Infinite Photorealistic Worlds using Procedural Generation

Alexander Raistrick*, Lahav Lipson*, Zeyu Ma*, (*equal contribution; alphabetical order)
Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han,
Yihan Wang, Alejandro Newell[†], Hei Law[†], Ankit Goyal[†], Kaiyu Yang[†], Jia Deng
Department of Computer Science, Princeton University

Abstract

We introduce Infinigen, a procedural generator of photorealistic 3D scenes of the natural world. Infinigen is entirely procedural: every asset, from shape to texture, is generated from scratch via randomized mathematical rules, using no external source and allowing infinite variation and composition. Infinigen offers broad coverage of objects and scenes in the natural world including plants, animals, terrains, and natural phenomena such as fire, cloud, rain, and snow. Infinigen can be used to generate unlimited, diverse training data for a wide range of computer vision tasks including object detection, semantic segmentation, optical flow, and 3D reconstruction. We expect Infinigen to be a useful resource for computer vision research and beyond. Please visit infinigen.org *for videos, code and pre-generated data.*

1. Introduction

Data, especially large-scale labeled data, has been a critical driver of progress in computer vision. At the same time, data has also been a major challenge, as many important vision tasks remain starved of high-quality data. This is especially true for 3D vision, where accurate 3D ground truth is difficult to acquire for real images.

Synthetic data from computer graphics is a promising solution to this data challenge. Synthetic data can be generated in unlimited quantity with high-quality labels. Synthetic data has been used in a wide range of tasks [10, 18, 44, 46, 52, 55, 65], with notable successes in 3D vision, where models trained on synthetic data can perform well on real images zero-shot [31, 51, 75–78, 82].

Despite its great promise, the use of synthetic data in computer vision remains much less common than real images. We hypothesize that a key reason is the limited diversity of 3D assets: for synthetic data to be maximally useful, it needs to capture the diversity and complexity of the real world, but existing freely available synthetic datasets are mostly restricted to a fairly narrow set of objects and shapes, often driving scenes (e.g. [35, 65]) or human-made objects in indoor environments (e.g. [25, 53]).

In this work, we seek to substantially expand the coverage of synthetic data, particularly objects and scenes from the natural world. We introduce Infinigen, a procedural generator of photorealistic 3D scenes of the natural world. Compared to existing sources of synthetic data, Infinigen is unique due to the combination of the following properties:

- *Procedural:* Infinigen is not a finite collection of 3D assets or synthetic images; instead, it is a *generator* that can create infinitely many distinct shapes, textures, materials, and scene compositions. Every asset, from shape to texture, is entirely procedural, generated from scratch via randomized mathematical rules that allow infinite variation and composition. This sets it apart from datasets or dataset generators that rely on external assets.
- *Diverse:* Infinigen offers a broad coverage of objects and scenes in the natural world, including plants, animals, terrains, and natural phenomena such as fire, cloud, rain, and snow.
- *Photorealistic:* Infinigen creates highly photorealistic 3D scenes. It achieves high photorealism by procedurally generating not only coarse structures but also fine details in geometry and texture.
- *Real geometry*: unlike in video game assets, which often use texture maps to fake geometrical details (e.g. a surface appears rugged but is in fact flat), all geometric details in Infinigen are real. This ensures accurate geometric ground truth for 3D reconstruction tasks.
- *Free and open-source:* Infinigen builds on top of Blender [17], a free and open-source graphics tool. Infinigen's code is released for free under the GPL license, same as Blender. Anyone can freely use Infinigen to obtain unlimited assets and renders [‡].

[†]work done while a student at Princeton University

[‡]The output of GPL code is generally not covered by GPL. See `www.gnu.org/licenses/gpl-faq.en.html#WhatCaseIsOutputGPL`

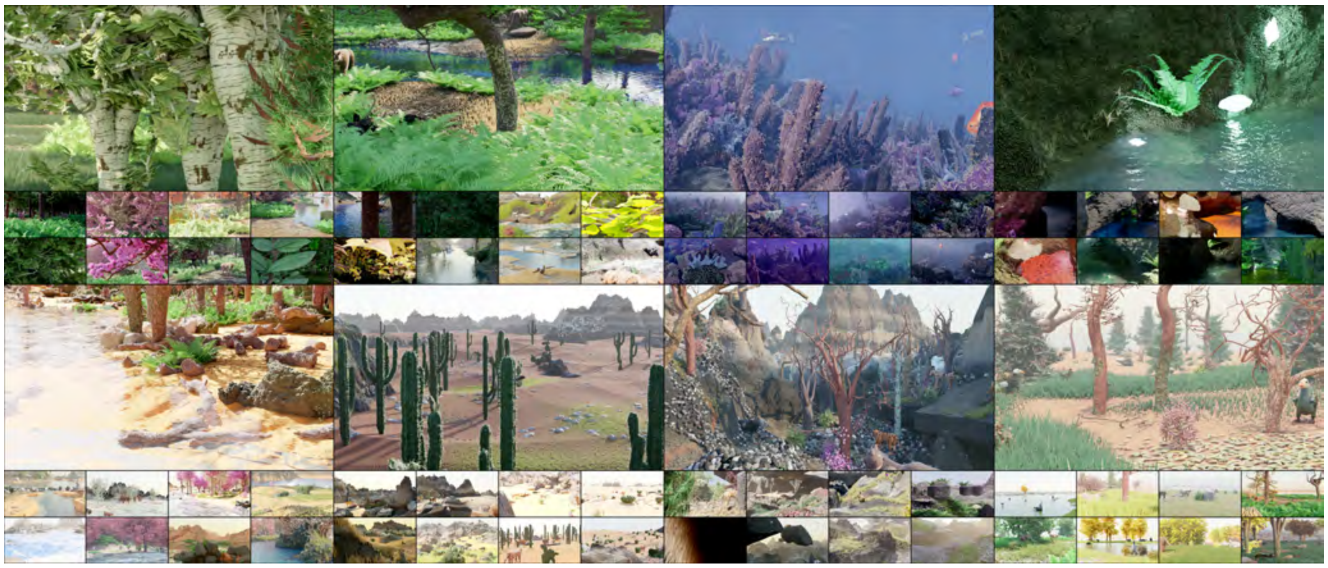


Figure 1. Randomly generated, *non cherry-picked* images produced by our system. From top left to bottom right: Forest, River, Underwater, Caves, Coast, Desert, Mountain and Plains. See the supplement for larger, higher resolution samples.

Infinigen focuses on the natural world for two reasons. First, accurate perception of natural objects is demanded by many applications, including geological survey, drone navigation, ecological monitoring, rescue robots, agriculture automation, but existing synthetic datasets have limited coverage of the natural world. Second, we hypothesize that the natural world alone can be sufficient for pretraining powerful "foundation models"—the human visual system was evolved entirely in the natural world; exposure to human-made objects was likely unnecessary.

Infinigen is useful in many ways. It can serve as a generator of unlimited training data for a wide range of computer vision tasks, including object detection, semantic segmentation, pose estimation, 3D reconstruction, view synthesis, and video generation. Because users have access to all the procedural rules and parameters underlying each 3D scene, Infinigen can be easily customized to generate a large variety of task-specific ground truth. Infinigen can also serve as a generator of 3D assets, which can be used to build simulated environments for training physical robots as well as virtual embodied agents. The same 3D assets are also useful for 3D printing, game development, virtual reality, film production, and content creation in general.

We construct Infinigen on top of Blender [17], a graphics system that provides many useful primitives for procedural generation. Utilizing these primitives we design and implement a library of procedural rules to cover a wide range of natural objects and scenes. In addition, we develop utilities that facilitate creation of procedural rules and enable all Blender users including non-programmers to contribute; the utilities include a transpiler that automatically converts Blender node graphs (intuitive visual representation of procedural rules often used by Blender artists) to Python code. We also develop utilities to render synthetic images and extract common ground truth labels including depth, occlusion boundaries, surface normals, optical flow, object category, bounding boxes, and instance segmentation. Constructing Infinigen involves substantial software engineering: the latest main branch of Infinigen codebase consists of 40,485 lines of code.

In this paper, we provide a detailed description of our procedural system. We also perform experiments to validate the quality of the generated synthetic data; our experiments suggest that data from Infinigen is indeed useful, especially for bridging gaps in the coverage of natural objects. Finally, we provide an analysis on computational costs including a detailed profiling of the generation pipeline.

We expect Infinigen to be a useful resource for computer vision research and beyond. In future work, we intend to make Infinigen a living project that, through open-source collaboration with the whole community, will expand to cover virtually everything in the visual world.

2. Related Work

Synthetic data from computer graphics have been used in computer vision for a wide range of tasks [53, 65]. We refer the reader to [60] for a comprehensive survey. Below we categorize existing work in terms of application domain, generation method, and accessibility. Tab. 1 provides detailed comparisons.

Application Domain. Synthetic datasets or dataset generators have been developed to cover a variety of domains. The

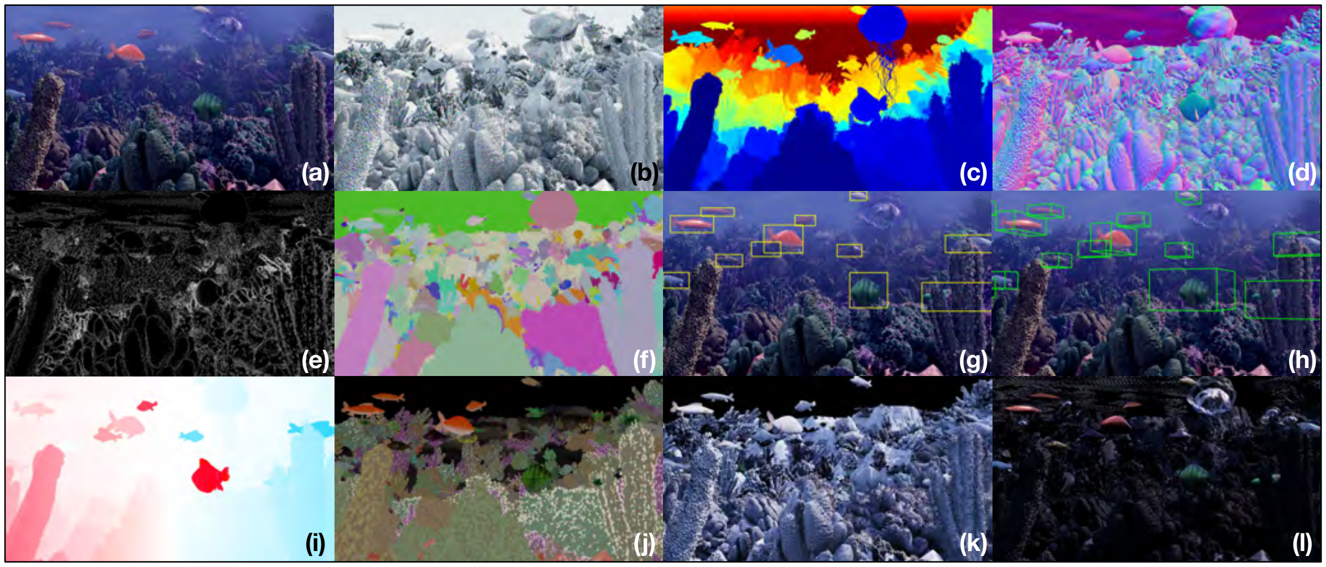
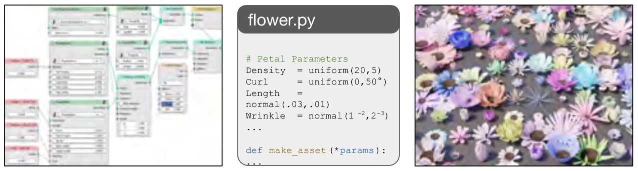


Figure 2. For each image (a), we have a high-res mesh (b), which readily yields Depth (c), Surface Normals (d), Occlusion Boundaries (e), Instance Segmentation masks (f), and 2D / 3D bounding boxes (g/h). From rendering metadata, we obtain Optical Flow (i), and material parameters such as Albedo (j), Lighting Intensity (k) and Specular Reflection (l).

Synthetic Dataset	Domain	# Triangles Per-Scene	# Scenes in Total	# Assets in Total	Free Assets	Procedural Arrangement	Procedural Assets	Provides Procedural Code	External Asset Source
GTA-V [65]	Driving, Urban	-	-	-	No	No	No	N/A	Grand Theft Auto
MOTSynth [22]	Urban	-	-	-	No	No	No	N/A	Grand Theft Auto
MVS-Synth [35]	Driving, Urban	-	-	-	No	No	No	N/A	Grand Theft Auto
DeformingThings4D [48]	Animals/Humanoids	-	2K	2K	Yes	No	No	N/A	Adobe Mixamo [2]
DeepFurniture [53]	Indoor	-	20K	-	No	No	No	N/A	Professional Designers
Robotrix [25]	Indoor	-	16	-	No	No	No	N/A	UE4Arch, UnrealEngine Marketplace [7]
SUNCG [70] (+ [49, 69, 88])	Indoor	-	46K	2.6K	No	No	No	N/A	Planner5D [6]
TartanAir [83]	In/Outdoor, Natural/Urban	-	30	-	No	No	No	N/A	UnrealEngine Marketplace [7]
Hypersim [66]	Indoor	100K-11M	461	59K	No ($6000)	No	No	N/A	Evermotion Architectures [4]
OpenRooms [50]	Indoor	1M	1.3K	3K	No ($500)	No	No	N/A	Scan2CAD [9], ShapeNet [16], Adobe Stock [3]
Sintel [15]	Medieval, Natural	300K	27	-	No ($11)	No	No	N/A	Blender Foundation [17]
Structured3D [89]	Indoor	-	22K	472K	No	No	No	N/A	Professional Designers
SceneNet-RGBD [57]	Indoor	420K	57	5.1K	Yes	No	No	N/A	ShapeNet [16], SceneNet [29]
3D-Front [23]	Indoor	60K	19K	13K	Yes	No	No	N/A	3D-FUTURE [24]
Jiang et al. [39]	Indoor	-	∞	54K	No	Yes	No	No	ShapeNet [16], Planner5D [6]
InteriorNet [47]	Indoor	-	22M	1M	No	Yes	No	No	Manufacturers / Kujiale [5]
FaceSynthetics [84]	Faces	7.4K	-	∞	No	N/A	Partial	No	Artist-Created Faces (textures, hair, clothing)
Meta-Sim2 [19]	Driving, Urban	-	∞	∞	No	Yes	Partial	No	-
Synscapes [80, 85]	Driving, Urban	-	25K	-	No	Yes	Partial	No	7D-Labs [1]
ProcSy [41]	Driving, Urban	-	∞	∞	Yes	Yes	Partial	No	CityEngine, OpenStreetMap [28], Manual Annotation
ProcTHOR [18]	Indoor	-	∞	1.6K	Yes	Yes	No	Yes	AI2-THOR [43], Professional Designers
Kubric [27]	Scattered Objects	161K	∞	52K	Yes	Yes	No	Yes	ShapeNet [16], Google Scanned Objects [21]
Infinigen (Ours)	Natural	Dynamic (16M @ 1080p)	∞	∞	Yes	Yes	Yes	Yes	None

Table 1. Comparison to existing synthetic datasets or generators. Ours is entirely procedural, relying on no external assets, and can produce infinite original assets and scenes. Many existing datasets use external, static asset libraries. Procedural generation is often limited to object placement or a subset of objects. The vast majority of datasets are also restricted to the built environment, especially indoor scenes. In terms of accessibility, many do not provide free assets or make code available. Many works do not report average triangles per scene; where possible, we calculate this using generous assumptions from the numbers they do report. Dashes represent numbers we were not able to obtain or estimate. In counting the number of assets, we exclude trivial modifications like re-lighting and re-scaling.

built environment has been covered by the largest amount of existing work [15, 22, 27, 46, 48, 56, 79, 83] especially indoor scenes [18, 25, 39, 45, 49, 50, 53, 66, 69, 70, 73, 88, 89] and urban scenes [19, 22, 35, 41, 65, 80, 85]. A significant source of synthetic data for the built environment comes from simulated platforms for embodied AI, such as AI2-THOR [43], Habitat [74], BEHAVIOR [72], SAPIEN [86], RLBench [37], CARLA [20]. Some datasets, such as TartanAir [83] and Sintel [15], include a mix of built and natural environments. There also exist datasets such as FlyingThings [56], FallingThings [79] and Kubric [27] that do not render realistic scenes and instead scatter (mostly artificial) objects against simple backgrounds. Synthetic humans are another important application domain, where high-quality synthetic data have been generated for understanding faces [84], pose [8, 81], and activity [40, 71]. Some datasets focus on objects, not full scenes, to serve object-centric tasks such as non-rigid reconstruction [48],



```
# Petal Parameters
Density  = uniform(20,5)
Curl     = uniform(0,50°)
Length   =
normal(.03,.01)
Wrinkle  = normal(1 -2,2-3)
...

def make_asset(*params):
...
```

Figure 3. Our *Node Transpiler* converts artist-friendly Node-Graphs (left) to procedural code (middle) which produces assets (right).

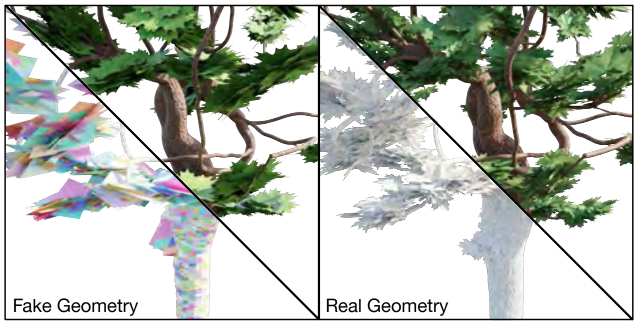


Figure 4. Real-time optimized assets (left) often use low res. geometry in conjunction with shading tricks and alpha-masked image textures to give the illusion of geometric detail. Infinigen assets (right) instead model objects in full geometric detail. Bottom left triangles show a random color per mesh face.

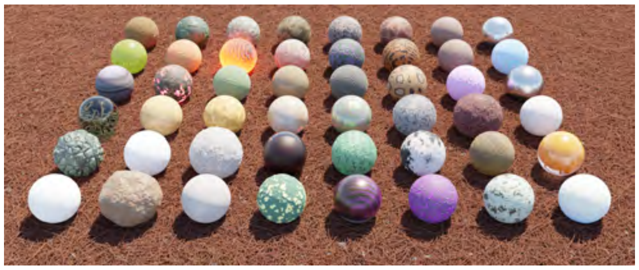


Figure 5. Examples of a subset of our material generators. Columns 1-4 are for terrain, 5-7 are for creatures, and 8 is miscellaneous.

view synthesis [59], and 6D pose [34].

We focus on natural objects and natural scenes, which have had limited coverage in existing work. Even though natural objects do occur in many existing datasets such as urban driving, they are mostly on the periphery and have limited diversity.

Generation Method. Most synthetic datasets are constructed by using a *static* library of 3D assets, either externally sourced or made in house. The downside of a static library is that the synthetic data would be easier to overfit. Procedural generation has been involved in some existing datasets or generators [18, 27, 32, 39, 47], but is limited in scope. Procedural generation is only applied to either object arrangement or a subset of objects, e.g. only buildings and roads but not cars [41, 85]. In contrast, Infinigen is entirely procedural, from shape to texture, from macro structures to micro details, without relying on any external asset.

Asset Type	Num. Generators	Interpretable DOF
Terrain	26	17
Materials	50	271
Weather, Fluid	19	61
Rocks	4	12
Small Plants	30	258
Trees	3	26
Creatures	39	315
Scattering	11	110
Total	182	1070

Table 2. Approximate degrees of freedom, as a proxy of overall diversity. We count only distinct human-understandable parameters with useful ranges of interpolation, with the caveat that this could be an overestimate as not all parameters are fully independent. Some asset classes (e.g terrain) are based on physics simulation and have many more *internal* degrees of freedom not counted here. See the supplement for a full list of named parameters.

Accessibility. A synthetic dataset or generator is most useful if it is maximally accessible, i.e. it provides free access to assets and code with minimum use restrictions. However, few existing works are maximally accessible. Often the rendered images are provided, but underlying 3D assets are unavailable, not free, or have significant use restrictions. Moreover, the code for procedural generation, if any, is often unavailable.

Infinigen is maximally accessible. Its *code* is available under the GPL license. Anyone can freely use Infinigen to generate unlimited assets.

3. Method

Procedural Generation. Procedural generation refers to the creation of data through generalized rules and simulators. Where an artist might manually create the structure of a single tree by eye, a procedural system creates infinite trees by coding their structure and growth in generality. Developing procedural rules is a form of world modeling using compact mathematical language.

Blender Preliminaries. We develop procedural rules primarily using Blender, an open-source 3D modelling software that provides various primitives and utilities. Blender represents scenes as a hierarchy of posed objects. Users modify this representation by transforming objects, adding primitives, and editing meshes. Blender provides import/export for most common 3D file-formats. Finally, all operations in Blender can be automated using its Python API, or by inspecting its open-source code.

For more complex operations, Blender provides an intuitive node-graph interface. Rather than directly edit shader code to define materials, artists edit *Shader Nodes* to compose primitives into a photo-realistic material. Similarly, *Geometry Nodes* define a mesh using nodes representing operators such as Poisson disk sampling, mesh boolean, extrusion etc. A finalized Geometry Node Tree is a generalized

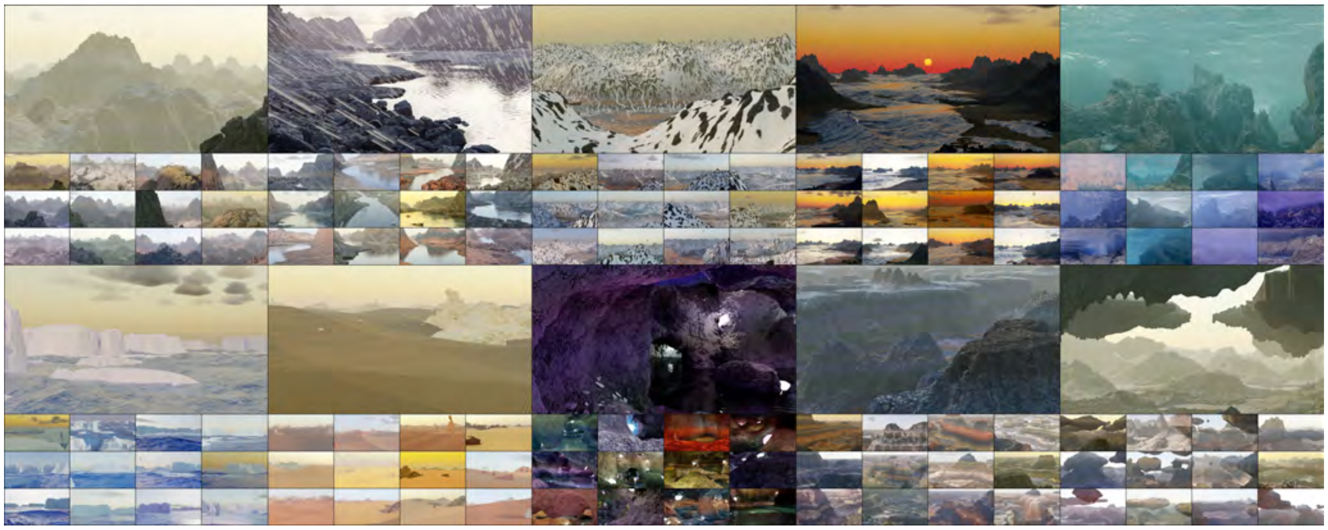


Figure 6. Random, *non cherry-picked* terrain-only scenes. We sample 13 images for various natural scene types. From top left to bottom right; Mountains, Rainy river, Snowy mountains, Coastal sunrise, Underwater, Arctic icebergs, Desert, Caves, Canyons and Floating islands. See the supplement for a larger, higher resolution sample.

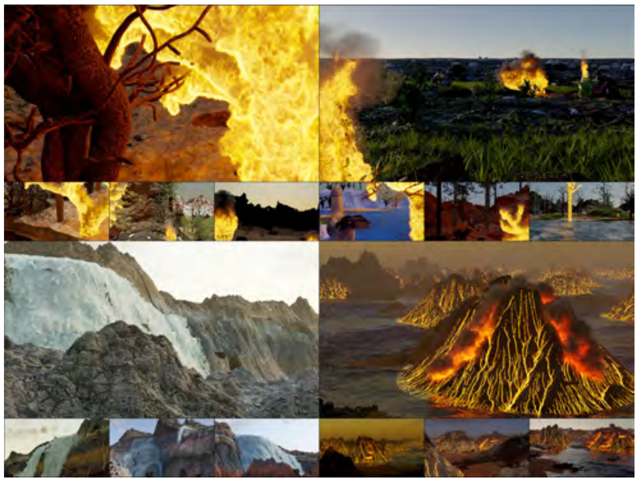


Figure 7. Random, *non cherry-picked* images of simulated fire, smoke, waterfalls, and volcano eruptions.

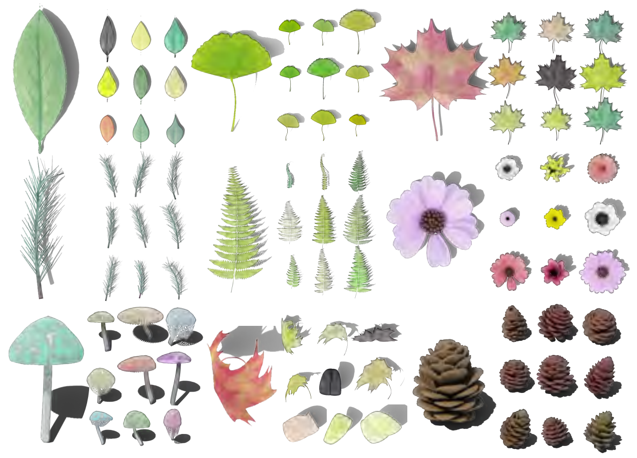


Figure 8. Random, *non cherry-picked* leaves, flowers, mushrooms and pinecones.

parametric CAD model, which produces a unique 3D object for each combination of its input parameters. These tools are intuitive and widely adopted by 3D artists.

Although we use Blender heavily, not all of our procedural modeling is done using node-graphs; a significant portion of our procedural generation is done outside Blender and only loosely interacts with Blender.

Node Transpiler. As part of Infinigen, we develop a suite of new tools to speed up our procedural modeling. A notable example is our *Node Transpiler*, which automates the process of converting node-graphs to Python code, as shown in Fig. 3. The resulting code is more general, and allows us to randomize graph *structure* not just input parameters. This tool makes node-graphs more expressive and allows easy integration with other procedural rules developed directly in Python or C++. It also allows non-programmers to contribute Python code to Infinigen by making node-graphs. See the supplement for more details.

***Generator* Subsystems**. Infinigen is organized into *generators*, which are probabilistic programs each specialized to produce one subclass of assets (e.g. mountains or fish). Each has a set of high-level parameters (e.g. the overall height of a mountain), which reflect the external degrees of freedom controllable by the user. By default, we randomly sample these parameters according to distributions tuned to mirror the natural world, with no input from the user. However, users can also override any parameter using our Python API to achieve fine grained control of data generation.

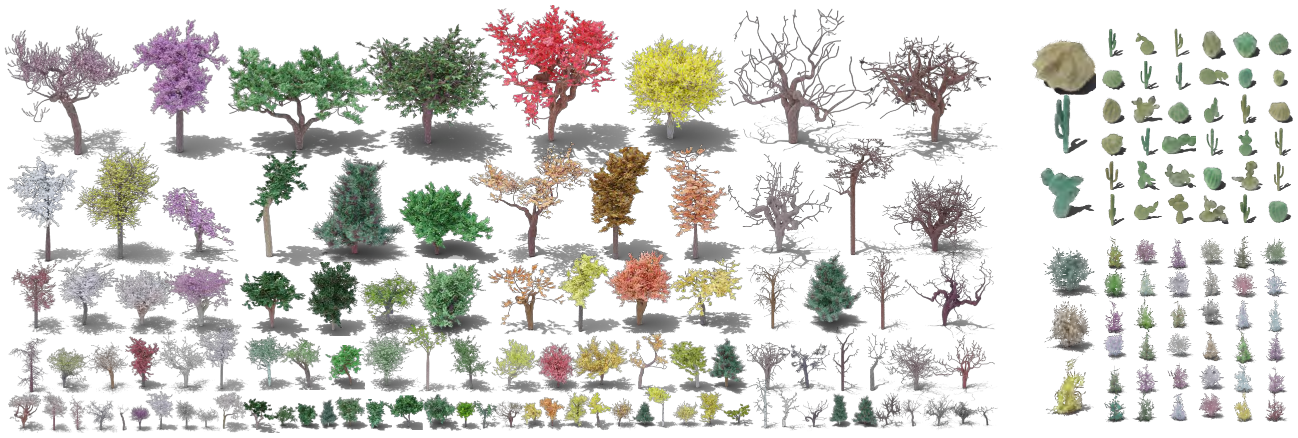


Figure 9. Random, *non cherry-picked* procedural trees (left), cacti (top right) and bushes (bottom right).

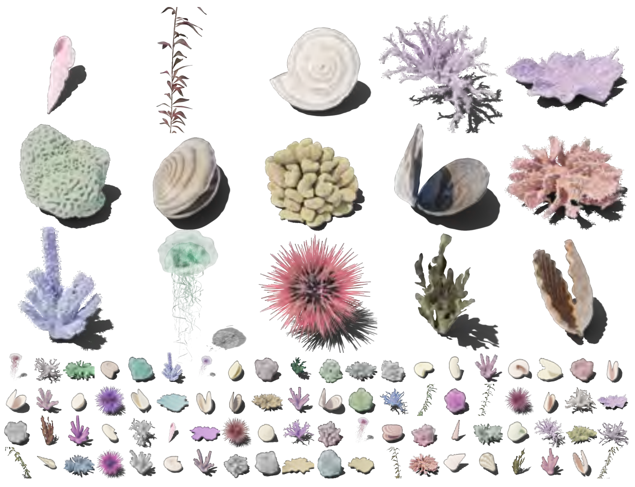


Figure 10. Random, *non cherry-picked* underwater objects.

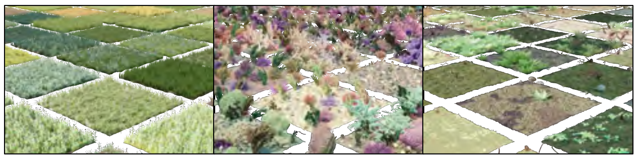


Figure 11. Random, *non cherry-picked* surface scatters. Dense coverage with procedural assets turns any surface into a convincing *grassland*, *sea floor* or *forest floor* environment.

Each probabilistic program involves many additional internal, low-level degrees of freedom (e.g. the heights of every point on a mountain). Randomizing over both the internal and external degrees of freedom leads to a distribution of assets which we sample from for unlimited generation. Tab. 2 summarizes the number of human-interpretable degrees of freedom in Infinigen, with the caveat that the numbers could be an over-estimation because not all parameters are fully independent. Note that it is hard to quantify the internal degrees of freedom, so the external degrees of freedom serve as a lower bound of the total degrees of freedom for our system.

Material Generators. We provide 50 procedural material generators (Fig. 5). Each is composed of a randomized shader, specifying color and reflectance, and a local geometry generator, which generates corresponding fine geometric details.

The ability to produce accurate ground-truth geometry is a key feature of our system. This precludes the use of many common graphics techniques such as Bump Mapping and Phong Interpolation [13, 64]. Both manipulate face normals to give the illusion of detailed geometric textures, but do so in a way that cannot be represented as a mesh. Similarly, artists often rely on image textures or alpha channel masking to give the illusion of high res. meshes where none exist. All such shortcuts are excluded from our system. See Fig. 4 for an illustrative example of this distinction.

Terrain Generators. We generate terrain (Fig. 6) using SDF elements derived from fractal noise [63] and simulators [12, 33, 36, 58, 63]. We evaluate these to a mesh using marching cubes [54]. We generate boulders via repeated extrusion, and small stones using Blender's built-in addon. We simulate dynamic fluids (Fig. 7) using FLIP [14], sun/sky light using the Nishita sky model [61], and weather with Blender's particle system.

Plants & Underwater Object Generators. We model tree growth with random walks and space colonization [67], resulting in a system with diverse coverage of various trees, bushes and even some cacti (Fig. 9). We provide generators for a variety of corals (Fig. 10) using Differential Growth [62], Laplacian Growth [42], and Reaction-Diffusion [26]. We produce Leaves (Fig. 8), Flowers [38], Seaweed, Kelp, Mollusks and Jellyfish using geometry node-graphs.

Surface Scatter Generators. Some natural environments are characterized by a dense coverage of smaller objects. To this end, we provide several scatter generators, which combine one or more existing assets in a dense layer (Fig. 11). In the forest floor example, we generate fallen tree logs

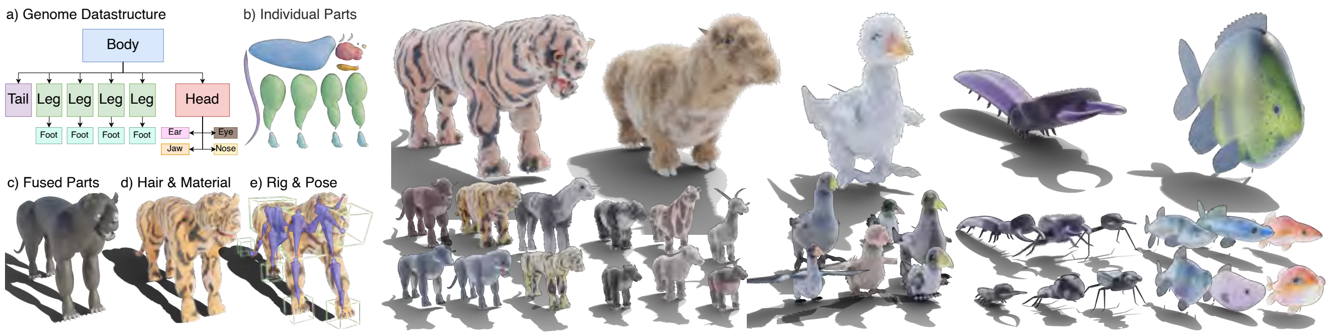


Figure 12. Creature Generation. Our system automatically generates genomes (a), parts (b), assembly (c), materials (d) and animation rigs (e). On the right, we show random, *non cherry-picked* samples from our *Carnivore*, *Herbivore*, *Bird*, *Beetle*, and *Fish* generators.

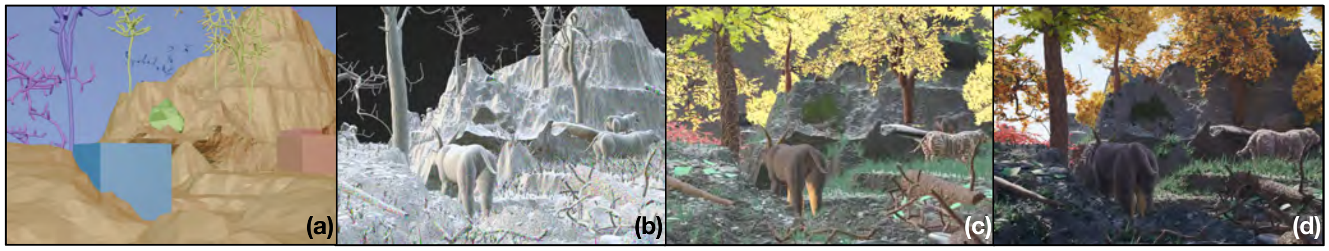


Figure 13. Data Generation Pipeline. We procedurally compose a scene layout (a) with random camera poses. We generate all necessary assets (b, showing a color per mesh face), and apply procedural materials and displacement (c). Finally, we render a photo-real image (d).

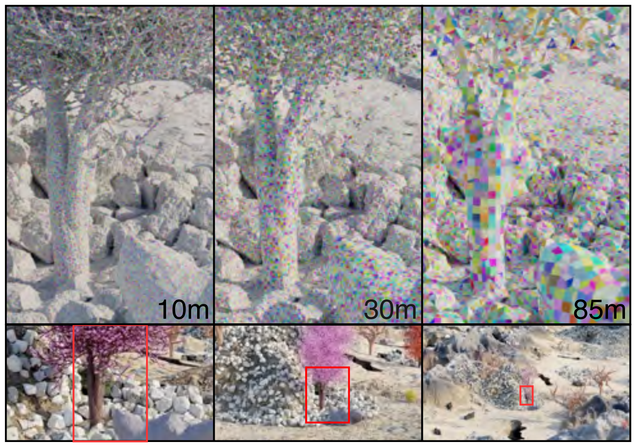


Figure 14. Dynamic Resolution Scaling. We show close-up mesh visualizations (top) of the same content for three different camera distances. Despite differing mesh resolution, no changes are visible in the final images (bottom).

by procedurally fracturing entire trees from our tree system.

Due to space constraints, all specific implementation details of the above are available in the supplement.

Creature Generators. The genome of each creature is represented as a tree data-structure (Fig. 12 a). This reflects the topology of real creatures, whose limbs don't form closed loops. Nodes contain part parameters, and edges specify part attachment. We provide generators for 5 classes of realistic creature genomes, shown in Fig. 12. We can also combine creature parts at random, or interpolate similar genomes. See the supplement for details.

Each part generator is either a transpiled node-graph, or a non-uniform rational basis spline (NURBS). NURBS parameter-space is high-dimensional, so we randomize NURBS parameters under a factorization inspired by lofting, composed of deviations from a center curve. To tune the random distribution, we modelled 30 example heads and bodies, and ensured that our distribution supports them.

Our system produces high-quality animation rigs, and optionally simulates realistic surface folding, sagging and motion of creature skin using cloth simulation. For hair, we use the transpiler to automate the process of grooming hairs, as usually performed by human character artists.

Dynamic Resolution Scaling. With the camera location fixed, we evaluate our procedural assets at precisely the level of detail such that each face is < 1px in size when rendered. This process is visualized in Fig. 14. For most assets, this entails evaluating a parametric curve at the given pixel size, or using Blender's built-in subdivision or re-meshing. For terrain, we perform Marching Cubes on SDF points in *spherical coordinates*. For densely scattered assets (incl. all assets in Fig. 11) we use *instancing* - that is, we generate a fixed number of assets of each type, and reuse them with random transforms within a scene. Even with this effort in optimization, the average complete scene has 16M polygons.

Image Rendering & Ground Truth Extraction. We render images using Cycles, Blender's physically-based path

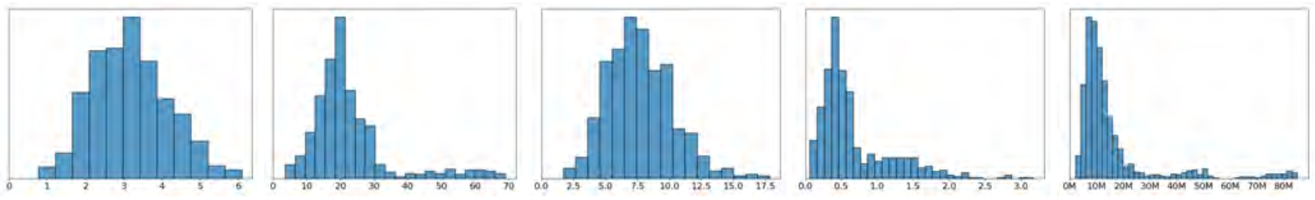

(a) Wall Time (Hours) (b) Memory (GB) (c) CPU Hours (d) GPU Hours (e) # Triangles per scene

Figure 15. Resource requirements to create a pair of stereo 1080p images using Infinigen.

Training Dataset	Adirondack	**Jadeplant**	Motorcycle	Piano	Pipes	Playroom	Playtable	Recycle	Shelves	Vintage	Avg
FallingThings [79]	8.3	43.3	**12.3**	18.2	25.3	29.7	50.0	10.4	43.3	45.6	28.6
Sintel-Stereo [15]	35.7	62.9	31.1	24.1	31.9	41.7	60.1	30.8	55.8	76.1	45.0
HR-VS [87]	43.5	43.2	17.0	29.6	32.1	34.6	68.4	24.7	57.4	34.9	38.5
Li et al. [46]	23.9	80.2	40.7	32.0	40.3	49.1	67.5	36.6	51.7	42.3	46.4
SceneFlow [56]	**7.4**	41.3	14.9	16.2	33.3	**18.8**	**38.6**	**10.2**	**39.1**	29.9	**25.0**
TartanAir [83]	15.5	45.1	18.1	**12.9**	28.4	25.6	51.0	20.9	49.1	**28.2**	29.5
InStereo2K [11]	17.1	59.7	21.3	23.8	35.8	33.9	**36.4**	20.0	**33.4**	44.1	32.5
Ours (Infinigen 30K)	**7.4**	**35.2**	15.2	20.7	**24.7**	29.3	50.0	12.6	55.1	46.9	29.7

Table 3. Bad 3.0 (%) ↓ error on the Middlebury [68] validation set. Infinigen generalizes well to natural objects (e.g. Jadeplant). However, natural objects contain very few planar or textureless surfaces; models trained exclusively on natural objects generalize less well on Middlebury's indoor scenes.

tracing renderer. We provide code to extract ground truth for common tasks, visualized in Fig. 2.

Cycles individually traces photons of light to accurately simulate diffuse and specular reflection, transparent refraction and volumetric effects. We render at 1920×1080 resolution using $10,000$ random samples per-pixel, which is standard for blender artists and ensures almost no sampling noise in the final image.

Prior datasets [15, 27, 30, 32, 46] rely on blender's built-in render-passes to obtain dense ground truth. However, these rendering passes are a byproduct of the rendering pipeline and not intended for training neural networks. Specifically, they are often incorrect due to translucent surfaces, volumetric effects, motion blur, focus blur or sampling noise. See the supplement for examples of these issues.

Instead, we provide OpenGL code to extract surface normals, depth and occlusion boundaries from the mesh directly without relying on blender. This solution has many benefits in addition to its accuracy. Users can exclude objects not relevant to their task (e.g. water, clouds, or any other object) independently of whether they are rendered. Many annotations like occlusion boundaries are also plainly not supported by Blender. Finally, our implementation is modular, and we anticipate that users will generate task-specific ground truth not covered above via simple extensions to our codebase.

Runtime. We benchmark our Infinigen on 2 *Intel(R) Xeon(R) Silver 4114 @ 2.20GHz* CPUs and 1 NVidia-GPU across 1000 independent trials. The wall time to produce a pair of 1080p images is 3.5 hours. Statistics are shown in Fig. 15.

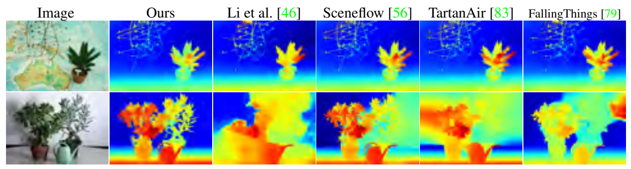


Figure 16. Qualitative results on the *Plant* and *Australia* Middlebury [68] test images. RAFT-Stereo trained using Infinigen generalizes well to images with natural objects.

4. Experiments

To evaluate Infinigen, we produced 30K image pairs with ground truth for rectified stereo matching. We train RAFT-Stereo [51] on these images from scratch and compare results on the Middlebury validation (Tab. 3) and test sets (Fig. 16). See the supplement for high resolution qualitative results on in-the-wild natural photographs.

5. Contributions & Acknowledgements

Alexander Raistrick, Lahav Lipson and Zeyu Ma contributed equally, and are ordered alphabetically by first name. Each has the right to list their name first in their CV. Alexander Raistrick performed team coordination, and developed the creature system, transpiler and scene composition. Lahav Lipson trained models and implemented dense annotations and rendering. Zeyu Ma developed the terrain system and camera selection. Lingjie Mei created coral, sea invertebrates, small plants, boulders & moss. Mingzhe Wang developed materials for creatures, plants and terrain. Yiming Zuo developed trees and leaves. Karhan Kayan created liquids and fire. Hongyu Wen & Yihan Wang developed creature parts and materials. Beining Han created ferns and small plants. Alejandro Newell designed the tree & bush system. Hei Law created weather and clouds. Ankit Goyal developed terrain materials. Kaiyu Yang developed an initial prototype with randomized shapes. Jia Deng conceptualized the project, led the team, and set the directions.

We thank Zachary Teed for helping with the early prototypes. This work was partially supported by the Office of Naval Research under Grant N00014-20-1-2634 and the National Science Foundation under Award IIS-1942981.

References

[1] 7d labs. https://www.7dlabs.com/. 3

[2] Adobe mixamo. https://www.mixamo.com. 3

[3] Adobe stock. https://stock.adobe.com/3dassets. 3

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

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

[6] Planner5d. https://planner5d.com/. 3

[7] Unreal engine marketplace. https://www.unrealengine.com/marketplace/en-US/store. 3

[8] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3686–3693, 2014. 3

[9] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2CAD: Learning CAD model alignment in RGB-D scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[10] Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 620–630, 2022. 1

[11] Wei Bao, Wei Wang, Yuhua Xu, Yulan Guo, Siyu Hong, and Xiaohu Zhang. Instereo2k: A large real dataset for stereo matching in indoor scenes. *Science China Information Sciences*, 63(11):1–11, 2020. 8

[12] Katherine R Barnhart, Eric WH Hutton, Gregory E Tucker, Nicole M Gasparini, Erkan Istanbulluoglu, Daniel EJ Hobley, Nathan J Lyons, Margaux Mouchene, Sai Siddhartha Nudurupati, Jordan M Adams, et al. Landlab v2. 0: a software package for earth surface dynamics. *Earth Surface Dynamics*, 8(2):379–397, 2020. 6

[13] James F. Blinn. Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics*, 12(3):286–292, aug 1978. 6

[14] J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38, Jan. 1988. 6

[15] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, pages 611–625, 2012. 3, 8

[16] 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

[17] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 1, 2, 3

[18] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, et al. ProcTHOR: Large-scale embodied AI using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022. 1, 3, 4

[19] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-Sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision (ECCV)*, pages 715–733. Springer, 2020. 3

[20] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, pages 1–16, 2017. 3

[21] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google Scanned Objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022. 3

[22] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Orcun Cetintas, Riccardo Gasparini, Aljoša Ošep, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. MOTSynth: How can synthetic data help pedestrian detection and tracking? In *International Conference on Computer Vision (ICCV)*, 2021. 3

[23] 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 *International Conference on Computer Vision (ICCV)*, pages 10933–10942, 2021. 3

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

[25] 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 *International Conference on Intelligent Robots and Systems (IROS)*, 2018. 1, 3

[26] Peter Gray and Stephen K. Scott. Chemical oscillations and instabilities: Non-linear chemical kinetics. 1990. 6

[27] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 4, 8

[28] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008. 3

[29] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4077–4085, 2016. 3

[30] Yana Hasson, Gül Varol, Dimitris Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 8

[31] 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

[32] Ju He, Enyu Zhou, Liusheng Sun, Fei Lei, Chenyang Liu, and Wenxiu Sun. Semi-synthesis: A fast way to produce effective datasets for stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 4, 8

[33] Daniel EJ Hobley, Jordan M Adams, Sai Siddhartha Nudurupati, Eric WH Hutton, Nicole M Gasparini, Erkan Istanbulluoglu, and Gregory E Tucker. Creative computing with landlab: an open-source toolkit for building, coupling, and exploring two-dimensional numerical models of earth-surface dynamics. *Earth Surface Dynamics*, 5(1):21–46, 2017. 6

[34] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6D object localization. *European Conference on Computer Vision Workshops (ECCVW)*, 2020. 4

[35] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2821–2830, 2018. 1, 3

[36] Eric Hutton, Katy Barnhart, Dan Hobley, Greg Tucker, Sai Nudurupati, Jordan Adams, Nicole Gasparini, Charlie Shobe, Ronda Strauch, Jenny Knuth, Margaux Mouchene, Nathan Lyons, David Litwin, Rachel Glade, Giuseppecipolla95, Amanda Manaster, Langston Abby, Kristen Thyng, and Francis Rengers. landlab, 4 2020. 6

[37] 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. 3

[38] Roger V Jean. Introductory review: Mathematical modeling in phyllotaxis: The state of the art. *Mathematical Biosciences*, 64(1):1–27, 1983. 6

[39] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3D scene synthesis and 2D image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision (IJCV)*, 126(9):920–941, 2018. 3, 4

[40] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 3

[41] Samin Khan, Puu Phan, Rick Salay, and Krzysztof Czarnecki. ProcSy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *CVPR Workshops*, 2019. 3, 4

[42] Ryo Kobayashi. Modeling and numerical simulations of dendritic crystal growth. *Physica D: Nonlinear Phenomena*, 63:410–423, 3 1993. 6

[43] 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. 3

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

[45] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. IGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021. 3

[46] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3, 8

[47] 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, 4

[48] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4DComplete: Non-rigid motion estimation beyond the observable surface. In *International Conference on Computer Vision (ICCV)*, pages 12706–12716, 2021. 3

[49] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proceedings of the European conference on computer vision (ECCV)*, pages 371–387, 2018. 3

[50] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, et al. OpenRooms: An open framework for photorealistic indoor scene datasets. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[51] 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, 8

[52] 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

[53] 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. 1, 2, 3

[54] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. 6

[55] 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

[56] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 3, 8

[57] 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? In *International Conference on Computer Vision (ICCV)*, pages 2678–2687, 2017. 3

[58] Nick McDonald. Soilmachine. `https://github.com/weigert/SoilMachine`, 2022. 6

[59] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 4

[60] Sergey I Nikolenko. *Synthetic data for deep learning*, volume 174. Springer, 2021. 2

[61] Tomoyuki Nishita, Takao Sirai, Katsumi Tadamura, and Eihachiro Nakamae. Display of the earth taking into account atmospheric scattering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 175–182, New York, NY, USA, 1993. Association for Computing Machinery. 6

[62] Boris Okunskiy. Introducing differential growth addon for blender. `https://boris.okunskiy.name/posts/blender-differential-growth`. 6

[63] Jordan Peck. Fastnoise lite. `https://github.com/Auburn/FastNoiseLite`, 2022. 6

[64] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, jun 1975. 6

[65] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, pages 102–118. Springer, 2016. 1, 2, 3

[66] 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 *International Conference on Computer Vision (ICCV)*, 2021. 3

[67] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. *NPH*, 7(63-70):6, 2007. 6

[68] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, 2014. 8

[69] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *International Conference on Computer Vision (ICCV)*, pages 8598–8607, 2019. 3

[70] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[71] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 3

[72] 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. In *Conference on Robot Learning (CoRL)*, pages 477–490, 2022. 3

[73] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 3

[74] 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 *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 3

[75] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 402–419. Springer, 2020. 1

[76] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1

[77] 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. 1

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

[79] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling Things: A synthetic dataset for 3D object detection and pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 8

[80] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. *arXiv preprint arXiv:1710.06270*, 2017. 3

[81] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3D human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 3

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

[83] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A dataset to push the limits of visual SLAM. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020. 3, 8

[84] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: face analysis in the wild using synthetic data alone. In *International Conference on Computer Vision (ICCV)*, pages 3681–3691, 2021. 3

[85] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018. 3, 4

[86] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[87] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 8

[88] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5287–5295, 2017. 3

[89] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A large photo-realistic dataset for structured 3D modeling. In *European Conference on Computer Vision (ECCV)*, 2020. 3