Infinigen Indoors: Photorealistic Indoor Scenes using Procedural Generation

Alexander Raistrick*, Lingjie Mei*, Karhan Kayan*, (*equal contribution; random order)
David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh,
Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, Jia Deng
Department of Computer Science, Princeton University

Abstract

We introduce Infinigen Indoors, a Blender-based procedural generator of photorealistic indoor scenes. It builds upon the existing Infinigen system, which focuses on natural scenes, but expands its coverage to indoor scenes by introducing a diverse library of procedural indoor assets, including furniture, architecture elements, appliances, and other day-to-day objects. It also introduces a constraint-based arrangement system, which consists of a domain-specific language for expressing diverse constraints on scene composition, and a solver that generates scene compositions that maximally satisfy the constraints. We provide an export tool that allows the generated 3D objects and scenes to be directly used for training embodied agents in real-time simulators such as Omniverse and Unreal. Infinigen Indoors is open-sourced under the BSD license. Please visit infinigen.org *for code and videos.*

1. Introduction

Synthetic data rendered by conventional computer graphics has seen increasing adoption in computer vision[10, 38, 41, 42, 46, 48, 61] and AI research[20, 28, 76], especially for 3D vision[26, 45, 71, 73–75, 83] and embodied AI[12, 32, 36, 68, 70, 87]. Synthetic data can be rendered in unlimited quantities and can automatically provide high-quality 3D ground truth, enabling large-scale training of computer vision models and embodied agents. Notably, many state-of-the-art 3D vision systems[72, 75] and robotic systems [33, 39] have been trained purely in simulation yet perform surprisingly well in the real world *zero-shot*.

A promising direction for creating synthetic data is procedural generation, which uses mathematical rules to create 3D objects and scenes, as opposed to manual sculpting or real-world scanning. These mathematical rules can have parameters that are randomized to allow infinite variations. For example, trees can be generated through a recursive set of rules that randomly branch off. Compared to reusing a fixed, static set of 3D assets, procedural generation can greatly improve the diversity of the synthetic data and the simulated environments.

Infinigen [60] is a recent work that pushed the idea of procedural generation to the limit. Infinigen is an open-source system that generates photorealistic 3D scenes *fully procedurally*, meaning that every 3D asset, from shape to material, from large structures to small details, is completely procedural, without using any external static asset. Being fully procedural means that every aspect of the 3D scene, from the details of individual objects to their arrangements in a scene, can be customized and controlled by simply modifying the underlying mathematical rules. As a result, a 3D scene can be randomized at all levels down to the smallest details, as opposed to only at the level of object arrangement, which was common in earlier work that used procedural generation.

However, the current Infinigen system is limited to natural scenes and objects (terrains, animals, plants, etc.). Although natural scenes could be sufficient for training foundation models as evidenced by natural evolution [60], this hypothesis remains unproven and may require additional advances in learning algorithms and architecture designs. Evidence from the current literature suggests that synthetic training data that more closely approximates the application domain is still likely to lead to better downstream performance.

To overcome this limitation, we introduce Infinigen Indoors, a procedural generator of photorealistic indoor scenes. It expands the coverage of Infinigen to indoor scenes, which are relevant for many high-impact applications including robotics and augmented reality. Infinigen Indoors generates diverse indoor objects, including furniture, appliances, cookware, dining utensils, architectural elements, and other common day-to-day objects. It also generates full indoor scenes, including the interior of multi-room, multi-floor buildings, with object arrangements that are physically and semantically plausible. Fig. 1 shows random samples of generated scenes, and Fig. 2 shows some automatic annotations.

Like the original Infinigen, Infinigen Indoors is not a fixed set of 3D models or scenes; instead, it is an open-source generator that can create unlimited variations both

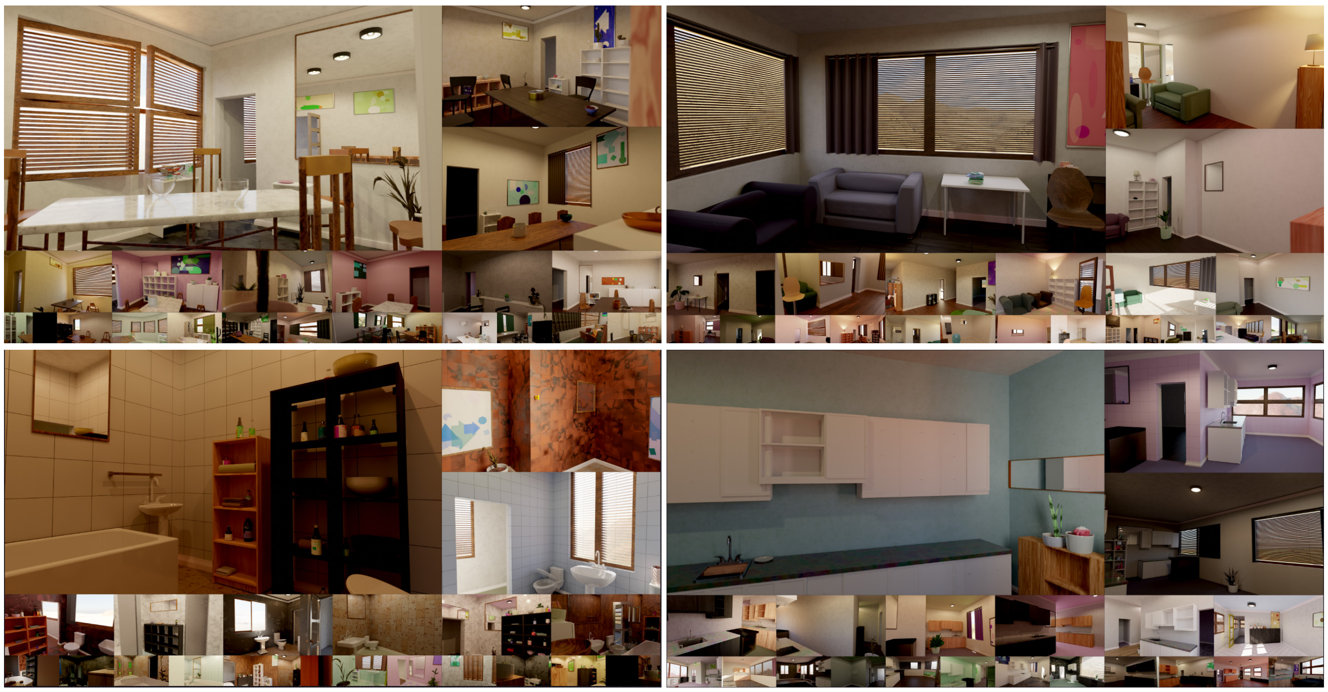
Figure 1. Random, *non cherry-picked* sample of images generated by our system. From top left to bottom right, we show images from dining rooms, bathrooms, living rooms and kitchens. Please see the supplement for an extended random sample.

at the object level and at the scene level. Infinigen Indoors is also 100% procedural, using no external assets and using only mathematical rules to generate everything from scratch.

Infinigen Indoors builds upon the original Infinigen and Blender [8] but makes significant new contributions. The main contributions include (1) a library of procedural generators of indoor assets, (2) a constraint-based arrangement system, (3) a tool to export the generated scenes to real-time simulators such as NVIDIA Omniverse [54] and Unreal Engine[13].

Our second contribution—a constraint-based arrangement system—offers a new capability specifically targeting indoor settings. Indoor scenes are artificial, and object arrangement exhibits a greater degree of regularity than natural scenes: for example, furniture usually does not block the entrance of a room. We thus develop a system that lets the user specify scene arrangement constraints through a domain-specific language using a set of Python APIs. The constraints cover many types of common arrangement, including symmetry ("*Place chairs symmetrically around the table*"), spatial relation ("*Place plant pots close to windows* "), quantity ("*An equal number of knives and forks*"), physics ("*Ensure vases do not overhang*"), and accessibility ("*Ensure there is free-space in-front of all appliances*"). The constraints can be understood as a type of declarative procedural rules, which express what the user desires but not how to achieve it. Like other procedural rules, the constraints can be randomized and can be customized by the user.

In addition to constraint specification, our arrangement system also includes a *constraint solver*, which searches for an arrangement that maximally satisfies a set of given constraints. Our solver greedily performs simulated annealing on whole-house floor plans, followed by large furniture layouts and then small objects. Compared to existing approaches for scene arrangement, our solver is highly expressive, supporting complex compositional constraints that are challenging or infeasible for existing approaches. In addition, it is the first solver integrated with an open-source and fully procedural generator.

Our constraint-based arrangement system is a significant contribution because it vastly improves the generation system's usability and customizability. Because it separates constraint specification from constraint solving, a user can conveniently express the objectives of procedural generation without worrying about implementation. This capability is not available in the original Infinigen, where the user has to customize the procedure rules at the implementation level.

Our third contribution—exporting to real-time simulators—is also noteworthy because it allows the generated 3D objects and scenes to be directly used for training embodied agents in real-time simulators such as Omniverse. Thus, Infinigen Indoors can supply diverse 3D assets for simulation environments and enhance their domain randomization.

To validate the effectiveness of the generated data and demonstrate our system's unique customizability, we use Infinigen Indoors to generate synthetic data for shadow removal and occlusion boundary detection, two tasks that lack

Dataset	Arrangement Method	Procedural Assets	Provides Procedural Code	# Scenes in Total	# Assets in Total	Free Assets	External Asset Source
3DSSG [78]	Real-world scans	No	N/A	1.5K	48K	Yes	None
Matterport3D [6]	Real-world scans	No	N/A	2K	-	Yes	None
Stanford 2D-3D-S [2]	Real-world scans	No	N/A	270	-	Yes	None
ScanNet [9]	Real-world scans	No	N/A	1.5K	-	Yes	None
SceneNN [27]	Real-world scans	No	N/A	100	-	Yes	None
OpenRooms [44]	Real-world scans	No	No	1.3K	3K	No ($500)	ShapeNet [7], Scan2CAD [3], Adobe Stock [30]
Replica [69]	Artist layouts	No	N/A	18	-	Yes	AI Habitat [49]
Structured3D [3]	Artist layouts	No	N/A	22K	472K	No	Professional Designers
Hypersim [63]	Artist layouts	No	N/A	461	59K	No ($6000)	Evermotion Architectures [14]
InteriorNet [43]	Artist layouts	No	N/A	22M	1M	No	Manufactures / Kujiale [37]
Habitat 3.0 [34]	Artist layouts	No	N/A	211	18.7K	Yes	Floorplanner [15], Proffesional Designers
3D-FRONT [17]	Artist layouts	No	N/A	19K	13K	Yes	3D-FUTURE [18]
Robotrix [19]	Artist layouts	No	N/A	16	-	No	UE4Arch [77], UnrealEngine Marketplace [31]
DeepFurniture [47]	Artist layouts	No	N/A	20K	-	No	Adobe Mixamo [29]
SceneNetRGBD [94]	Obj. Cat. Dist.	No	N/A	∞	5.1K	Yes	SceneNet [23], ShapeNet [7]
LUMINOUS [93]	Optimizer	No	Yes	∞	2K	Yes	AI2-THOR [36]
SceneNet [24]	Optimizer	No	No	∞	3.7K	Yes	3DModelFree [16], ModelNet [86], Archive3D [1], Stanford database
ProcTHOR [10]	Heuristics	No	Yes	∞	1.6K	Yes	AI2-THOR [36], Professional Designers
Aria [52]	-	-	No	100K	8K	-	-
Infinigen Indoors (Ours)	Constraint Language	Yes	Yes	∞	∞	Yes	None

Table 1. Comparisons to existing datasets and generators. Many existing datasets/generators use external, static asset libraries and have limited number of scenes. Ours is fully procedural, without using any external source. Dashes represent numbers we could not acquire or estimate.

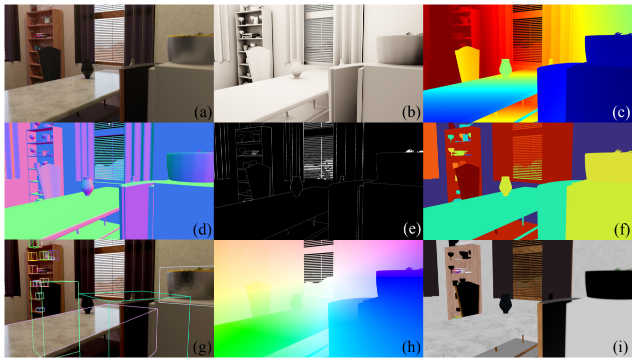

Figure 2. Each image (a) is rendered from a mesh (b), from which we can also extract Depth (c), Surface Normals (d), Occlusion Boundaries (e), Segmentation (f), Bounding Boxes (e) and Optical Flow (h), with Albedo (i) from rendering metadata.

abundant existing training data. Our experiments show that data from our system improves generalization performance on indoor scenes.

Like the original Infinigen, Infinigen Indoors will be open-sourced under the BSD license to enable free and unlimited use by everyone, and to enable community contributions of additional procedural generators.

2. Related Work

We provide a detailed comparison of Infinigen Indoors with existing datasets and generators in Tab. 1.

Real-world datasets. Various real-world datasets have been introduced for indoor scene understanding [2, 6, 9, 27, 66, 67, 78], including the earlier and widely used NYUv2 [66] and Sun RGB-D [67], as well as more recent datasets [2, 4, 9, 78]. However, real-world datasets are labor intensive to collect and limited in size. In addition, real-world 3D ground truth can be difficult to acquire due to the limitations of depth sensors which include limited resolution and range, errors with transparent/reflective surfaces, and artifacts at object edges.

Synthetic Indoor Datasets. There are many existing synthetic datasets for indoor scenes [3, 17, 19, 24, 34, 36, 43, 47, 63, 69]. However, the underlying 3D assets of many datasets are not freely accessible, limiting their utility. In addition, most use a *static* library of 3D assets, limiting their diversity. Recent work [10, 93] has incorporated procedural generation for scene layout and floor plan generation, but still relies on static libraries of objects and materials. In contrast, Infinigen Indoors is 100% procedural, with all assets from shape to texture generated from scratch with unlimited variation.

Object arrangement and layout generation. Constraints are potent tools to describe the layout of a scene. Early works like [88] represent constraints as hard-coded programs, and [50] represent them as physical relations. Data-driven works like [40, 56, 57, 62, 82, 84] learn constraints implicitly from data. Such implicit constraints are less customizable, interpretable, and controllable than Infinigen Indoors. Recently, modeling constraints using probabilistic graphs have become more popular: [89] uses pairwise grouping, while [11, 91] further extends it to spatial and hierarchical constraints. [90] uses factor graphs to parse the constraints, while [51] models them with Bayesian network. [58, 92] formulates constraints as potentials Markov Random Fields on a fixed graph, which capture only non-compositional and associative constraints for rooms and objects.

Compared to existing systems, ours is the first to integrate directly with procedural object generators, and our constraint language is higher level and more easily extendable than existing systems. Our system specifies high level goals for abstract classes of objects (e.g. 'furniture', 'stor-

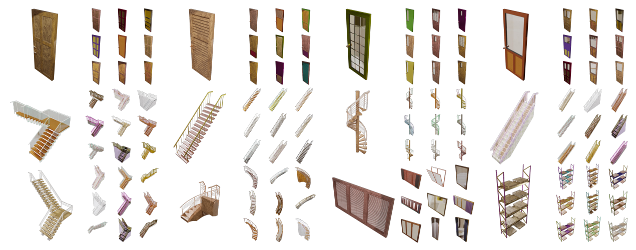
Figure 3. Random samples of procedurally generated doors (top), staircases (middle/bottom) and windows/warehouse shelving (bottom-right).

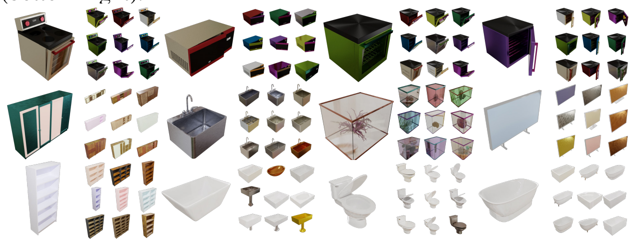
Figure 4. Random samples of procedurally generated ovens, dishwasher and sinks (top/middle), living-room furniture (middle) and bathroom fixtures (bottom).

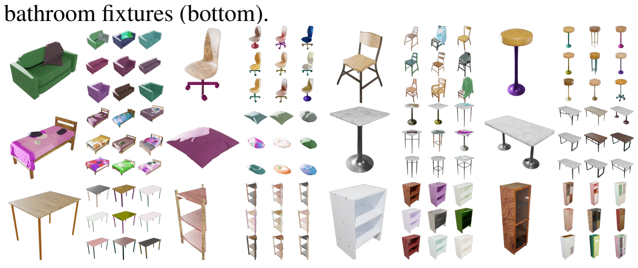
Figure 5. Random samples of procedurally generated furniture, including sofa, chairs and beds (top), tables (middle/bottom), and shelves (bottom).

age'), rather than exhaustive distance/angle distributions for specific objects [91] which must be fitted to example scenes. Our language also supports compositional constraints such as "place glassware only on shelves against a diningroom wall." These features allow users to write new constraints for their specific needs, including domains without existing artist-made scenes. Our constraint solver uses simulated annealing, following prior work [91], but involves moves that are unique to our constraint language, including updates to object-object relations or changing the parameters of procedural objects (e.g. the size of a table).

3. Method

3.1. Procedural Asset Generation

All assets used in Infinigen Indoors are generated from scratch by compact probabilistic programs. These programs have many human-controllable parameters, which are randomized by default, or can be manually overridden by the

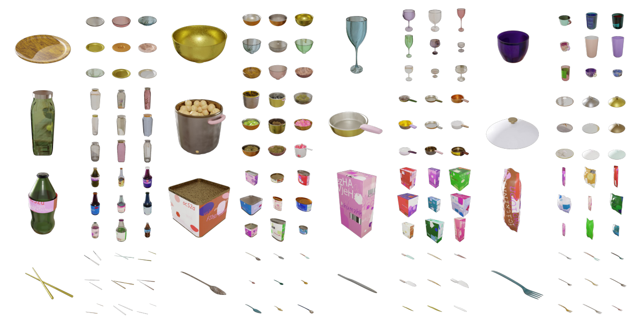
Figure 6. Random samples of procedurally generated tableware, including dinnerware (Row 1-2), cookware (Row 2), food containers (Row 3) and dining utensils (Row 4).

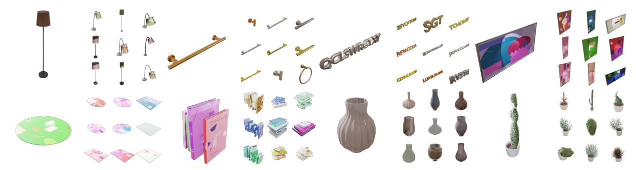
Figure 7. Random samples of procedurally generated home decorations, including lamps, hardware, balloons, wall decor (top), rugs, book stacks, vases and plants (bottom). Small assets for decoration purposes, usually attached to the ground or walls.

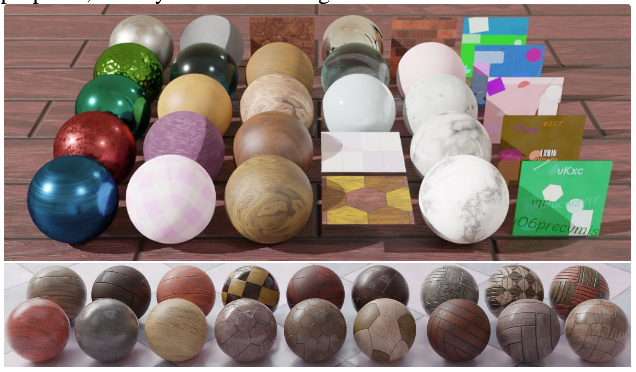
Figure 8. A collection of materials generated in Infinigen Indoors. The first figure shows one material per generator, with columns (1-3) used on assets of various sizes, (4-5) used on assets and rooms, and (6) for abstract art and text. The second figure shows multiple materials from the same generator with different parameters.

user. These parameters are used along with additional low-level random noise to generate meshes via geometry nodes, modifiers, or mesh manipulations in Blender. We provide a total of 79 randomized procedural object generators. By category, we cover Appliances (10 generators, 112 params), Windows/Doors/Staircases (14 generators, 127 params), Furniture (17 generators, 216 params), Decorations (15 generators, 92 params), and Small Objects (19 generators, 194 params). See the supplement for a list.

Architectural elements shown in Fig. 3 are integrated into room as fixtures. We use array repetition of atomic compo-

```
tables = scene.with_semantics(Tag.Chair)
chairs = scene.with_semantics(Tag.Table)
on_floor = cl.StableAgainst(Tag.Bottom, Tag.Floor)
constraints = cl.forall(
    rooms[Tag.DiningRoom],
    r,
    (tables.related_to(r, on_floor).count() == 1,
     chairs.related_to(r, on_floor).count().in_range(2, 10)),
)
score = (
    chairs.related_to(rooms[Tag.DiningRoom], on_floor)
    .mean(c, cl.reflection_symmetry(c, tables) + \
                cl.min_distance(t, tables) * (-1))
    .maximize(weight=2)
)
```

Figure 9. Example usage of our constraint specification API, specifying the quantity and aesthetic constraints for a dining table and chairs.

nents to build staircases, and mesh booleaning to cut out the panels of doors and windows.

Large objects shown in Fig. 4 and 5 provide assets related to cooking, seating and storage. We use soft-body collision simulation to model soft blankets, clothing, and stuffed pillows when they are put on a supporting surface.

Small objects shown in Fig. 6 and 7 can be attached to support surfaces, walls, or ceilings. We also devised a combined text-and-shape logo generator that produces procedural texture for fabrics, food packaging, and art decor. We use cloth simulation to inflate balloons and food packaging with air.

Materials are all procedurally created with Blender's shader nodes, as shown in Fig. 8. We provide 30 material generators with 120 controllable parameters total, split approximately evenly between types of Wood, Ceramic, Fabric, Metal, and others. We cover 78% of OpenSurfaces'[5] material categories, up from 21% for Infinigen.

3.2. Constraint Specification API

Indoor scene layouts are highly regular, and follow complex rules governing ergonomics, aesthetics and functionality. Moreover, the rules that apply to a particular object depend on context - for example, tables are placed against walls when used as desks in study rooms, but must be far from walls and surrounded by chairs when used in a dining room. To capture this, we provide a high-level Constraint Specification API, which allows the user to write complex geometric scoring functions of sets of objects in the scene. This score is optimized by the solver to find a satisfactory layout. An example of the Constraint Specification API is shown in 9.

Each constraint in the specification API can be expressed as a composition of geometric operators. Geometric operators are designed to compute spatial and geometric properties, including *minimum distance*, *rotational and reflection symmetry*, *angle alignment*, *2D free-space*, *accessibility* and *volume* or *area* of objects. By composing these operators with nested filtering by semantics and relations, the user can create scoped constraints that apply only to objects attached to specific surfaces or rooms. Additionally, these geometry terms are affected by the parameters (length, width, etc.) of the procedurally generated assets, meaning that optimizers can automatically discover optimal furniture parameters given available space and constraints. Our system features common scalar arithmetic operations, comparisons, and `forall` / `sum` operators to gather results over sets of objects. Please see the supplement for the full API and more constraint specification examples.

A more concrete example of our constraint language can be found in Fig. 9, where the constraint program specifies common-sense human ergonomics and semantic relations found in residential homes. This constraint graph has a total of 1058 nodes, which compute 11 hard constraints and 25 score terms (soft constraints). We provide example constraint specifications for living rooms, bathrooms, dining rooms, kitchens, and warehouses. See the supplement for the full constraint program and example constraints.

For generating training data, we believe many users will consider creating custom constraints tailored to particular applications. Our constraint system is designed to allow easy customization. Our initial spec. has avg. 15 constraints specific to each room: approx. 15 lines of Python. We believe this cost is very tractable when users need customization.

3.3. Arrangement Solver

Because our Constraint Specification API is flexible, the solver needs to search a prohibitively large space in which finding an exact minimum is impossible. To deal with this, it uses Simulated Annealing[35] with Metropolis-Hastings criterion [25, 53]. The solver first takes the current state s and randomly chooses a move category. It then uses the constraint graph to generate a proposed state s' that can be reached using the move. The current and proposed states are evaluated on the graph specified by the provided constraints and score terms, yielding loss terms $l(s)$ and $l(s')$. Then, the solver calculates transition probability between the s and s' as

$$p(s'|s) = \min[\exp(l(s) - l(s'))/\tau, 1]$$

where τ is the temperature of the solver, which cools exponentially from $\tau = 0.25$ to $\tau = 0.001$.

Our solver allows both discrete and continuous moves:

Addition - Adds a procedural object to the scene.
Deletion - Deletes an object from the scene.
Relation Plane Change - Assigns an object to another plane.
Resample - Regenerates an object with new parameters.
Reinitialize Pose - Samples a new random pose for an object.
Translate - Translates the object within its DOF plane.
Rotate - Rotates the object around its DOF axis.

We observe that not all moves are equally significant at each point in the optimization. In an empty scene, object addition and relation change allow for higher loss reduction, whereas in a cluttered scene, continuous object movement

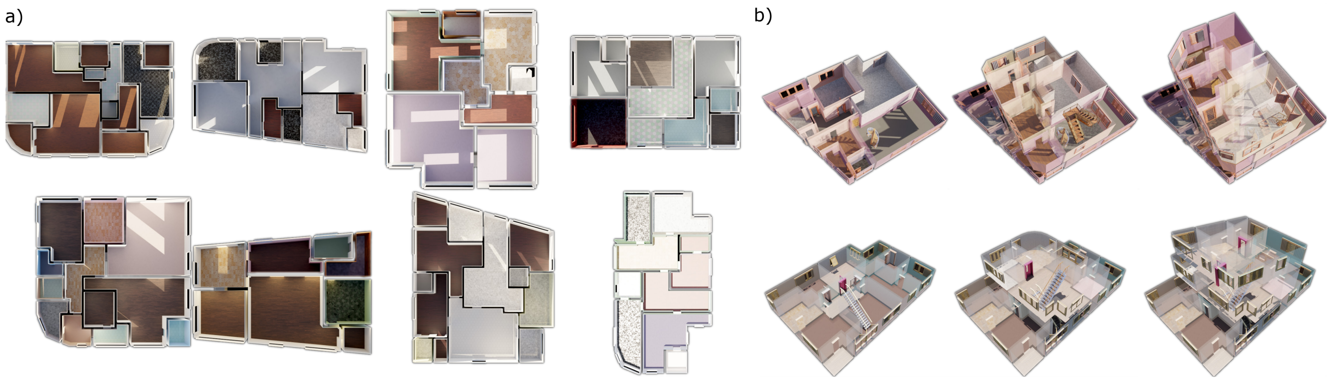

Figure 10. In Fig. a), we show ten randomly generated single-story floor plans with a diverse set of room combinations, connectedness, and overall contours. In Fig. b), we show results for the generation of multistory floor plans. Floors 0,1,2 are displayed separately. Staircases connect adjacent floors. We remove exterior walls and ceilings for visibility.

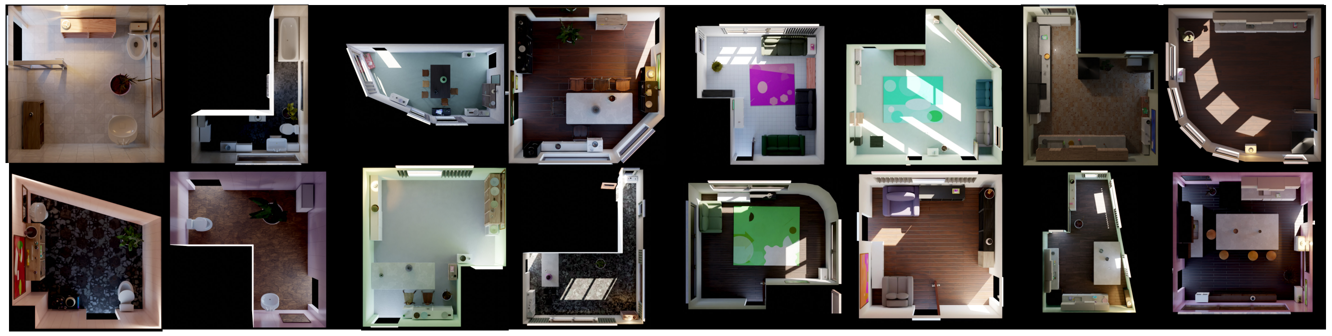

Figure 11. Qualitative room arrangement results, grouped by room type. From left to right, we show bathrooms, dining rooms, living rooms and kitchens.

allows for higher loss reduction. Thus, we provide a schedule for moves so that probabilities for discrete moves decay gradually and probabilities for continuous moves increase.

The objects in an indoor scene are interdependent on each other, which makes it unfeasible to optimize over all of them simultaneously. However, they usually depend on each other hierarchically (e.g. a cup is on the table, which is on the floor). To exploit this hierarchy, we divide our optimization into three stages: large object optimization, medium object optimization, and small object optimization.

Each object is constrained in its movement due to the constraints specified by the user and the discrete moves proposed by the solver. For instance, a bookshelf that is stable against a wall is only allowed to move along the 1D line between the wall and the floor. Consequently, an object's degrees of freedom (DoFs) for rotation and translation are determined based on its relations to other objects. When the solver samples a continuous move, it restricts the object's motion to these DoFs. When the solver samples a discrete move, it places the object in the constrained subspace.

Floorplan-specific solver and constraints Our floorplan generator creates realistic full-house room meshes as shown in Fig. 10. First, we procedurally generate a room adjacency graph specifying the number, type, and connectedness of individual rooms required in the floor plan. This graph is produced by inference on a probabilistic context-free grammar on room types, or can be wholly or partially derived from user input. We define our objective function as a weighted combination of the terms below, and optimize it using simulated annealing subject to constraints from the room adjacency graph. See the supplement for full definitions.

- Shortest path to entrance
- Typical room area
- Room aspect ratio
- Room convexity
- Room wall conciseness
- Functional room area
- Room collinearity
- Narrow passages
- Exterior length by room
- Exterior corners by room
- Staircase occupancies
- Staircase IOU with rooms

We initialize our floorplan solver by generating a random house outline, and subdividing it using a Mondrian Process [65] until it produces sufficient spaces for each room. We extrude a wall segment inwards or outwards at each step, or swap the assignment subject to the room adjacency graph. Either action will lead to a change in loss, which we convert to an acceptance probability using Metropolis-Hasting as in Sec 3.3. Once solving is complete, we add floor, wall, and ceiling materials, doors, windows, and staircases, all subject to constraints based on room type and adjacency. Fig

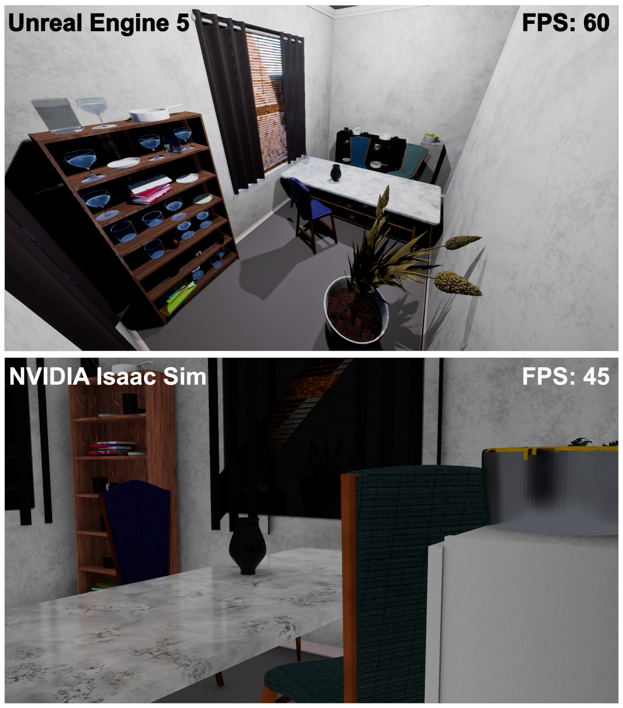

Figure 12. A single scene from two different viewpoints in Unreal Engine 5 (above) and Omniverse Isaac Sim (below). Unreal Engine runs at 60 FPS and Isaac Sim with physics simulation enabled runs at 45 FPS, both on RTX 3090s. Differences in lighting are caused by the variations in simulator import systems.

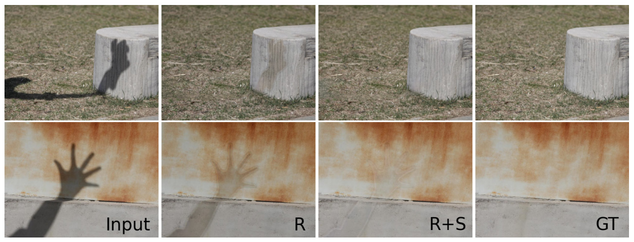

Figure 13. Qualitative zero-shot results on SRD [59] test dataset.

11 shows the solutions to room arrangements with objects placed inside.

3.4. Data Export

		All Image		Shadow Region		Non-Sh Region	
Test Set	Model	PSNR↑	RMSE↓	PSNR↑	RMSE↓	PSNR↑	RMSE↓
ISTD [81]	R	**31.96**	**4.27**	**38.04**	**6.58**	**34.13**	**3.85**
	R+S	31.72	4.30	37.41	7.06	33.80	3.85
SRD [59]	R	22.83	10.84	25.59	18.75	27.93	7.68
	R+S	**24.56**	**9.54**	**27.39**	**16.63**	**29.37**	**6.67**

Table 2. Shadow removal task quantitative performance on ISTD, ISTD+ and SRD dataset across the three variations of the model.

We develop a one-click tool to export assets from Infinigen Indoors to real-time simulators, using Universal Scene Description (USD) or other formats. As seen in Fig. 12, indoor scenes can be exported to Omniverse Isaac Sim and Unreal Engine 5 and can be run at interactive frame rates. This exporting capability allows Infinigen Indoors to help train embodied agents in virtual environments.

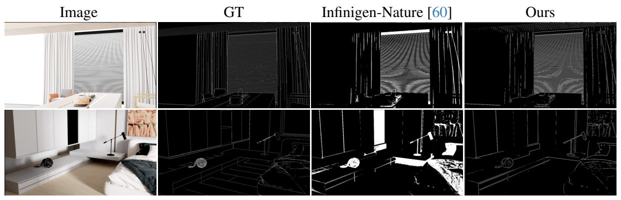

Figure 14. Qualitative results on synthetic artistic scenes [21].

Infinigen Indoors uses Blender's procedural material system, which is by default not portable to other simulators or scene editors. To resolve this, we provide tools to automatically post-process and UV-map entire indoor scenes, and use texture baking to create standard texture maps for material color, roughness, metallicity and more. We also provide export code to convert single objects to textured OBJ, FBX or STL meshes, and automatically generate collision and articulation information as Universal Robot Description Format (URDF) files.

4. Experiments

4.1. Solver Performance

To efficiently solve large numbers of constraints in cluttered scenes, we optimized our solver with various features for faster convergence. Plane hashing enables faster access to bounding planes during discrete optimization. BVH caching enables faster mesh distance and collision calculations by reusing Bounding-Volume-Hierarchies except when mutated by a state update. Evaluation caching maintains a cache of results in the evaluation graph. Move filtering narrows down the search space in continuous optimization by selectively pruning candidates to those that can reduce the loss. Placeholder optimization only generates full meshes when other objects are assigned to them; otherwise it keeps bounding boxes. See the supplement for more details.

To analyze the importance of these features, we conducted an ablation in Tab. 3 and Fig. 15, which shows solver performance with each feature removed. All results are averaged over 20 random scenes with 5k solver steps. Our full system provides a $\approx$ 3x speedup compared to the non-optimized version. Most of the performance gains come from BVH caching and Plane Hashing. We observe that discrete changes such as pose re-initialization, relation changes, and object resampling are necessary for compelling visuals, but decrease the quantitative score and increase the runtime. When we run our fully optimized system as long as the non-optimized version we get a 28% score increase.

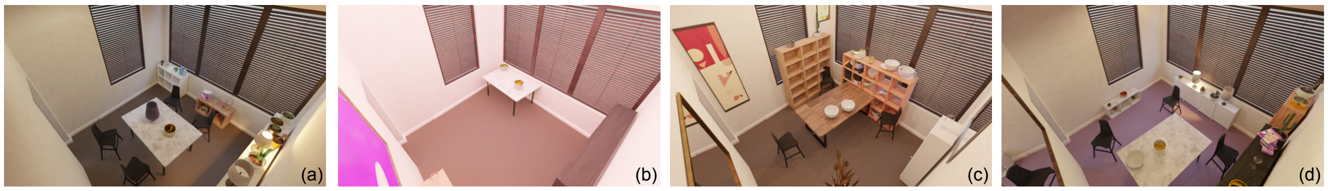

Figure 15. Qualitative Ablation. From left to right, we show scenes generated by our system with 10K solving steps (a), 1K solving steps (b), with collision checking removed (c) and with symmetry terms disabled (d)

Method	Runtime ↓	Avg. Score (s) ↑	#Objects ↑
Full System	2280.68	80.84	28.94
w/o Discrete change	**1872.60**	86.20	30.82
w/o Eval. cache	2242.79	80.84	28.94
w/o Move Filter	2633.95	84.75	45.06
w/o Placeholder optim.	2665.18	80.46	27.71
w/o Plane Hashing	3522.86	92.65	34.12
w/o BVH cache	4740.48	80.84	29.06
Full System 100 min.	6080.84	**114.72**	**49.65**
w/o Optimization	6308.92	89.43	47.76

Table 3. Ablation of solver performance optimizations.

Perceptual Study We recruited human subjects following [57] to evaluate the realism of the scenes and the realism of the layouts. Subjects preferred Infinigen Indoors over [11, 57, 62, 84] in terms of both realism, layout realism, and the lack of errors (caveat: "realism" may be influenced by asset and lighting quality). See the supp. for more details.

4.2. Shadow Removal

To demonstrate its flexibility in data generation, we used Infinigen-Indoors to create a dataset consisting of 2k image pairs of shadow and shadow-free variants. These pairs were generated by toggling the shadow property of lighting within Blender. For each pair, shadow masks were produced using Otsu's thresholding [55] method. We use ShadowFormer [22] model for the experiments and consider two variations: only trained on ISTD [81] real dataset (R), and trained on combination of ISTD and 2k Infinigen Indoors synthetic dataset (R+S). The results are shown in Tab. 2. While using synthetic data leads to slightly worse performance on ISTD dataset, the zero-shot application on SRD [59] dataset shows clear improvement for generalization to new test datasets. Qualitative results are shown in Fig. 13.

4.3. Occlusion Boundary Estimation

To validate the effectiveness of Infinigen Indoors, we also evaluate on the task of occlusion boundary estimation, a task with limited available data. We produce 1464 images annotated with ground truth. We train three U-Net [64] models from scratch separately on these images, on images generated from Infinigen [60] and Hypersim [63]. We then compare their performance on a curated test set of photorealistic artist-designed synthetic 3D scenes for architecture visualization [21], since no existing photorealistic indoor dataset provides such annotations. See the supplement for more details.

Training Dataset	ODS	OIS	mAP
Infinigen-Nature [60]	14.38	19.43	10.80
Hypersim [63]	26.02	19.44	15.69
Infinigen Indoors (Ours)	**29.47**	**30.29**	**19.09**

Table 4. Occlusion boundary quantitative results on a curated test set of photorealistic artist-designed synthetic 3D scenes for architecture visualization [21].

We report the following three metrics [79, 80, 85]: (i) *optimal dataset scale F-score (ODS)*, representing the best F-score achieved on the dataset using a uniform threshold across all test images; (ii) *optimal image scale F-score (OIS)* indicating the cumulative F-score on the dataset obtained with thresholds dependent on individual images; and (iii) *mean average precision (mAP)* denoting the mean precision across the complete recall range.

As we can see in Tab. 4, our Infinigen Indoors-trained model generalizes better. The model achieves higher performance across all metrics. These findings underscore the usefulness of Infinigen Indoors as a valuable training resource. Qualitative results are depicted in Fig. 14.

5. Contributions & Acknowledgements

Alexander Raistrick, Lingjie Mei and Karhan Kayan contributed equally and are ordered randomly. Each has the right to list their name first in their CV. Alexander Raistrick performed team coordination and developed the constraint language and object graph solver. Lingjie Mei developed the room solver and many procedural assets. Karhan Kayan developed the geometric constraints and object pose solver. David Yan developed the scene exporter and other utilites. Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Zeyu Ma and Lahav Lipson developed procedural assets and utilities. Jia Deng conceptualized and led the project, and set directions. This work was partially supported by the National Science Foundation and Amazon.

References

[1] Archive 3D. Archive 3d. https://archive3d.net/. 3

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, 2017. 3

[3] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 2614–2623, 2019. 3

[4] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. Arkitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021. 3

[5] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics (SIGGRAPH)*, 32(4), 2013. 5

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 3

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 2

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3

[10] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 1, 3

[11] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *ArXiv*, abs/2206.06994, 2022. 3, 8

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. pages 1–16, 2017. 1

[13] Epic Games. Unreal engine. 2

[14] Evermotion. Evermotion architectures. https://evermotion.org/shop. 3

[15] Floorplanner. Floorplanner.com. https://floorplanner.com/. 3

[16] 3D Model Free. 3d model free. http://www.3dmodelfree.com/. 3

[17] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 3

[18] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3D-FUTURE: 3D furniture shape with texture. *IJCV*, 129(12):3313–3337, 2021. 3

[19] Alberto Garcia-Garcia, Pablo Martinez-Gonzalez, Sergiu Oprea, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Alvaro Jover-Alvarez. The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6790–6797. IEEE, 2018. 3

[20] John T Guibas, Tejpal S Virdi, and Peter S Li. Synthetic medical images from dual generative adversarial networks. *arXiv preprint arXiv:1709.01872*, 2017. 1

[21] Gumroad. Gumroad. https://discover.gumroad.com/. 7, 8

[22] Lanqing Guo, Siyu Huang, Dingshuo Liu, Hao Cheng, and Bihan Wen. Shadowformer: Global context helps image shadow removal. *ArXiv*, abs/2302.01650, 2023. 8

[23] Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743. IEEE, 2016. 3

[24] Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743, 2016. 3

[25] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. _eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. 5

[26] Rasmus Laurvig Haugaard and Anders Glent Buch. Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6749–6758, 2022. 1

[27] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 fourth international conference on 3D vision (3DV)*, pages 92–101. Ieee, 2016. 3

[28] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE, 2019. 1

[29] Adobe Inc. Adobe mixamo. https://www.mixamo.com, . 3

[30] Adobe Inc. Adobe stock. https://stock.adobe.com/3d-assets, . 3

[31] Epic Games Inc. Unreal engine marketplace. https://www.unrealengine.com/marketplace/en-US/store, . 3

[32] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. 1

[33] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Mueller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620:982–987, 2023. 1

[34] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Schacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. *arXiv preprint arXiv:2306.11290*, 2023. 3

[35] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6): 975–986, 1984. 5

[36] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017. 1, 3

[37] Kujiale. Kujiale.com. https://www.kujiale.com/. 3

[38] Hei Law and Jia Deng. Label-free synthetic pretraining of object detectors. *arXiv preprint arXiv:2208.04268*, 2022. 1

[39] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5, 2020. 1

[40] Kurt Leimer, Paul Guerrero, Tomer Weiss, and Przemyslaw Musialski. LayoutEnhancer: Generating Good Indoor Layouts from Imperfect Data. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–8, 2022. arXiv:2202.00185 [cs]. 3

[41] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *CVPR*, 2022. 1

[42] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16263–16272, 2022. 1

[43] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. *arXiv preprint arXiv:1809.00716*, 2018. 3

[44] Zhengqin Li, Ting Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Milovs Havsan, Ravi Ramamoorthi, and Manmohan Chandraker. Openrooms: An open framework for photorealistic indoor scene datasets. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7186–7195, 2020. 3

[45] Lahav Lipson, Zachary Teed, and Jia Deng. RAFT-Stereo: Multilevel recurrent field transforms for stereo matching. In *International Conference on 3D Vision (3DV)*, 2021. 1

[46] Lahav Lipson, Zachary Teed, Ankit Goyal, and Jia Deng. Coupled iterative refinement for 6d multi-object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6728–6737, 2022. 1

[47] Bingyuan Liu, Jiantao Zhang, Xiaoting Zhang, Wei Zhang, Chuanhui Yu, and Yuan Zhou. Furnishing your room by what you see: An end-to-end furniture set retrieval framework with rich annotated benchmark dataset. *arXiv preprint arXiv:1911.09299*, 2019. 3

[48] Zeyu Ma, Zachary Teed, and Jia Deng. Multiview stereo with cascaded epipolar raft. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, pages 734–750. Springer, 2022. 1

[49] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3

[50] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2697–2706, 2017. 3

[51] Paul C. Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. *ACM SIGGRAPH Asia 2010 papers*, 2010. 3

[52] META. Project aria. https://www.projectaria.com/datasets/ase/. 3

[53] Nicholas C. Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and A. H. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953. 5

[54] NVIDIA. Omniverse. https://developer.nvidia.com/omniverse. 2

[55] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. 8

[56] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas Guibas, and Peter Wonka. Generative Layout Modeling using Constraint Graphs, 2020. arXiv:2011.13417 [cs]. 3

[57] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems*, pages 12013–12026. Curran Associates, Inc., 2021. 3, 8

[58] Siyuan Qi, Yixin Zhu, Huang Siyuan, Chenfanfu Jiang, and Song Zhu. Human-centric indoor scene synthesis using stochastic grammar. 2018. 3

[59] Liangqiong Qu, Jiandong Tian, Shengfeng He, Yandong Tang, and Rynson W. H. Lau. Deshadownet: A multi-context embedding deep network for shadow removal. In *2017 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 2308–2316, 2017. 7, 8

[60] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12630–12641, 2023. 1, 7, 8

[61] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118. Springer, 2016. 1

[62] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models, 2018. arXiv:1811.12463 [cs]. 3, 8

[63] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10912–10922, 2021. 3, 8

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 8

[65] Daniel M. Roy and Yee Whye Teh. The mondrian process. In *Neural Information Processing Systems*, 2008. 6

[66] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 601–608, 2011. 3

[67] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3

[68] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. pages 477–490, 2022. 1

[69] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 3

[70] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[71] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020. 1

[72] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In *Neural Information Processing Systems*, 2021. 1

[73] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *NeurIPS*, 2021. 1

[74] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021.

[75] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. *arXiv preprint arXiv:2208.04726*, 2022. 1

[76] Vajira Thambawita, Pegah Salehi, Sajad Amouei Sheshkal, Steven A Hicks, Hugo L Hammer, Sravanthi Parasa, Thomas de Lange, Pål Halvorsen, and Michael A Riegler. Singan-seg: Synthetic training data generation for medical image segmentation. *PloS one*, 17(5):e0267976, 2022. 1

[77] UE4Arch. Ue4arch. `https://ue4arch.com/`. 3

[78] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[79] Chaohui Wang, Huan Fu, Dacheng Tao, and Michael J Black. Occlusion boundary: A formal definition & its detection via deep exploration of context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2641–2656, 2020. 8

[80] Guoxia Wang, Xiaochuan Wang, Frederick WB Li, and Xiaohui Liang. Doobnet: Deep object occlusion boundary detection from an image. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part VI 14*, pages 686–702. Springer, 2019. 8

[81] Jifeng Wang, Xiang Li, and Jian Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *CVPR*, 2018. 7, 8

[82] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Planit: planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.*, 38:132:1–132:15, 2019. 3

[83] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. In *Conference on Robot Learning*, pages 1761–1772. PMLR, 2021. 1

[84] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. *2021 International Conference on 3D Vision (3DV)*, pages 106–115, 2020. 3, 8

[85] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Learning to detect motion boundaries. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2578–2586, 2015. 8

[86] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 3

[87] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. SAPIEN: A simulated part-based interactive environment. In *CVPR*, 2020. 1

[88] Ken Xu, James Stewart, and Eugene Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface*, 2002. 3

[89] Wenzhuo Xu, Bin Wang, and Dong-Ming Yan. Wall grid structure for interior scene synthesis. *Comput. Graph.*, 46: 231–243, 2015. 3

[90] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)*, 31:1 – 11, 2012. 3

[91] Lap-Fai Craig Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and S. Osher. Make it home: automatic optimization of furniture arrangement. *ACM SIGGRAPH 2011 papers*, 2011. 3, 4

[92] Yizhou Zhao, Kaixiang Lin, Zhiwei Jia, Qiaozi Gao, Govind Thattai, Jesse Thomason, and Gaurav S. Sukhatme. Luminous: Indoor scene generation for embodied ai challenges. *ArXiv*, abs/2111.05527, 2021. 3

[93] Yizhou Zhao, Kaixiang Lin, Zhiwei Jia, Qiaozi Gao, Govind Thattai, Jesse Thomason, and Gaurav S Sukhatme. Luminous: Indoor scene generation for embodied ai challenges. *arXiv preprint arXiv:2111.05527*, 2021. 3

[94] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 519–535. Springer, 2020. 3