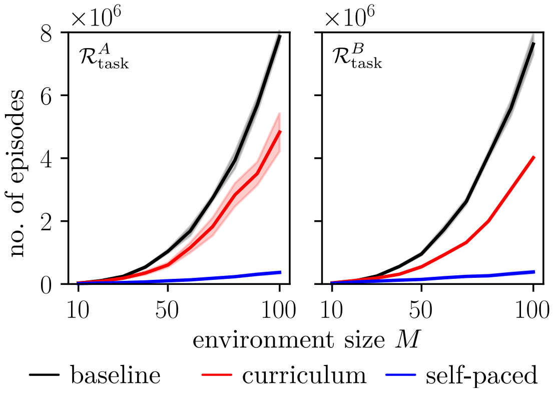


Fig. 5. Required episodes for each learning strategy. Curves and shaded area show mean and standard deviation over 5 different seeds.

above of the plate, and $\mathcal{R}^{1}_{\text{safe}}$ encodes a minimum distance of $\epsilon_1 = 0.25$ between them. We train with $M = 20$k training samples, of which 17641 (88%) samples satisfy at least one specification and 5057 (25%) samples satisfy both specifications, $e = 50$, $b = 32$, and $lr = 0.001$. Fig. 6 shows rendered scenarios generated by the AAE. We query the AAE to generate scenarios for varying task ϕ_t (top row) and safety ϕ_s (bottom row) satisfaction values independently, while satisfying the non-varied other specification.

In general, we condition the scenario distribution on the satisfaction value. With infinitely many random samples, we could perfectly represent this conditioned scenario distribution. However, this sampling strategy is infeasible. With our generative models, we aim to approximate this conditioned distribution instead. We qualitatively analyze the learned scenario distributions for fixed desired satisfaction value pairs, $(\phi_t, \phi_s) = (0.19, 0.08)$ and $(\phi_t, \phi_s) = (-0.17, -0.09)$, chosen randomly from satisfying training examples.

We map the generated scenarios $\mathbf{x}$ into a two dimensional latent space using Isomaps [58], see Fig. 7. Both models generate new scenarios that are not present in the training data, i.e., gaps in the latent space are filled. We observe that the AAE covers a larger area of the latent space than the training examples, whereas the GAN covers a smaller part of the scenario distribution. To evaluate the models' accuracy a, we compared the ground truth value with the actual value (by computing their satisfaction value) of the generated scenarios for the satisfaction value pair $(\phi_t, \phi_s) = (0.19, 0.08)$ to 5K queried values, i.e., $a_i = \phi(\mathbf{x}_i, \mathcal{R}_{\text{task}}) - \phi_t$ (safety accuracy is computed analogously). The GAN achieved a higher accuracy with averages of -0.007 (std 0.038) and 0.002 (std 0.035) for ϕ_t and ϕ_s, respectively. The AAE had a lower accuracy with averages of -0.085 (std .003) and 0.117 (std 0.157) for ϕ_t and ϕ_s.

C. Experiment 2 - Specifying table setting tasks

We consider a table setting scenario with four objects: a plate, a mug, a knife and a crackerbox. We define a rectangular area where the plate could be placed, which we refer to as placemat in our specifications. An advantage of our spatial logic is that we can easily compose new relations from existing ones. For instance, the relation onPlacemat encodes that an object p needs to be on the placemat:

$$\text{onPlacemat}(p) := \text{rightOf}(p, P_L) \wedge \text{leftOf}(p, P_R) \wedge \\ \text{belowOf}(p, P_T) \wedge \text{aboveOf}(p, P_B),$$

where P_L, P_R, P_T, P_B correspond to the center points of the placemat's left, right, top, and bottom side. The relations aboveOfPlacemat and rightOfPlacemat are similarly defined, and we use $\text{NORTH} = \pi/2$ to orient objects upwards. We condition the models on $\mathcal{R}^{2}_{\text{task}}$ (see Tab. I): the plate needs to be on the placemat, the mug close to and above the placemat, the knife close to and right of the placemat with the blade pointing north, and the crackerbox left and above of the plate. We created $M = 500$k training samples, however, due to the complex specification, only 5 (0.001%) samples initially satisfied the $\mathcal{R}^{2}_{\text{task}}$. We perturbed these satisfying examples by adding random noise, resulting in 607782 samples of which 107787 (18%) are satisfying, $e = 50$, $b = 512$, and $lr = 0.001$. Fig. 1 visualizes a satisfying scenario generated by the AAE with $\phi_t = 0.03$.

We computed the models' accuracy for $\phi_t = (0.03)$ and $5k$ generated scenarios. The GAN achieved a higher accuracy with an average of -0.025 (std 0.030) than the AAE with an average of -0.445 (std 0.271) for ϕ_t. Fig. 8 shows 700 generated scenarios and 700 random scenarios from the training data, all with $\phi_t = 0.03$ to compare the learned scenario distributions. Again, we observe that the GAN covers a smaller part of the distribution, whereas the AAE covers a much wider area than the training data.

D. Experiment 3 - Sorting objects

We use scenarios with $N \in \{3, 5, 7\}$ objects that need to be sorted either horizontally or vertically according to $\mathcal{R}^{3}_{\text{task}}$ in Table I. We created training data with 56% positive samples for all three scenarios, resulting in a total of $M =$

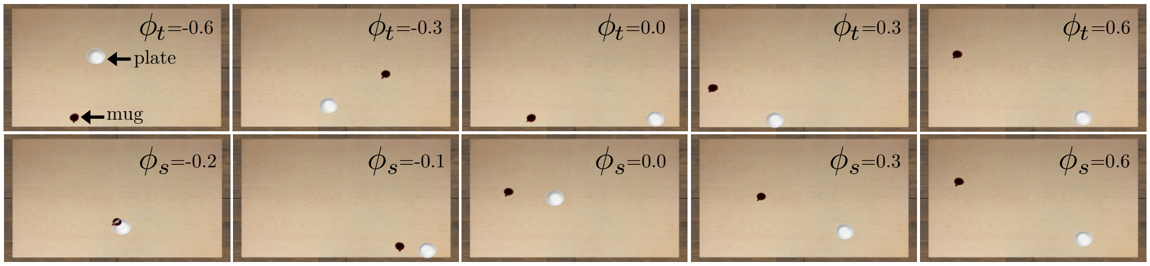


Fig. 6. Renderings of Exp. 1. The top row varies the task satisfaction value ϕ_t while satisfying safety and the bottom row the safety satisfaction value ϕ_s while satisfying the task.

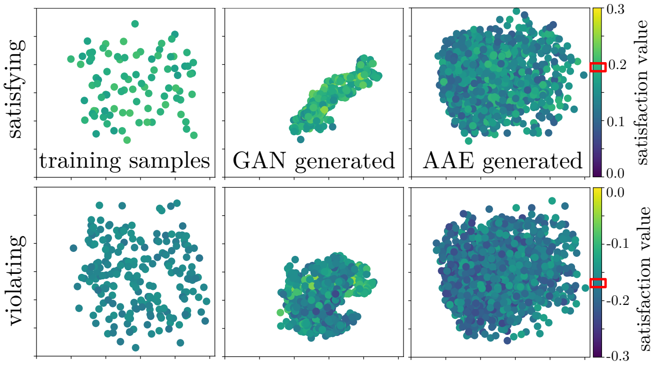


Fig. 7. Scenario distributions in Exp. 1 for fixed satisfying/violating values, mapped into a 2D latent space. We colored the points according to the task satisfaction value ϕ_t, where the red bar indicates the queried ϕ_t.

20k for 3 objects, $M \approx$ 110k, and $M \approx$ 114k for 5 and 7 objects, respectively. Fig. 9 shows renders of scenarios generated by the GAN.

We computed the models' accuracy for $\phi_t = (0.05)$ and $5k$ generated scenarios. The models achieved high accuracy across all numbers of objects: for 3 objects (GAN: $\mu = -0.016$, std= 0.095, AAE: $\mu = -0.052$, std= 0.122), 5 objects (GAN: 0.019, std 0.093, AAE: $\mu = -0.015$, std= 0.068), and 7 objects (GAN: $\mu = -0.085$, std= 0.106, AAE: $\mu = -0.068$, std= 0.063).

E. Comparison to naive rejection sampling

We compare the GAN and AAE with naive rejection sampling for Exp. 1 (placing a mug and a plate) and Exp. 2 (table setting). For complex specifications with multiple objects, the feasible scenario space for a given satisfaction value might be a fraction of the full scenario space. As a consequence, rejection sampling would reject many samples before we find satisfying ones. We evaluated how many samples we require to obtain a similar number of generated scenarios by our models for a given queried satisfaction value (we used the same values as in the previous sections).

For Exp. 1, the GAN generates around 8.1k scenarios with 20k training samples, while we required 700k random samples, corresponding to a 35x times increase. For Exp. 2, the AAE generates 4.4k samples for 500k training data. Rejection sampling was not successful with a time bound of 7 hours on our hardware: we only found 60 valid scenarios in 100M random samples, corresponding to minimum 200 times more data. Even if we perform rejection sampling on our boosted training data, this approach will generate the data seen in Fig. 8 and the AAE can create scenarios with more diverse object poses, e.g., other positions and rotations of the crackerbox. Secondly, for on-the-fly scenario generation on hardware, we compared the required space to store a lookup table from rejection sampling (with a resolution of 0.025 in the satisfaction value range and 24Byte per object represented by three floats) with the space of the network parameters. For Exp. 1, we require 5600MB for the table compared to 0.5MB for the network (increase of 11.2K times). For Exp. 2, we require 28400MB of data compared to 0.4MB for the network (increase of 71k).

V. Discussion and Future Work

Our results showed that we can successfully generate spatially-structured synthetic datasets by conditioning generative models on satisfaction values of spatial logic specifications, and that these spatially-structured datasets can be used in curriculum learning to improve the learning of sparse reward manipulation tasks.

We considered spatial relations, such as $\mathrm{leftOf}, \mathrm{rightOf}$ and $\mathrm{aboveOf}$, for object rearrangement tasks. However, these relations could be extended by defining new spatial relations. In Sec IV-C, we described how we can compose new spatial relations, e.g., $\mathrm{onPlacemat}$. We can also extend the existing relations, e.g., define partial/full versions of $\mathrm{aboveOf}$, and add more diverse ones, e.g., $\mathrm{overlapping}$. Our approach can be combined with specifications that define scenarios in PDDL [2] by converting them into spatial logics. However, considering more relations can also result in more complex specifications which might be difficult to express for users. In those cases, it might be beneficial to augment human-curated scenarios with spatially-structured datasets generated from logic specifications. Additionally, our quantitative semantics only capture spatial constraints, for many robotics tasks it is important to extend it to include temporal aspects, allowing us to create temporally coherent scenario sequences with time bounds, e.g., the mug should be moved to the placemat within 30 seconds.

Our focus was to generate scenarios for a desired task specification. We currently have to train a generative model per specification. Alternatively, a natural extension would be to generate scenarios for a wide range of goal configurations

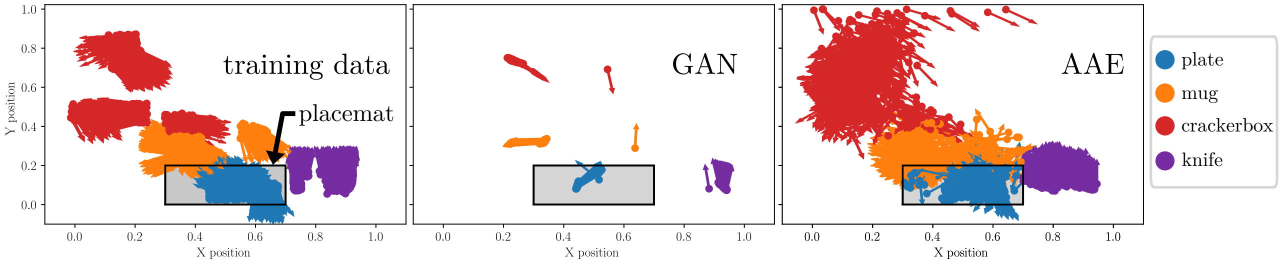


Fig. 8. Generated scenarios for $\phi_t = 0.03$ in Exp. 2 with 700 scenarios per figure.

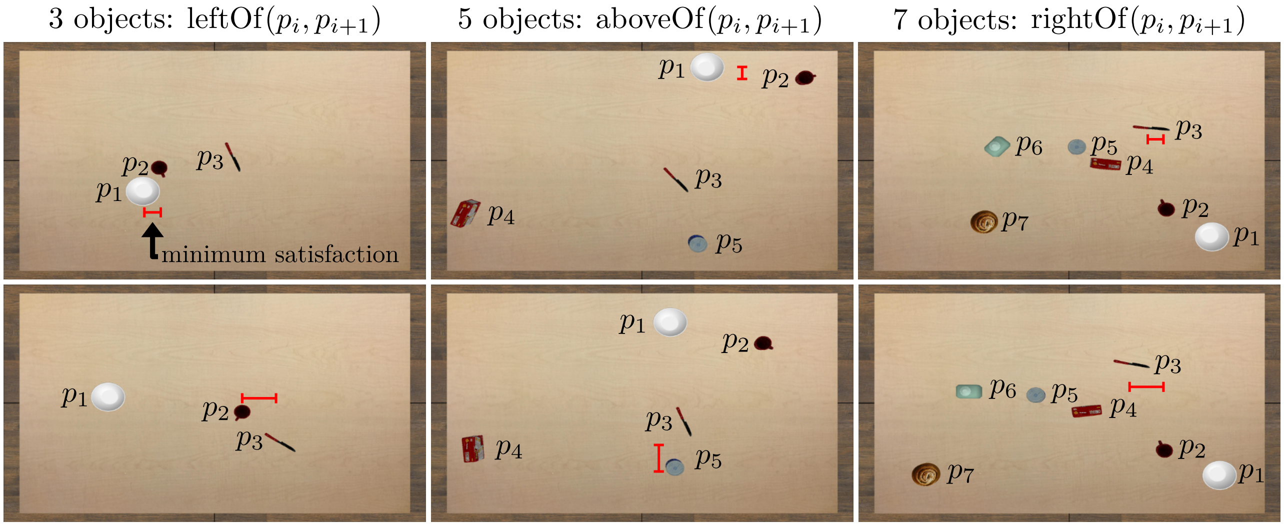

3 objects: $\text{leftOf}(p_i, p_{i+1})$
5 objects: $\text{aboveOf}(p_i, p_{i+1})$
7 objects: $\text{rightOf}(p_i, p_{i+1})$
p_1
p_2
p_3
p_4
p_5
p_6
p_7
minimum satisfaction

Fig. 9. Generated scenarios in Exp. 3. The top row shows $\phi_t = 0.05$ and the bottom row $\phi_t = 0.1$. The red bar indicates the closest distance between two objects, equal to ϕ_t, according to the spatial relations $\text{leftOf}, \text{aboveOf}, \text{rightOf}$ for the left, center, and right column, respectively.

without retraining the generative model to enable handling multiple specifications at once.

While our experiments demonstrated that spatially-structured datasets can drastically improve the learning of downstream tasks, we would like to explore the performance of other curriculum strategies, e.g., different pacing functions, over a wider variety of tasks. Furthermore, future work could investigate student-teacher strategies [6] that select which scenario from our generated spatially-structured dataset the agent should explore next to improve its policy.

VI. Conclusions

We condition generative models on spatial logic specifications to create spatially-structured scenarios of a desired difficulty for object rearrangement tasks. Our spatial logic allows one to specify high-level specifications that define spatial relations between objects, e.g., $\text{leftOf}(\text{mug}, \text{plate})$, and provides quantitative semantics to compute satisfaction values that denote how much scenarios satisfies or violates the desired specification. We showed that generative models are more effective and data-efficient than rejection sampling and that the spatially-structured scenarios can drastically improve training by orders of magnitude compared to traditional reinforcement learning. Our experiments further showed that AAEs are better suited when a large variety of scenarios is desired, e.g., for training, whereas GANs are preferred to generate accurate scenarios for given satisfaction values, e.g., for validation.

References

[1] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols,” *arXiv preprint arXiv:1502.03143*, 2015.

[2] T. Silver and R. Chitnis, “PDDLGym: Gym environments from PDDL problems,” *arXiv preprint arXiv:2002.06432*, 2020.

[3] C. Innes and S. Ramamoorthy, “ProbRobScene: A probabilistic specification language for 3D robotic manipulation environments,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2021, pp. 9446–9452.

[4] G. Izatt and R. Tedrake, “Generative modeling of environments with scene grammars and variational inference,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 6891–6897.

[5] M. C. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, “On the importance of environments in human-robot coordination,” in *Robotics: Science and Systems*, 2021.

[6] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, “Curriculum learning: A survey,” *Int. Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.

[7] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, “PCGRL: Procedural content generation via reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 95–101.

[8] P. Bontrager and J. Togelius, “Learning to generate levels from nothing,” in *Proc. of the IEEE Conference on Games*, 2021, pp. 1–8.

[9] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, “Reverse curriculum generation for reinforcement learning,” in *Conf. on Robot Learning*. PMLR, 2017, pp. 482–495.

[10] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone, "BaRC: Backward reachability curriculum for robotic reinforcement learning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*. IEEE, 2019, pp. 15–21.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[12] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 1988–1993.

[13] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM*, vol. 43, no. 1, pp. 116–146, 1996.

[14] A. Puranic, J. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic," in *Conf. on Robot Learning*, 2021, pp. 2228–2242.

[15] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. of Robotics: Science and Systems XII*, 2016.

[16] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Int. Conf. on Formal Modeling and Analysis of Timed Systems*, 2010, pp. 92–106.

[17] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 3834–3839.

[18] L. Lindemann and D. V. Dimarogonas, "Robust motion planning employing signal temporal logic," in *Proc. of the American Control Conference*, 2017, pp. 2950–2955.

[19] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016.

[20] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Trans. on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[21] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *Proc. of the IEEE Conference on Games*, 2019, pp. 1–8.

[22] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3D object arrangements," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 1–11, 2012.

[23] L. Majerowicz, A. Shamir, A. Sheffer, and H. H. Hoos, "Filling your shelves: Synthesizing diverse style-preserving artifact arrangements," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 11, pp. 1507–1518, 2013.

[24] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "MotionBenchMaker: A tool to generate and benchmark motion planning datasets," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, 2021.

[25] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *Int. Conf. on machine learning*, 2020, pp. 2048–2056.

[26] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*, 2020, pp. 1094–1100.

[27] M. C. Fontaine and S. Nikolaidis, "A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy," in *Robotics: Science and Systems*, 2021.

[28] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7382–7431, 2020.

[29] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic Curriculum Learning For Deep RL: A Short Survey," in *Proc. of the Int. Joint Conf. on Artificial Intell.*, 2021, pp. 4819–4825.

[30] K. Fang, Y. Zhu, S. Savarese, and L. Fei-Fei, "Adaptive procedural task generation for hard-exploration problems," *arXiv preprint arXiv:2007.00350*, 2020.

[31] ——, "Discovering generalizable skills via automated generation of diverse tasks," *arXiv preprint arXiv:2106.13935*, 2021.

[32] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel, "Evolving curricula with regret-based environment design," *arXiv preprint arXiv:2203.01302*, 2022.

[33] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments," in *Conf. on Robot Learning*, 2020, pp. 835–853.

[34] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving Rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[35] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *Proc. of the IEEE Int. Conf. on Robotics and Automation)*, 2020, pp. 6232–6238.

[36] J. Lundell, F. Verdoja, and V. Kyrki, "DDGC: Generative deep dexterous grasping in clutter," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6899–6906, 2021.

[37] S. Gomez-Gonzalez, S. Prokudin, B. Schölkopf, and J. Peters, "Real time trajectory prediction using deep conditional generative models," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 970–976, 2020.

[38] J. Bütepage, A. Ghadirzadeh, Ö. Öztimur Karadağ, M. Björkman, and D. Kragic, "Imitating by generating: Deep generative models for imitation of interactive tasks," *Frontiers in Robotics and AI*, vol. 7, p. 47, 2020.

[39] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2020.

[40] S. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, *et al.*, "Attend, infer, repeat: Fast scene understanding with generative models," *Advances in Neural Inf. Processing Systems*, vol. 29, 2016.

[41] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in Neural Inf. Processing Systems*, vol. 34, pp. 8780–8794, 2021.

[42] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Inf. Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[43] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Int. Conf. on machine learning*, 2015, pp. 1530–1538.

[44] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *J. Mach. Learn. Res.*, vol. 22, no. 57, pp. 1–64, 2021.

[45] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in Neural Inf. Processing Systems*, vol. 29, 2016.

[46] A. Mishra, S. Krishna Reddy, A. Mittal, and H. A. Murthy, "A generative model for zero shot learning using conditional variational autoencoders," in *Proc. of the IEEE Conf. on computer vision and pattern recognition workshops*, 2018, pp. 2188–2196.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Inf. Processing Systems*, vol. 27, 2014.

[48] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[49] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," *Advances in Neural Inf. Processing Systems*, vol. 31, 2018.

[50] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Inf. Processing Systems*, vol. 32, 2019.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[54] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," Pybullet, Tech. Rep., 2016.

[55] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[56] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *AAAI*, vol. 93, 1993, pp. 99–105.

[57] T. Salimans and R. Chen, "Learning Montezuma's revenge from a single demonstration," *arXiv preprint arXiv:1812.03381*, 2018.

[58] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.