URDFormer: Constructing interactive Realistic Scenes from Real Images via Simulation and Generative Modeling

Anonymous Author(s)
Affiliation
Address
email

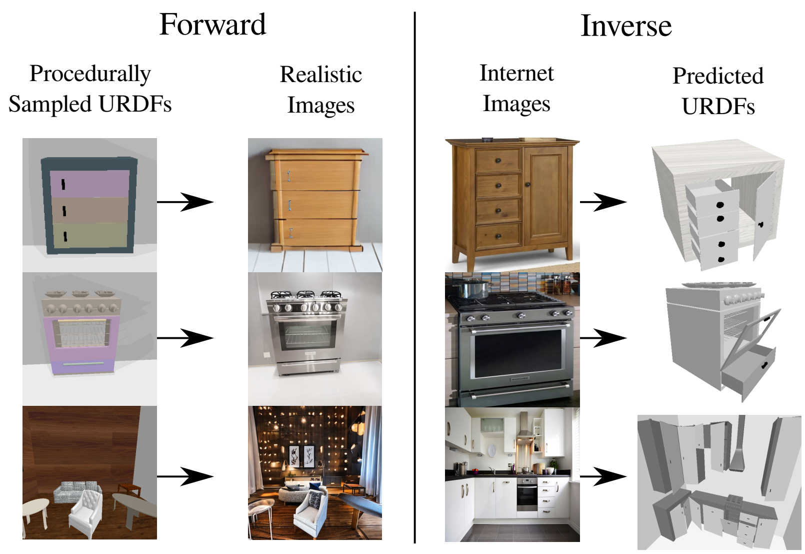

Figure 1: Our method uses generative models in a "Forward" process to produce structurally consistent realistic images from procedurally generated simulation content. We then use these generated simulation/image pairs to train an "Inverse" process that is able to estimate the underlying structure of diverse real-world images.

Abstract: Constructing accurate and targeted simulation scenes that are both visually and physically realistic is a significant practical interest in domains ranging from robotics to computer vision. However, this process is typically done largely by hand - a graphic designer and a simulation engineer work together with predefined assets to construct rich scenes with realistic dynamic and kinematic properties. While this may scale to small numbers of scenes, to achieve the generalization properties that are requisite of data-driven machine learning algorithms, we require a pipeline that is able to synthesize large numbers of realistic scenes, complete with "natural" kinematic and dynamic structure. To do so, we develop models for inferring structure and generating simulation scenes from natural images, allowing for scalable scene generation from web-scale datasets. To train these image-to-simulation models, we show how effective generative models can be used in generating training data, the network can be *inverted* to map from realistic images back to complete scene models. We show how this paradigm allows us to build large datasets of scenes with semantic and physical realism, enabling a variety of downstream applications in robotics and computer vision. More visualizations are available at: https://sites.google.com/view/urdformer/home

Keywords: Generative Modeling, Scene Generation, Simulation

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

1 Introduction

Simulation offers the dual advantages of scalable and cheap data collection and an easy way to encode domain-specific prior knowledge into end-to-end machine-learning problems [1, 2, 3]. This is particularly important for data-scarce problems such as robotics, where collecting real data can lead to costly and unsafe failures or may require expensive human supervision. Critical to each of these endeavors is a rich and accurate simulation environment, complete with complex scene layouts and kinematic structure. The de-facto process for generating simulation content is either manual [4] or procedural [5]. The manual process for creating simulation scenes involves the algorithm designer working to characterize, identify, and model a particular real-world scene, a painstaking and impractical process. This leads to content that is not very diverse due to the onerous human effort required. On the other hand, rule-based procedural generation methods [5, 6] have seen success in particular machine learning applications such as embodied navigation, but often struggle to capture the natural complexity of the real world. Moreover, the procedural generation process is not controllable, making it hard to generate simulation content corresponding to a *particular* real-world environment. The inability of the current status quo in the generation of simulation content - both procedural generation and manual creation, makes apparent the necessity of a targeted technique for scalable content creation in simulation that is able to retain realistic kinematic and semantic structure.

To enable a variety of downstream use cases, scalable content creation in simulation must be (1) realistic enough such that inferences made in simulation transfer back to the real world (2) diverse in a way that captures natural statistics so as to enable learning generalizable models and policies (3) controllable in a way that allows for targeted generation of particular scenes of interest. While a variety of methods for scene generation and inverse graphics [7, 8, 9] satisfy one or more of these criteria, to the best of our knowledge, it has proven challenging to develop content creation methods that satisfy them all. We develop methods that map directly from isolated real-world images to corresponding simulation content (expressed as a Unified Robot Description File (URDF)) that could plausibly represent the semantics, kinematics, and structure of the scene. This is an inverse mapping problem going from real-world images to kinematically accurate, interactive simulation. While inverse modeling problems in the literature have been tackled with data-driven techniques such as supervised learning, in this case, a large-scale paired dataset of realistic images and their corresponding simulation environments does not readily exist in the literature.

Our key idea is that we can generate a suitable dataset for inverse modeling from images to plausible simulations by leveraging controllable text-to-image generative models [10]. From a set of procedurally or manually constructed scenes, we can generate realistic images that are representative of that particular simulation scene. This paired dataset of simulation scenes and corresponding realistic images can then be *inverted* via supervised learning to learn a model that maps from realistic images directly to plausible simulation environments. This learned model can generate realistic and diverse content directly from real-world images mined from the web without any additional annotation. The resulting models can be used in several use cases - (1) diverse generation: generating a large and diverse set of realistic simulation environments that correspond directly to real-world images, or (2) targeted generation: generating a simulation environment (or narrow distribution of environments) corresponding to a particular set of desired images.

2 Related Work

Inverse-Graphics: A variety of work in inverse-graphics focuses on inferring scene properties such as geometry, lighting, and other geometric properties from single images [11]. This work has both been optimization-based [12] and learning-based[13]. In a similar vein, a rich body of work [14] focuses on mesh reconstruction and novel view synthesis using a variety of techniques such as implicit neural fields [15, 16, 17], Gaussian splatting [18, 19], differentiable rendering [20, 21, 22] amongst many other techniques. Importantly, the focus of many of these works on inverse graphics has been on geometric reconstruction rather than our focus on scene-level simulation construction complete with kinematic and semantic structure like object relationships and articulation. There have been a number of efforts in inferring physical properties such as articulation [23, 24, 25], friction and surface properties [26, 27, 28, 29], although these typically require either interaction or video access. In contrast, our work focuses less on exact geometry reconstruction but rather on generating correct scene statistics at the articulation/kinematics/positioning level for entire scenes or complex objects from single RGB images. Our goal is a fast generation process that can scale to generate hundreds of scenes with natural statistics, without requiring interaction or targeted data collection per domain.

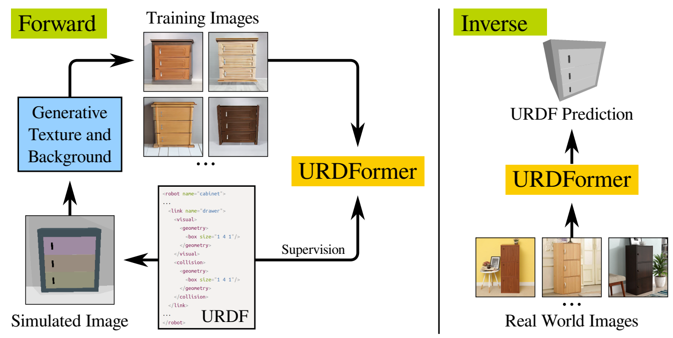

Figure 2: An overview of the training and application of URDFormer . During the forward process, existing simulation assets are first used to generate a large paired dataset of simulation assets and realistic rendered images. This paired dataset is used to train the URDFormer inverse model that can predict URDFs from RGB images. This model can then be used with real-world images to generate novel simulations.

Generating indoor scenes is a long-standing problem in computer vision and machine learning. This has been approached by building learned generative models of indoor scenes [30, 31, 32, 33] and floorplans [34, 35, 36], while others have produced text-to-scene models [37, 38]. While generating scenes this way can be promising, these methods either fail to achieve the targeted generation of complex scenes with articulation and complex kinematic structure intact or require extremely expensive inference processes to do so. On the other hand, procedural generation techniques have been popular in generating grid-world environments [39, 40, 41, 42] and in generating home environments at scale [5]. These scenes are diverse but are not controllable to particular target scenes or complete with physical properties and articulation. Other techniques such as [43, 44] are able to generate large datasets of interactive scenes but require interactive scanning with either a phone or other hardware for dataset generation specific to indoor scenes. URDFormer is able to generate realistic, diverse, and controllable scenes while retaining rich kinematic and semantic structure from internet images alone.

Data Augmentation with Generative Models Our work is certainly not the first [45] to use synthetic data generated by generative models for training networks that can then be deployed on real data. These models have been used the context of data augmentation [46, 47, 48], representation learning via self supervised learning [49, 50, 51], model selection [52] and even applications like healthcare [53]. In contrast, our work shows that controllable generative modeling can be used to generate datasets that are suitable for inverse modeling for creating simulation assets at scale.

3 URDFormer : Generating Interactive Simulation Environments by Learning Inverse Models from Generated Datasets

3.1 Controlled Generation of Paired Datasets with Generative Models

Given a simulated scene z (drawn from a dataset such as [54], or procedurally generated), we use the fact that controllable generative models are both diverse and realistic enough to take an unrealistic rendering of a scene in simulation and generate a distribution of corresponding *realistic* images. This allows the scene in simulation with unrealistic appearance and texture to be translated into a diverse set of visually realistic images that plausibly match the same underlying environment. To ensure piecewise consistency and realism of the generated images, we use two different dataset generation techniques for the scene structure and object structure respectively. These share the same conceptual ideas but differ to account for consistency properties in each case.

Scene-Level Dataset Generation: To generate training data for the scene model, we feed the rendered image from simulation along with a templated text prompt to an image-and-text guided diffusion model [10]. This model generates a new image that attempts to simultaneously match the content described in the text prompt while retaining the global scene layout from the provided image. We found that this model is able to reliably maintain the scene layout, but it may change some individual components of the scene, for example replacing objects with a different but plausible category, or changing the number of components under an object such as the drawers or handles. Despite these failures, the large-scale structural consistency still provides a useful source of training data. After running our simulated image through the generated model, we have realistic images that contain known high-level object positions and spatial relationships, but unknown category and low-level part structures. This means that the scene model dataset contains complete images, but incomplete labels. Rather than complete (x, z) pairs, we have a dataset $\mathcal{D}_{\text{scene}} = \{(x, \tilde{z})\}$ of $(x, \tilde{z})$ pairs where $\tilde{z}$ only contains the bounding boxes, transforms and parents of the high-level (non-part) objects $\tilde{z} = \{(b_1, T_1, p_1) \dots (b_n, T_n, p_n)\}$.

Object-Level Dataset Generation: The process for generating object-level training data is similar, but requires more care due to the tendency of generative models to modify low-level details. For objects with complex kinematic structure, such as cabinets, we procedurally generate a large number of examples of these objects and render them in isolation from different angles. Rather than using the generative model to construct entirely new images, we use it to produce diverse texture images, which are overlaid in the appropriate locations on the image using perspective warping. We then change the background of the image using the generative model with appropriate masking derived from the original render. For less complex objects that do not have important part-wise structure, we simply replace the rendered image with a new sample from the image-and-text guided generative model. Unlike the scene dataset which contains complete images but partial labels, the object dataset contains partial images in the sense that they contain only a single object, but complete labels for the object and its kinematic parts. We can say that this dataset $\mathcal{D}_{\text{object}}$ contains $(\tilde{x}, z)$ pairs where $\tilde{x}$ is an image of a single object rather than a full scene (hence the partial x), and z is complete for the single object and its parts. The result of these two data generation processes is a high-level scene structure dataset $\mathcal{D}_{\text{scene}}$, and a low-level object dataset $\mathcal{D}_{\text{object}}$.

3.2 URDFormer : Learning Inverse Generative Models for Scene Synthesis

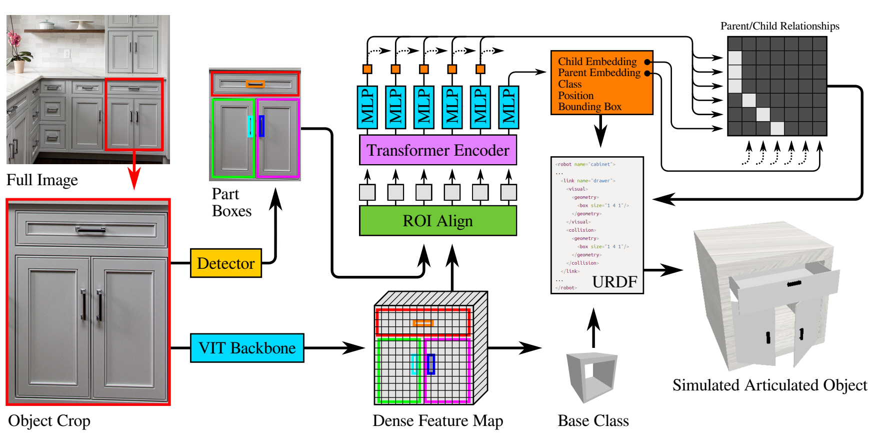

Figure 3: Architecture of URDFormer : an inverse model (URDFormer) that predicts simulation parameters from RGB images. URDFormer can translate web-scraped real-world RGB images of scenes into complete simulation assets. The model shown here is used to estimate the part structure of an individual object. When estimating the scene structure, the Object Crop image would be replaced by an image of the entire scene.

Given the datasets $\mathcal{D}_{\text{object}} = (\tilde{x}, z)$ and $\mathcal{D}_{\text{scene}} = (x, \tilde{z})$ constructed as described above, we can use supervised learning methods to learn an *inverse model* that maps images of a complex object or scene to the corresponding simulation asset. In order to take advantage of these partially complete datasets, we must add some structure to our prediction model. We do this by splitting our learned inverse model in correspondence with the split in our forward model: we train one network f_θ^{-1} to predict the high-level scene structure using dataset $\mathcal{D}_{\text{scene}}$ and another network g_ϕ^{-1} to predict the low-level part structure of objects using $\mathcal{D}_{\text{object}}$.

To model both the scene-level prediction model (f_θ^{-1}) and the low-level part prediction model (g_ϕ^{-1}), we propose a novel network architecture - URDFormer, that takes an RGB image and predicts URDF primitives as shown in Figure 3. Note that both the scene-level prediction and the low-level part prediction use the same network architecture, the scene-level simply operates on full images with object components segmented, while the part-level operates on crops of particular objects with parts segmented. In the URDFormer architecture, the image is first fed into a ViT visual backbone[55] to extract global features. We then obtain bounding boxes of the objects in the image using the masks rendered from the original procedurally generated scene in simulation (these are known at training time, and can be easily extracted using segmentation models at test time). We then use ROI alignment [56] to extract features for each of these bounding boxes. These feature maps are combined with an embedding of the bounding box coordinates and then fed through a transformer [57] to produce a feature for each object in the scene. An MLP then decodes these features into an optional class label (used only when training the object-level model), and a discretized 3D position and bounding box. In addition, it also produces a child embedding and a parent embedding that are used to predict the hierarchical relationships in the scene (object to its parent and so on). To construct these relationships, the network uses a technique from scene graph generation [58] that produces an $n \times n$ relationship score matrix by computing the dot product of every possible parent with every possible child. The scene model also has learned embeddings for six different root objects corresponding to the four walls, the floor, and the ceiling so that large objects like countertops and sinks can be attached to the room.

Due to the unpredictable nature of the generative transforms that are used to make the scene image realistic, which may change class identities, only the position, bounding box, and relationship information are used when computing the high-level scene structure. To compute the class labels for the top-level objects, we use max-pooling of the dense ViT features along with an MLP in the part-prediction model g_ϕ^{-1}. To generate a full estimate of the scene description from a natural image at test time, the image and a list of high-level bounding boxes are first fed to the scene prediction model f_θ^{-1}, which predicts the location and parent for each object. The image regions corresponding to these boxes are then extracted and further segmented to produce part-level bounding boxes. Each of these image regions and the corresponding part boxes are then fed into the part prediction model to compute the kinematic structure of the low-level parts. This nested prediction structure can be used to generate entire scenes from web-scraped RGB images drawn from any image dataset to generate novel simulation content both at the scene level and at the object level.

4 Experiments

4.1 Phase 1: (Forward) Paired Dataset Generation

To synthesize the initial paired dataset, we first procedurally generate a set of URDF representations of scenes in simulation both for global scenes like kitchens and for single objects like fridges, cabinets, and drawers. These initially generated simulation scenes are shown in Fig4 (Left). We can then follow the procedure outlined in Section 3.1 for the controlled generation of paired datasets to generate a large dataset of simulation scenes and paired realistic RGB images as shown in Fig4 (Right) (More visualizations and videos are available on the website). For objects with diverse parts, we observe that depth-guided stable diffusion [10] often ignores the semantic details of local parts, leading to inconsistency issues shown as Fig 7 in Appendix A.1. To overcome this issue, we use images of texture to guide diffusion models to generate large and diverse texture templates and randomly choose one template and warp it back to the original part region using perspective transformation. We apply in-painting models for smoothing the boundary of the parts and generating content for the background. We visualize this process in A.1 Fig 6. In total, we generated 260K

Figure 4: Qualitative results on (forward) paired dataset generation. Left: Original simulation images. Right: Generated realistic images that match the URDF descriptions of the scene on the left.

images for global scenes of kitchens and living rooms, and 235K images of 14 types of objects such as cabinets, ovens, and fridges. Details of the dataset can be found in Appendix B.1.

4.2 Phase 2: (Inverse) Real-World URDF Prediction

Given the generated paired dataset shown in Fig 4, we next evaluate how successful a trained inverse model is at generating simulation scenes representing unseen real-world test images.

Real World Dataset: We create two types of datasets for evaluation: (a) Obj300 includes URDFs of 300 internet images of individual objects from 5 categories including 100 cabinets, 50 ovens, 50 dishwashers, 50 fridges and 50 washing machines. (b) Global scenes include URDFs of 80 internet images including 54 kitchens and 26 living rooms. For each scene, we manually label the bounding box for each object and its parts, as well as the URDF primitives including mesh types, parent bounding box ID, positions, and scales relative to its parent. We use the mesh types such as ”left door”, and ”right door” to infer link axis and joint types. All the position values and scale values are discretized into 12 bins.

Evaluation Metrics: Evaluating entire scenes is challenging given the mixed structure and subjective nature of human labelling. We adopt an edit-distance based metric for structural comparison, and use a small dataset of manually labelled examples for evaluation.

(1) Edit Distance with Bounding Box Offset: We evaluate our predicted scene structure using a tree edit-distance metric. This method requires access to a predicted and ground-truth kinematic tree. We start at the root of the kinematic tree and use the Hungarian method to compute the lowest-cost assignment between the children of the predicted root and the children of the ground truth root where the cost is based on their spatial coordinates. If there are more predicted children than ground truth, the unassigned predicted children and all of their descendants are marked as **False Positive** edits. Conversely, if there are more ground truth children than predicted children, the unmatched ground truth children and all of their descendants are marked as **False Negative** edits. We then compare the spatial parameters of the matched predicted and ground truth children. If they are not close enough

Table 1: Comparison with baseline methods: trained with random colors, selected textures, and random textures, as well as prompt guided BLIP2. URDFormer with generated realistic textures predicts more accurate simulation content from unseen real-world images.

	Obj300 (↓)	Global (Obj) (↓)	Global (Parts) (↓)
URDFormer (Random Colors)	1.08	10.81	19.62
URDFormer (Selected Textures)	0.63	9.87	19.11
URDFormer (Random Textures)	1.22	11.85	18.67
Guided BLIP2	4.27	14.64	24.58
URDFormer (Generated Textures (ours))	**0.42**	**9.51**	**18.21**

to each other according to a fixed threshold, the predicted child and its descendants are marked as **False Positives**, and the ground truth child and its descendants are marked as **False Negatives**. If the two are close enough, the class label of the predicted child is compared against the class label of the ground truth child. If they do not match, we add a **Class Incorrect** edit. Regardless of whether the classes match, this process is recursively applied to the matching children. To compute a single score, we assign weights to these edits based on their position in the hierarchy and sum them. For the experiments in this paper, we assigned a weight of 1.0 to all edits at the top level corresponding to objects, a weight of 0.5 to the parts such as cabinet doors, and a weight of 0.1 to all objects further down the hierarchy such as handles and knobs attached to doors.

(2) Edit Distance with IoU: Similar to bounding box offset, we simply replace the spatial coordinate cost with IoU between two bounding boxes. We define levels of threshold based on overlapping areas: ED $\mathrm{IoU}_{0.25}$, ED $\mathrm{IoU}_{0.5}$, ED $\mathrm{IoU}_{0.75}$. We show evaluation using both metrics in ablations, but in general, we found the two metrics yield the same performance, thus we only use edit distance with a bounding box for baseline evaluation.

Baselines We compare URDFormer against several other baselines in Table 1. In particular, to show the importance of pixel realism, we compare with training on (1) Random Colors (2) Selected Realistic Textures (3) Random Textures (Visualizations of their differences are in Appendix A.3). In addition, we also compare our method against recent Vision-Language Models with guided prompts: Guided BLIP2. In particular, (1) Random Colors randomly selects RGB values for each part inside the scene and (2) Selected Realistic Textures manually selects texture images for corresponding objects. (3) Random Textures selects random images. (4) Guided BLIP2 takes a sequence of question prompts and guides pretrained BLIP2 models [59] to output the URDF primitives in the valid format (Please check Appendix C.3 for prompt details). We observe that training with generated realistic visual features improves the generalization to real-world images. Although trained on large real-world datasets, BLIP2 fails to reason about the 3D structure of the scene as well as the kinematics structure of individual objects, showing using a more structured and targeted dataset is important during training. Here Global (Obj) represents the evaluation of high-level position/parent reasoning, while Global (Parts) represents the evaluation of the full scene including the high-level and detailed kinematic structure of each object.

Ablations To study how different components of URDFormer impact the performance, we perform an ablation study on (1) Do backbones pretrained on real-world images help with generalization? (2) What are the important features of learning 3D kinematic structures, as shown in Table 2. In particular, we train URDFormer with three types of backbones: (1) vit-small-patch16-224 trained from scratch (2) finetune vit-small-patch16-224 pretrained on ImageNet (3) finetune vit-small-patch16-224 trained in [60] on 197K kitchen scenes and evaluate on 54 real-world kitchen images. We observe that finetuning the vision backbone that is pretrained on real images performs better than training from scratch, and pretrained in [60] achieves the best performance, which is likely due to the fact that it was trained on more diverse datasets than ImageNet. We observe that both training with only image features and training with only bounding box features decrease the performance, indicating the importance of both spatial and visual features.

Qualitative Results: We show the qualitative results of our URDF predictions in Fig 5. We use the same color to represent the same mesh type for better visualization. We observe that training

Table 2: Ablation study on training with different vision backbones and input features, showing training using both visual/spatial features, with a backbone pretrained on diverse real images achieves higher performance.

	ED Box (↓)	ED IoU$_{0.25}$ (↓)	ED IoU$_{0.5}$ (↓)	ED IoU$_{0.75}$ (↓)
Scratch	7.00	6.15	8.37	14.48
Pretrained on ImageNet	6.33	5.48	7.74	13.85
Pretrained MAE	**5.70**	**5.11**	**7.07**	**13.41**
Pretrained MAE (No bbox)	6.19	5.26	7.63	14.11
only with bbox	7.04	6.52	8.26	14.26

with data generated using the method described in section 3.1 provides diverse visual information compared to baseline methods such as random colors or random textures. This is important for distinguishing mesh types such as stove and fridge, and reasoning about structure relations such as "cabinet on the right" and "cabinet in the front".

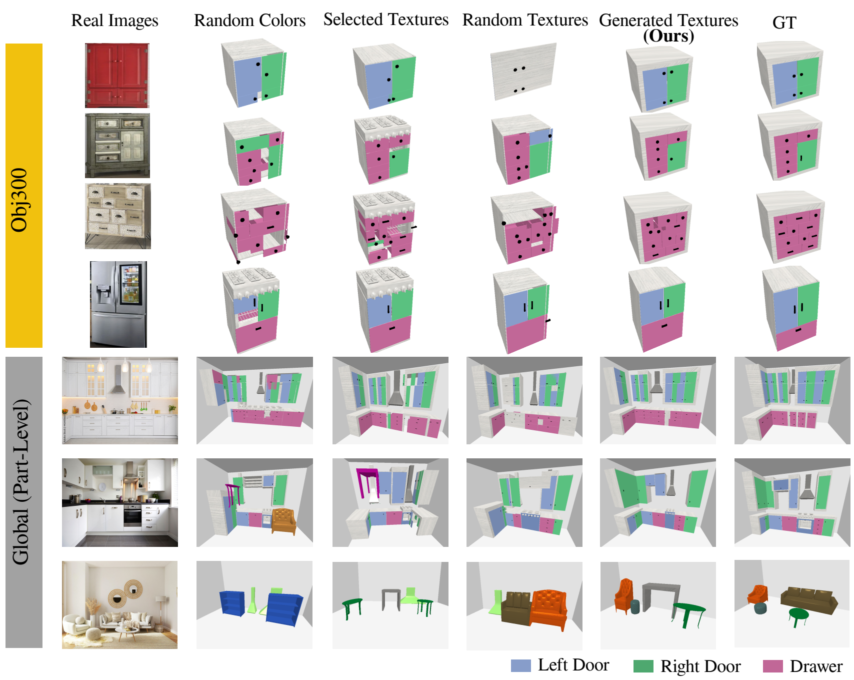

Figure 5: Evaluations of generated simulations on unseen real-world images. The left-most column indicates the real-world image input and each column indicates the performance of an inverse URDF prediction model trained with different training sets. We evaluate training datasets generated using random colors, selected textures, random textures, and textures generated with pre-trained generative models (ours), and compare these with ground truth URDF labels.

5 Conclusion

In this work, we presented URDFormer , a general-purpose, scalable technique for generating simulation content from real-world RGB images. We first generate a large-scale paired dataset of procedurally generated simulation content and a corresponding realistic RGB image using pre-trained controllable generative models. We then use our generated paired dataset to train an inverse model that maps directly from single RGB images to corresponding representations of scenes or complex objects in simulation. This inverse model can then be used with large image datasets of real-world RGB images to scalably generate simulation data complete with kinematic and semantic structure, without requiring any hand-crafting or hand-designing of these simulation assets. We show in our experimental results the efficacy of this scheme in generating assets at scale from real-world datasets of RGB images.

References

[1] J. Collins, S. Chand, A. Vanderkop, and D. Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021. doi:10.1109/ACCESS.2021.3068769. URL `https://doi.org/10.1109/ACCESS.2021.3068769`.

[2] Y. S. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, Á. Moravánszky, G. State, M. Lu, A. Handa, and D. Fox. Factory: Fast contact for robotic assembly. In K. Hauser, D. A. Shell, and S. Huang, editors, *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, 2022. doi:10.15607/RSS.2022.XVIII.035. URL `https://doi.org/10.15607/RSS.2022.XVIII.035`.

[3] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *Int. J. Comput. Vis.*, 126(9):902–919, 2018. doi:10.1007/s11263-018-1073-7. URL `https://doi.org/10.1007/s11263-018-1073-7`.

[4] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[5] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

[6] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, A. Newell, H. Law, A. Goyal, K. Yang, and J. Deng. Infinite photorealistic worlds using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 12630–12641. IEEE, 2023. doi:10.1109/CVPR52729.2023.01215. URL `https://doi.org/10.1109/CVPR52729.2023.01215`.

[7] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015.

[8] S. Lunz, Y. Li, A. W. Fitzgibbon, and N. Kushman. Inverse graphics GAN: learning to generate 3d shapes from unstructured 2d data. *CoRR*, abs/2002.12674, 2020. URL `https://arxiv.org/abs/2002.12674`.

[9] M. Jaques, M. Burke, and T. M. Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=BJeKwTNFvB`.

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[11] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

[12] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[13] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[14] T. Samavati and M. Soryani. Deep learning-based 3d reconstruction: a survey. *Artif. Intell. Rev.*, 56(9):9175–9219, 2023. doi:10.1007/s10462-023-10399-2. URL https://doi.org/10.1007/s10462-023-10399-2.

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[16] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. Goldman, S. Seitz, and R. Martin-Brualla. Deformable neural radiance fields. *https://arxiv.org/abs/2011.12948*, 2020.

[17] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. NERF++: Analyzing and improving neural radiance fields. *https://arxiv.org/abs/2010.07492*, 2020.

[18] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[19] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[20] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. *Advances in neural information processing systems*, 31, 2018.

[21] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[22] H.-T. D. Liu, M. Tao, and A. Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6):221–1, 2018.

[23] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[24] Z. Xu, Z. He, and S. Song. Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 7(2):2447–2454, 2022.

[25] B. DeMoss, P. Duckworth, N. Hawes, and I. Posner. Ditto: Offline imitation learning with world models. *arXiv preprint arXiv:2302.03086*, 2023.

[26] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. *Advances in Neural Information Processing Systems*, 30, 2017.

[27] L. S. Piloto, A. Weinstein, P. Battaglia, and M. Botvinick. Intuitive physics learning in a deep-learning model inspired by developmental psychology. *Nature human behaviour*, 6(9): 1257–1267, 2022.

[28] J. R. Kubricht, K. J. Holyoak, and H. Lu. Intuitive physics: Current research and controversies. *Trends in cognitive sciences*, 21(10):749–759, 2017.

[29] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

[30] D. Ritchie, K. Wang, and Y.-a. Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6182–6190, 2019.

[31] M. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019.

[32] M. Keshavarzi, A. Parikh, X. Zhai, M. Mao, L. Caldas, and A. Y. Yang. Scenegen: Generative contextual scene augmentation using scene graph priors. *arXiv preprint arXiv:2009.12395*, 2020.

[33] D. A. Hudson and C. L. Zitnick. Generative adversarial transformers. *CoRR*, abs/2103.01209, 2021. URL https://arxiv.org/abs/2103.01209.

[34] R. Hu, Z. Huang, Y. Tang, O. Van Kaick, H. Zhang, and H. Huang. Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, 39(4):118–1, 2020.

[35] N. Nauata, S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, and Y. Furukawa. House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13632–13641, 2021.

[36] K. Wang, X. Xu, L. Lei, S. Ling, N. Lindsay, A. X. Chang, M. Savva, and D. Ritchie. Roominoes: Generating novel 3d floor plans from existing 3d rooms. *Comput. Graph. Forum*, 40(5): 57–69, 2021. doi:10.1111/cgf.14357. URL https://doi.org/10.1111/cgf.14357.

[37] A. Chang, M. Savva, and C. D. Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 2028–2038, 2014.

[38] A. Chang, W. Monroe, M. Savva, C. Potts, and C. D. Manning. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*, 2015.

[39] 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*, volume 16, pages 95–101, 2020.

[40] S. Earle, M. Edwards, A. Khalifa, P. Bontrager, and J. Togelius. Learning controllable content generators. In *2021 IEEE Conference on Games (CoG)*, pages 1–9. IEEE, 2021.

[41] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[42] L. Gisslén, A. Eakins, C. Gordillo, J. Bergdahl, and K. Tollmar. Adversarial reinforcement learning for procedural content generation. In *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*, pages 1–8. IEEE, 2021. doi:10.1109/CoG52621.2021.9619053. URL https://doi.org/10.1109/CoG52621.2021.9619053.

[43] Z. Li, T. Yu, S. Sang, S. Wang, M. Song, Y. Liu, Y. Yeh, R. Zhu, N. B. Gundavarapu, J. Shi, S. Bi, H. Yu, Z. Xu, K. Sunkavalli, M. Hasan, R. Ramamoorthi, and M. Chandraker. Openrooms: An open framework for photorealistic indoor scene datasets. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 7190–7199. Computer Vision Foundation / IEEE, 2021. doi:10.1109/CVPR46437.2021.00711. URL https://openaccess.thecvf.com/content/CVPR2021/html/Li_OpenRooms_An_Open_Framework_for_Photorealistic_Indoor_Scene_Datasets_CVPR_2021_paper.html.

[44] M. Deitke, R. Hendrix, A. Farhadi, K. Ehsani, and A. Kembhavi. Phone2proc: Bringing robust robots into our chaotic world. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 9665–9675. IEEE, 2023. doi:10.1109/CVPR52729.2023.00932. URL https://doi.org/10.1109/CVPR52729.2023.00932.

[45] P. Eigenschink, T. Reutterer, S. Vamosi, R. Vamosi, C. Sun, and K. Kalcher. Deep generative models for synthetic data: A survey. *IEEE Access*, 11:47304–47320, 2023. doi:10.1109/ACCESS.2023.3275134. URL https://doi.org/10.1109/ACCESS.2023.3275134.

[46] Z. Q. Chen, S. C. Kiami, A. Gupta, and V. Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.010. URL https://doi.org/10.15607/RSS.2023.XIX.010.

[47] T. Yu, T. Xiao, J. Tompson, A. Stone, S. Wang, A. Brohan, J. Singh, C. Tan, D. M, J. Peralta, K. Hausman, B. Ichter, and F. Xia. Scaling robot learning with semantically imagined experience. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.027. URL https://doi.org/10.15607/RSS.2023.XIX.027.

[48] B. Trabucco, K. Doherty, M. Gurinas, and R. Salakhutdinov. Effective data augmentation with diffusion models. *CoRR*, abs/2302.07944, 2023. doi:10.48550/arXiv.2302.07944. URL https://doi.org/10.48550/arXiv.2302.07944.

[49] S. Fu, N. Tamir, S. Sundaram, L. Chai, R. Zhang, T. Dekel, and P. Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *CoRR*, abs/2306.09344, 2023. doi:10.48550/arXiv.2306.09344. URL https://doi.org/10.48550/arXiv.2306.09344.

[50] Y. Tian, L. Fan, P. Isola, H. Chang, and D. Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. *CoRR*, abs/2306.00984, 2023. doi:10.48550/arXiv.2306.00984. URL https://doi.org/10.48550/arXiv.2306.00984.

[51] A. Jahanian, X. Puig, Y. Tian, and P. Isola. Generative models as a data source for multiview representation learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=qhAeZjs7dCL.

[52] A. Shoshan, N. Bhonker, I. Kviatkovsky, M. Fintz, and G. G. Medioni. Synthetic data for model selection. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31633–31656. PMLR, 2023. URL https://proceedings.mlr.press/v202/shoshan23a.html.

[53] E. Choi, S. Biswal, B. A. Malin, J. Duke, W. F. Stewart, and J. Sun. Generating multi-label discrete patient records using generative adversarial networks. In F. Doshi-Velez, J. Fackler, D. C. Kale, R. Ranganath, B. C. Wallace, and J. Wiens, editors, *Proceedings of the Machine Learning for Health Care Conference, MLHC 2017, Boston, Massachusetts, USA, 18-19 August 2017*, volume 68 of *Proceedings of Machine Learning Research*, pages 286–305. PMLR, 2017. URL http://proceedings.mlr.press/v68/choi17a.html.

[54] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 909–918. Computer Vision Foundation / IEEE, 2019. doi:10.1109/CVPR.2019.00100. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Mo_PartNet_A_Large-Scale_Benchmark_for_Fine-Grained_and_Hierarchical_Part-Level_3D_CVPR_2019_paper.html.

[55] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[56] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] J. Yang, W. Peng, X. Li, Z. Guo, L. Chen, B. Li, Z. Ma, K. Zhou, W. Zhang, C. C. Loy, and Z. Liu. Panoptic video scene graph generation. In *CVPR*, 2023.

[59] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[60] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

Appendix

A (Forward) Data Generation

A.1 Part-Consistency

We compare our part-wise generation method with other approaches qualitatively in Fig 7. In particular, we observe that depth-guided or in-painting stable diffusion models [10] often ignore local consistency, making it difficult to render high-quality images that are paired with the simulation content. Instead, we only use stable diffusion to change the style of texture and warp the original texture to each part of the object, as shown in Fig6 and the masks of each part can be obtained directly from the simulator.

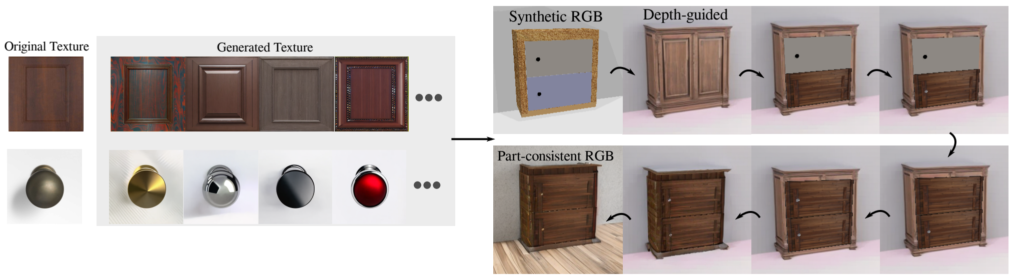

Figure 6: Paired dataset generation using texture and prompt templates to guide Stable Diffusion [10] and create a diverse texture dataset, which can be then warped on the targeted individual part of the object, as described in Section 3.1

Figure 7: Qualitative comparison among different rendering methods: depth-guided diffusion models, inpainting stable diffusion and part-wise generation

A.2 Dataset Generation

Qualitative details of our dataset of articulated, rigid objects as well as the full scenes are shown in Fig8, the top row in each section represents the original synthetic images from the simulation, the bottom images are the generated pair images that match the original kinematic structures.

A.3 Baseline Data

We visualize the different training data for baseline methods shown in Table 1: URDFormer with random colors, selected textures, random textures, and generated textures. All baseline inputs are

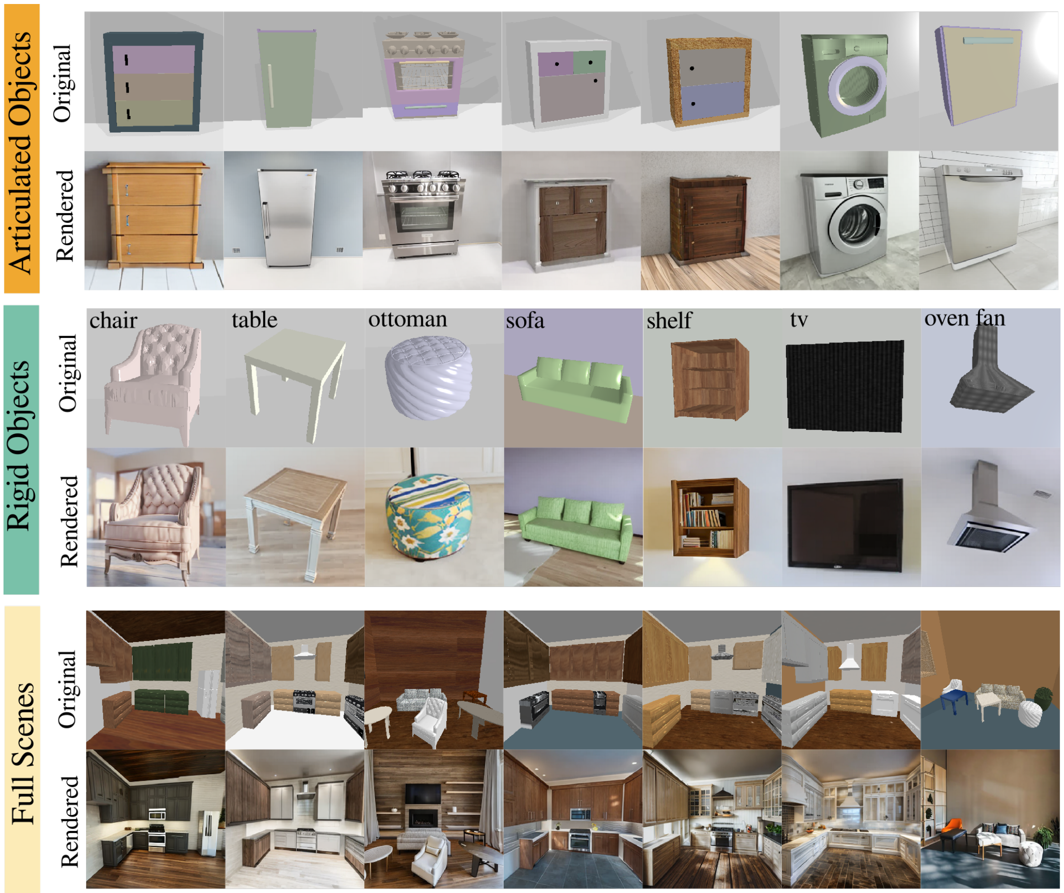

Figure 8: Data Generation for articulated objects, rigid objects, and full scenes. The top row in each section represents the original synthetic images from the simulation, the bottom images are the generated pair images that match the original kinematic structures.

captured from the same camera angles. As shown in Fig 9, the generated texture shows high pixel realism that is closer to the distribution of the real world. As shown in Table 1, training on such data improves performance in predicting URDF structures from real-world images during the test time.

B Training Details of URDFormer

B.1 Dataset

Our training dataset includes 267K global scene labels (197K kitchen scenes and 70K living room scenes) and 235K objects, which include 14 types of objects including cabinet, oven, dishwasher, washer, fridge, oven fan, shelf, tv, sofa, chair, square table, ottoman, coffee table and stuffed toy. Among these objects, 5 categories are articulated: cabinet, oven, dishwasher, washer, and fridge. These articulated objects include part meshes in from 8 types: drawer, left door, right door, oven door, down door, circle door, handle and knob.

B.2 Training Details

We use a pretrained vit-small-patch16-224 trained in [60] as the vision backbone, which outputs the global image features dimensions of 14x14x384. To predict the base type, the global features are first max-pooled followed by a MLP to predict a class type over 14 object types. We then perform ROI alignment on cropped features with bounding boxes of the objects or parts. In ROI Alignment, we set the spatial scale=1 / 16 and the sampling ratio=2. The ROI size is set to 14.

Figure 9: Comparison among baseline methods with different training input: Random Colors, selected textures, random textures and generated textures. Generated textures shows photo-realism that closer to the real-world distribution.

The roi-aligned features are then fed into a 3-layer MLP followed by a norm layer. To compute the positional encoder, we feed the bounding box coordinates into a 3-layer MLP as well as a norm layer. These normalized roi features together with the normalized spatial features are summed as the token features and feed into the transformer, which are then fed into MLPs to compute URDF primitives: position start (relative to parent), position end (relative to parent), mesh type, and parent-child relation matrix. Here instead of regressing to a position value, we treat it as a classification problem, where we discretize the x,y, and z axis of the parent mesh into 12 bins. During training, the maximum sequence length is set to 32, which means the maximum number of bounding boxes per image is 32. All baseline methods (URDFOrmer with random colors, selected textures, and random textures) are trained on one A40 GPU with a batch size of 256. All baselines are trained with an equal number of epochs and evaluated using the last checkpoint.

C Experiment Details

C.1 Details for Dataset Assets

We procedurally generate scenes using both rigid and articulated objects. In particular, we collected 9 categories of common rigid objects in the kitchen and living room and 5 categories of common articulated objects for kitchens, and randomly rescale them during data generation.

C.2 Prompts for Data Generation

Textures:

(1) material prompt: 'bright', 'colorful', 'modern', 'multicolor', 'fancy color', 'accent', 'glass', 'chestnut', 'Oakwood', 'Maplewood', 'Cherrywood','Birchwood', 'Walnut', 'Mahogany', 'Pine', 'Beech', 'Ash', 'Hickory', 'Teak', 'Rosewood', 'Alder', 'Cedar', 'Bamboo', 'Plywood', 'Acacia', 'Poplar', 'fir'.

(2) full texture prompt: "a material wooden panel texture, high resolution, 4k, photorealistic".

Objects: "A object name, nice detailed, fancy, photorealistic, inside a home, 4k, natural light"

Full Scenes:

(1) style prompt: ”bright”, ”warm”, ”modern”, ”mediterranean”, ”vintage”, ”contemporary”, ”transitional”.

(2) kitchen: ”a high-resolution picture of a bright style kitchen, very pretty, very natural lighting, ultra-high resolution, 8k, 16k, natural light, photorealistic, realism.”

(3) living room: ”a high-resolution picture of a bright style living room, with sofa, chairs, tv, ottoman, floor lamps, etc, very pretty, very natural lighting, ultra-high resolution, 8k, 16k, natural light, photorealistic, realism”.

C.3 Prompts for BLIP2

In this section, we show examples of how we guide Vision-Language Models such as BLIP2 [59] to produce anwsers that can be converted into comparison results with ours.

Global Parent Prompt: ”which of the wall is this object most likely on? choose one from ’floor’, ’ceiling’, ’front wall’, ’left wall’ and ’right wall’”

Object Base Prompt: ”what’s the name of the object. choose one word from cabinet, oven, dishwasher, washer, fridge, oven fan, shelf, tv, sofa, chair, square table, ottoman, coffee table and stuffed toy.”

Object Position Prompt: ”This image has a width of 512 and height of 512, the object box coordinate x is at 215, if the object scale is from 0 to 12, where do you imagine putting this bounding box relative to the object along the length in the 3D space. Choose from an integer from 0 to 12”

C.4 Compare with Other Scene Generation Methods

We compare our pipeline with other methods of scene generation 10. In particular, we evaluate on (1) If the generated content follows the real-world structure (2) If the method works only on RGB images (3) if the method is fully automatic without human interaction with the scene (4) If it is scalable (5) if it can be applied to global scenes and (6) if the generated scenes are fully articulated.

	Real World Distribution	RGB	Fully Automatic	Scalable	Scene Layout	Articulated Objects
Ditto	✓	✗	✗	✗	✗	✓
Ditto in the house	✓	✗	✗	✗	✓	✓
ProThor	✗	N/A	✓	✓	✓	✗
Phone2Proc	✗	✗	✗	✗	✓	✗
Ours	✓	✓	✓	✓	✓	✓

Figure 10: Comparison against different approaches in scene generation: Ditto, Ditto in the house, ProcThor, Phone2Proc.