# Kubric: A scalable dataset generator

Klaus Greff[1] Francois Belletti[1] Lucas Beyer[1] Carl Doersch[6] Yilun Du[5]
Daniel Duckworth[1] David J Fleet[1,2] Dan Gnanapragasam[1] Florian Golemo[4, 9]
Charles Herrmann[1] Thomas Kipf[1] Abhijit Kundu[1] Dmitry Lagun[1] Issam Laradji[3, 9]
Hsueh-Ti (Derek) Liu [2] Henning Meyer[1] Yishu Miao[10] Derek Nowrouzezahrai[3,4]
Cengiz Oztireli[1,8] Etienne Pot[1] Noha Radwan[1] Daniel Rebain[1,7] Sara Sabour[1,2]
Mehdi S. M. Sajjadi[1] Matan Sela[1] Vincent Sitzmann[5] Austin Stone[1] Deqing Sun[1]
Suhani Vora[1] Ziyu Wang[10] Tianhao Wu[8] Kwang Moo Yi[7] Fangcheng Zhong[8]
Andrea Tagliasacchi[1,2,11]

[1]Google [2]University of Toronto [3]McGill University [4]Mila [5]MIT [6]DeepMind
[7]UBC [8]University of Cambridge [9]ServiceNow [10]Haiper [11]Simon Fraser University

## Abstract

*Data is the driving force of machine learning, with the amount and quality of training data often being more important for the performance of a system than architecture and training details. But collecting, processing and annotating real data at scale is difficult, expensive, and frequently raises additional privacy, fairness and legal concerns. Synthetic data is a powerful tool with the potential to address these shortcomings: 1) it is cheap 2) supports rich ground-truth annotations 3) offers full control over data and 4) can circumvent or mitigate problems regarding bias, privacy and licensing. Unfortunately, software tools for effective data generation are less mature than those for architecture design and training, which leads to fragmented generation efforts. To address these problems we introduce* Kubric, *an open-source Python framework that interfaces with PyBullet and Blender to generate photo-realistic scenes, with rich annotations, and seamlessly scales to large jobs distributed over thousands of machines, and generating TBs of data. We demonstrate the effectiveness of Kubric by presenting a series of 13 different generated datasets for tasks ranging from studying 3D NeRF models to optical flow estimation. We release Kubric, the used assets, all of the generation code, as well as the rendered datasets for reuse and modification.*

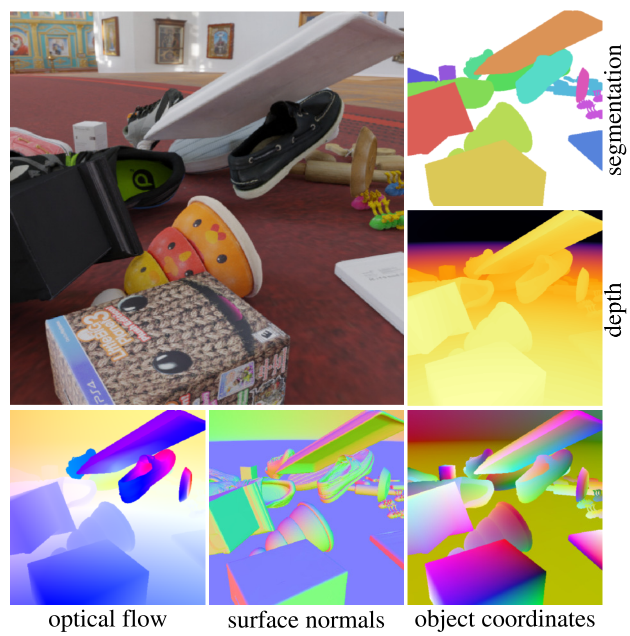


Figure 1. Example scene created and rendered with Kubric along with some of the automatically generated annotations.

## 1. Introduction

High quality data – at scale – is essential for deep learning. It is arguably as or more important than many architectural and training details. Nevertheless, even for many straightforward vision tasks, collecting and curating sufficient amounts of data continues to be a daunting challenge. Some of the key barriers include the expense of high quality, detailed annotations, data diversity, control over task domain complexity, as well as concerns over privacy, fairness and licensing [4]. This paper advocates the use of synthetic data to circumvent many of these problems, for which we introduce Kubric, an open-source pipeline for generating realistic image and video data with rich ground truth annotations for myriad vision tasks.

Synthetic data has long been used for benchmark evaluation (e.g. for optical flow [6, 7]), as it supports rich ground-truth annotations, and fine-grained control over data complexity. It also enables systematic model evaluation under violations of model assumptions (e.g. rigidity). Synthetic data has also been used effectively for training. This includes seminal work on 3D human pose estimation from RGBD [85], and more recently on myriad tasks, including facial landmark detection [103], human pose from video [24], and semantic segmentation [116]. Photo-realism is often considered essential for narrowing the generalization gap, but even without perfect realism, synthetic data can be remarkably effective (e.g., flying chairs [26], MPI-Sintel [13] and more recently AutoFlow [89]).

Unfortunately, effective software tools for data generation are less mature than those for architecture design and training. It is therefore not surprising that most generation efforts, although costly, have been one-off and task specific. Although challenging to design and develop, what is needed is a general framework for photo-realistic generation that supports reuse, replication, and shared assets, all at scale, enabling workflows with large jobs concurrently on thousands of machines. Kubric addresses these issues with coherent framework, a simple Python API, and a full set of tools for generation at scale, integrating assets from multiple sources, with a common export data format for porting data into training pipelines, and with rich annotations for myriad vision tasks. In summary, our key contributions are:

- We introduce Kubric[1], a framework for generating photo-realistic synthetic datasets for myriad vision tasks, with fine-grain control over data complexity and rich ground truth annotations.
- Kubric enables generation at scale, seamlessly running large jobs over thousands of machines, generating TBs of data in a standard export data format.
- The versatility of Kubric is demonstrated by the creation of 13 datasets for new vision challenge problems, spanning 3D NeRF models to optical flow estimation, along with benchmark results.

[1] Source code and datasets are available at https://github.com/google-research/kubric.

## 2. Related Work

Synthetic data provides high quality labels for many image tasks such as semantic [16] and instance [102] segmentation, text localization [37], object detection [40], and classification [32]. There are many large synthetic datasets such as CLEVR [44], ScanNet [21], SceneNet RGB-D [65], NYU v2 [67], SYNTHIA [80], virtual KITTI [33], and flying things 3D [64] for specific tasks. However, these datasets rarely contain all possible annotations for all image tasks, lacking key signals such as camera pose, instance or semantic segmentation masks, or optical flow. This is particularly challenging for multi-task problems like co-training a neural scene model with semantic segmentation [118]. Moreover, fixed datasets can introduce biases [94, 95] such as an object-centric bias [71] and photographer's bias [5]. By contrast, Kubric automatically generates the image cues for each frame and easily support a variety of viewing angles and lighting conditions.

**Specialized Synthetic Data Pipelines**. There are many hand-crafted pipelines for synthetic data generation [37, 49, 66] built off of rendering engines like Blender [9] and Unity3D [11]. While these pipelines mitigate biases in viewing angles and lighting, they are often specialized for a *particular* task. This makes it challenging to adapt them to provide additional annotations without in-depth knowledge of the underlying rendering engine. Real world to sim pipelines capture real-world data via 3D scans and then convert them into a synthetic data format. [56] creates high quality room scenes, but has many manual steps including pose alignment and material assignment. [27] also utilizes 3D scans, and provides control over a wide range of scene parameters, including camera position, field of view, and lighting, as well as a number of per frame image cues. While these approaches produce high quality data for a particular captured scene, the pipeline still relies on 3D scans of the full scene, which imposes a bottleneck for scaling.

| Name | Rendering | GI | Physics | Scaling | DL |
|---|---|---|---|---|---|
| Playing4Data [78] | (Game) | × | (Game) | × | × |
| UnrealCV [75] | UE4 | × | UE4 | ✓ | × |
| TDW [34] | Unity | × | PhysX | ✓ | × |
| iGibson [106] | PyRender | × | PyBullet | ✓ | × |
| Habitat [91] | Magnum | × | Bullet | ✓ | × |
| OpenRooms [56] | OptiX | ✓ | – | × | × |
| Omnidata [27] | Blender | ✓ | – | × | ✓ |
| Blenderproc [23] | Blender | ✓ | Bullet | × | × |
| Kubric | Blender | ✓ | PyBullet | ✓ | ✓ |

Table 1. Rendering: Blender any OptiX are ray tracing engines, all others are based on rasterization; GI: support for global illumination; Physics: engine for physics simulation; Scaling: Easy to scale to very large datasets. DL: Data-loader integration with machine learning frameworks (PyTorch/TF).

**Generic Dataset Creation Pipelines**. General purpose synthetic data pipelines (like Kubric) aim to address these issues by supporting arbitrary random compositions of meshes, textures, pre-existing scenes, etc. from collections of 3D assets. This mitigates some of the scaling considerations of real world to sim pipelines and more easily supports composition of assets from different datasets. These pipelines differ along various dimensions (see Tab. 1). One important differences is the use of rendering engine, where ray-tracing engines support global illumination and other advanced lighting effects, which allow for a higher degree of realism than rasterization engines at the cost of a higher computational demand. Most general purpose synthetic data generation pipelines such as [53, 83, 84, 93, 106] are build on rasterization, which makes them very fast, often to the point of generating entire datasets on a single GPU machine. ThreeDWorld [34] is an excellent example for such an engine with a flexible Python API, comprehensive export capabilities to a Unity3D based rasterization engine, the NVidia Flex [61] physics simulator and even sound generation via PyImpact [96]. The framework closest in scope to Kubric is BlenderProc [23]: a ray-tracing based pipeline built on Blender, which supports the generation of high-quality renders and comprehensive annotations as well as rigid-body physics simulations. The main differences lie in Kubric's focus on scaling workloads to many workers, and its integration with tensorflow datasets.

# 3. Infrastructure

Kubric is a high-level python library that acts as glue between: a rendering engine, a physics simulator, and data export infrastructure; see Figure 2. Its main contribution is to streamline the process and reduce the hurdle and friction for researchers that want to generate and share synthetic data.

## 3.1. Design Principles

**Openness**. Data-generation code should be freely usable by researchers both in academia and in industry. Kubric addresses this by being open-source with an Apache2 licence, and by only using software with similarly permissive licenses. Together with the use of free 3D assets and textures, this enables researchers to share not just the data, but also enable others to reproduce and modify it.

**Ease of use**. The fragmentation of computer graphics formats, conventions and interfaces is a major pain point for setting up and reusing data-generation code. Kubric minimizes this friction by offering a simple object-oriented API interface with PyBullet and Blender behind the scenes, hiding the complexities of setup, data transfer, and keeping them in sync. We also provide pre-processed 3D assets from a variety of data sources, that can be used with minimal effort.

**Realism**. To be maximally useful, a data-generator should

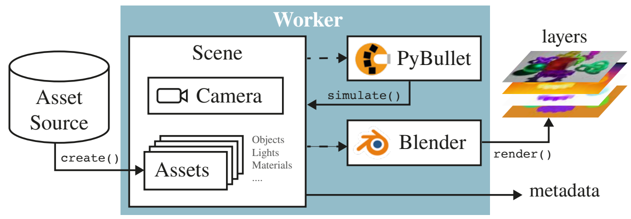


Figure 2. *Overview* – a typical Kubric worker randomly populates a scene with assets loaded from an external source, possibly runs a physics simulation, renders the resulting frames, and finally exports the images, annotation layers, and other metadata.

be able to model as much as possible of the structure and complexity of real data. The Cycles raytracing engine of Blender supports a high level of realism and can model complex visual phenomena such as reflection, refraction, indirect illumination, subsurface-scattering, motion-blur, depth of field, etc. Studying these effects is important, and they also help to reduce the generalization gap.

**Scalability**. Data generation workloads can range from simple toy-data prototyping all the way to generating massive amounts of high-resolution video data. To support this entire range of usecases, Kubric is designed to seamlessly scale from a local workflow to running large jobs on thousands of machines in the cloud.

**Portable and Reproducible**. To facilitate reuse of data-generation code, it is important that the pipeline is easy to setup and produces the same results — even when executed on different machines. This is especially important due to the difficulty in installing the Blender Python module [31] and the substantial variations between versions. By distributing a Kubric Docker image, we ensure portability and remove the bulk of the installation pain.

**Data Export**. Kubric by default exports a rich set of ground truth annotations from segmentation, optical flow, surface normals and depth maps, to object trajectories, collision events and camera parameters. We also introduce SunDs (see Sec. 3.4), a unified multi-task frontend for richly annotated scene-based data.

## 3.2. Kubric Worker – Figure 3

The typical Kubric workflow consists of writing a worker script that creates, simulates, and renders a single random scene. The full dataset is then generated by running this worker many times, and afterwards collecting the generated data. This division into independent scenes mirrors the i.i.d. structure of most datasets and supports scaling the generation process from local prototyping to a large number of parallel jobs; e.g. using the Google Cloud Platform (GCP), for which we provide convenience launcher scripts. We also plan to support an Apache Beam pipeline that combines generation, collection and post-processing of datasets into a single convenient (but ultimately harder to debug) workflow.

```python
import numpy as np
import kubric as kb
asset_source = kb.AssetSource.from_manifest("KuBasic.json")
scene = kb.Scene(resolution=(640, 640))
renderer = kb.renderer.Blender(scene)
simulator = kb.simulator.PyBullet(scene)
# --- populate the scene
scene.camera = kb.PerspectiveCamera(position=(0, 5, 5),
                                    look_at=(0, 0, 0))
scene += kb.Cube(name="floor", scale=(10, 10, .1),
                 position=(0, 0, -0.1), static=True)
scene += kb.PointLight(position=(-2.5, -1, 5), intensity=300)
rng = np.random.RandomState(seed=42)
for i in range(8): # place 8 random objects within a region
  mat = kb.PrincipledBSDFMaterial(color=kb.random_hue_color(rng=rng),
                                  metallic=rng.choice([0.0, 1.0]),
                                  transmission=rng.choice([0.0, 1.0]))
  obj = asset_source.create(rng.choice(asset_source.all_asset_ids),
                            material=mat, velocity=rng.normal(size=3))
  scene += obj
  kb.move_until_no_overlap(obj, simulator, rng=rng,
                           spawn_region=[[-1, -1, 0], [1, 1, 1]])
# --- execute the simulation, render, and save data to files
simulator.run()
renderer.save_state("output/scene.blend")
frames_dict = renderer.render()
kb.write_image_dict(frames_dict, "output")
```

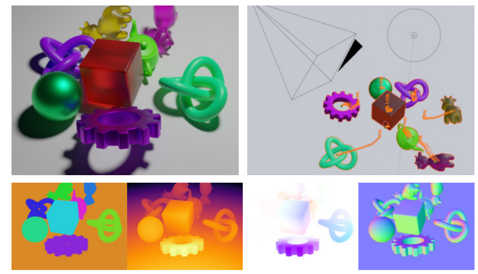

Figure 3. **Example worker** – A simple environment with a floor, a point light, a perspective camera and eight KuBasic objects placed without overlap (by rejection sampling) and a random velocity. `PyBullet` then simulates the physics, `Blender` renders the video. Infinite random variations of the scene can be generated by varying the random seed (rng), and the result can be inspected in Blender *even before* rendering (top right). The exported data includes annotations such as segmentation, depth, flow, and normals.

**Scene structure**. Each worker sets up a `Scene` object, which keeps track of global settings (e.g., resolution, number of frames to render, gravity), a `Camera`, and all the objects, including lights, materials, animations, etc., which we refer to collectively as `Assets`. They are the main abstractions used in Kubric to control the content of a scene, and they each expose a set of properties such as position, velocity or color. When an `Asset` is added to the scene, the corresponding objects are created in each of the `Views`. Currently this comprises the `PyBullet` simulator and the `Blender` renderer, but Kubric can be extended to support other views (e.g., the recently open-sourced MuJoCo). Kubric also maintains a link with the resulting data structure, and automatically communicates all changes of the assets to the connected views. That way, the user only has to work with the abstractions provided by Kubric, and does not have to worry about differences in interfaces or conventions.

**Simulator**. For physics simulation we interface with the open-source PyBullet physics engine [18] that is widely used in robotics (e.g., [43, 46, 107]). It can be used to populate the scene with non-overlapping objects, or to run a (rigid-body) simulation, and to convert the resulting trajectories into keyframes and collision events. Bullet can also handle rigged models, softbody simulations, and various constraints that Kubric does not yet support.

**Renderer**. Kubric uses the `bpy` module as an interface to Blender, a powerful open-source 3D computer graphics suite which is widely used for game development and visual effects. Blender comes with a powerful UI that can be used for interactively debugging and adjusting scenes, as well as creating and exporting new assets. For rendering we rely on cycles – Blender's raytracing engine – which, unlike rasterized rendering engines, supports global illumination, accurately capturing effects such as soft shadows, reflection, refraction, and subsurface scattering. These effects are crucial for visual realism, and together with the vast set of other features of Blender, they enable artists to create photo-realistic 3D scenes. The downside is that cycles is up to two orders of magnitude slower than a rasterized rendering engine, but for Kubric we decided that this computational cost is a worthwile tradeoff in exchange for the added realism and the ability to systematically study complex visual effects.

**Annotations**. Another important feature of Blender is the use of specialized render passes that compute auxiliary ground truth information. We leverage this feature to export (in addition to the RGB image) ① depth maps, ② instance segmentation, ③ optical flow, ④ surface normals, and ⑤ object coordinates (see Fig. 1). In addition to these image space annotations, Kubric also automatically collects object-centric metadata, such as 2D/3D trajectories, 2D/3D bounding boxes, velocities, mass, friction, camera parameters, collision events, as well as custom metadata.

## 3.3. Assets

A limiting factor in the creation of synthetic scenes is the availability of high-quality 3D assets. Several asset collections exist, but their use often requires substantial cleanup and conversion to make them compatible with a given pipeline. Kubric provides several preprocessed collections of assets in a public Google Cloud bucket. Using these assets is as simple as changing the `path` of the asset source with `kb.AssetSource(path)`. At the core level, each dataset source is associated with a `manifest.json` file storing high level aggregated information, without the need

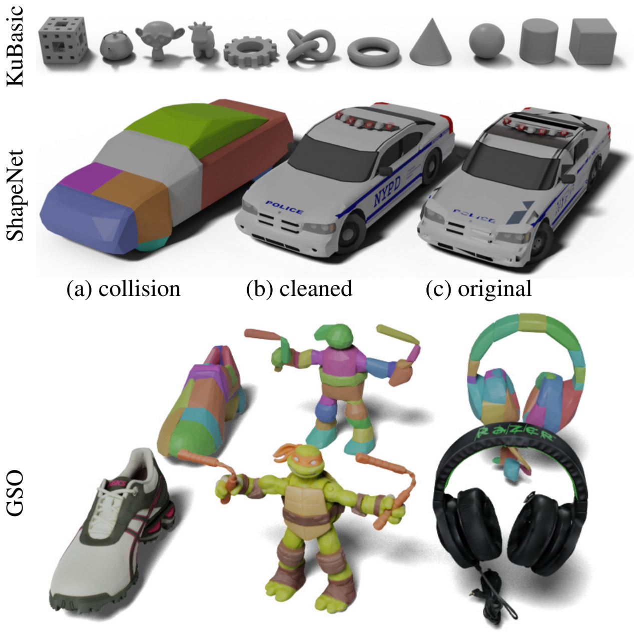

Figure 4. (top) The KuBasic assets collection. (middle) ShapeNet objects by default do not render well in Blender (c) due to problems with auto-smoothing and the lack of backface culling in cycles. (b) We processed all ShapeNet objects to fix these issues and (a) generated a collision mesh by first making the model watertight and then performing an approximate convex decomposition using VHACD. (bottom) Example assets from the Google Scanned Objects (GSO) dataset along with the generated collision meshes.

to traverse the entire folder structure. The `"id"` property of each entry in the manifest is in one-to-one correspondence to an archive file containing the data for the asset. Each of these archives a contains a JSON-file with metadata, including the paths to the sub-asset for rendering and for collision detection, and the definition of physical properties in the Unified Robot Description Format (URDF) used by PyBullet. For textured models, we employ the GLTF standard [79].

**KuBasic**. For simple prototyping we ship a small collection of eleven simple assets depicted in the top row of Fig. 4.

**ShapeNetCore.v2**. This dataset is a subset of the full ShapeNet dataset [14] with $51,300$ unique 3D models from $55$ with canonical alignment and common object categories annotations (both manually verified). Extensive pre-processing was performed to simplify the integration of these assets within Kubric, which included making the models watertight using [41], generating collision geometry using VHAC-D [62], and fixing raytracing artifacts due to auto-smoothing and intersecting faces(see Appendix B for details).

**Google Scanned Objects (GSO) [77]**. Is a dataset of common household objects that have been 3D scanned for use in robotic simulation and synthetic perception research. It is licensed under the CC-BY 4.0 License and contains $\approx$ 1k high-quality textured meshes; see Fig. 4. We publish pre-processed version of this dataset in the Kubric format, which again includes generated collision meshes.

**Polyhaven [115]**. is a public (CC0 licensed) library from which we have collected and pre-processed HDRI images for use as backgrounds and lighting, and textures for use in high-quality materials.

### 3.4. Scene Understanding Datasets (SunDs)

To facilitate ingesting data into machine learning models, we introduce, alongside Kubric, the SunDs (Scene Understanding Datasets) dataset front-end[2]. SunDs is an API to access public scene understanding datasets. The field names and structure, shape, dtype are standardized across datasets. This allow to trivially switch between datasets (e.g. switch from synthetic to real data). All SunDs datasets are composed of two sub-datasets:

- The scenes dataset contains high level scene metadata (e.g. scene boundaries, mesh of the full scene, etc.).
- The frames dataset contains the individual examples within a scene (e.g., RGB image, bounding boxes, etc.).

SunDs abstracts away the dataset-specific file format (json, npz, folder structure, ...), and returns tensors directly ingestible by machine learning models (TF, Jax, Torch). Internally, SunDs is a wrapper around TFDS, which allows one to scale to huge datasets ($\approx$ TB), to provide native compatibility with distributed cloud file systems (e.g. GCS, S3), and to leverage `tf.data` pipeline capabilities (prefetching, multi-threading, auto-caching, transformations, etc.).

To simplify even further data ingestion, SunDs introduce, on top of TFDS, the concept of tasks. Each SunDs dataset can be loaded for different tasks. Tasks control:

- Which features of the dataset to use/decode. Indeed, scene understanding datasets often have many fields (lidar, optical flow, ...), but only a small subset are used for any given task. Selecting which fields are used avoids the cost of decoding unnecessary features.
- Which transformation to apply to the pipeline. For example, the NeRF task will dynamically generate the rays origin/directions from the camera intrinsics/extrinsics contained in the dataset.

## 4. Kubric Datasets and Challenges

To demonstrate the power and versatility of Kubric, we next describe a series of new challenge problems, each with data[3] generated by Kubric (see Tab. 2). They cover 2D and 3D tasks at different scales, with dataset sizes ranging from

[2] https://github.com/google-research/sunds

[3] The presented datasets along with the corresponding worker scripts can be found at https://github.com/google-research/kubric.

| | Section | task domain | flow | segmentation | depth | camera 3D pose | object 3D poses | physics sim. | rigged animation | control backgrnd. | control materials | control lighting | new challenge | sim-to-real | hypothesis testing | PII / legal | scale |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4.1* | Object discovery | 2D | ✓ | ✓ | × | × | × | ✓ | × | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | *TB* |
| 4.2* | Optical flow | 2D | ✓ | × | × | × | × | ✓ | × | ✓ | × | ✓ | × | ✓ | × | × | *TB* |
| 4.3* | NeRF & Texture | 3D | × | × | ✓ | × | × | × | × | × | ✓ | × | × | × | ✓ | × | *MB* |
| 4.4 | Pose-estimation | 2D | × | × | × | × | ✓ | × | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | ✓ | *GB* |
| 4.5* | Pre-training | 2D | × | × | × | × | × | × | × | × | ✓ | ✓ | × | ✓ | × | ✓ | *GB* |
| C.1* | Robust NeRF | 3D | × | × | × | ✓ | × | × | × | × | ✓ | × | ✓ | × | ✓ | × | *MB* |
| C.2* | Multi-view SOD | 2D | × | ✓ | × | × | × | × | × | ✓ | × | ✓ | × | ✓ | ✓ | × | *GB* |
| C.3* | Complex BRDFs | 3D | × | × | × | ✓ | × | × | × | × | ✓ | ✓ | ✓ | × | ✓ | × | *GB* |
| C.4* | 3D reconstruction | 3D | × | ✓ | × | ✓ | ✓ | × | × | × | × | × | ✓ | × | × | × | *GB* |
| C.5* | Robust 3D recons. | 3D | ✓ | ✓ | × | × | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | *MB* |
| C.6* | Point tracking | 2D | × | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | ✓ | ✓ | × | × | × | *TB* |
| C.7 | ToyBox | 3D | × | ✓ | ✓ | ✓ | ✓ | × | × | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | *GB* |
| C.8 | Novel View Synthesis | 3D | × | × | × | ✓ | × | ✓ | × | ✓ | × | ✓ | ✓ | × | ✓ | × | *GB* |

Table 2. Overview of the datasets / challenges presented in Sec. 4 (**claimed as dataset contributions for this paper*).

MBs to TBs. Each relies on a different subset of annotations (flow, segmentation, depth, camera pose, or object pose), makes use of a different subset of features (e.g., physics or rigged animation), and requires control over different factors (background, materials, or lighting). Any one dataset might have been generated by a simpler, specialized code-base, but this would have been extremely inefficient. Rather, with the versatility of Kubric, it was straightforward to create, extend and combine datasets, leveraging a common platform and shared engineering efforts.

These different challenges also highlight different uses of synthetic data. Some serve as benchmarks for comparing existing and future methods, while others provide additional training data for real-world applications (sim-to-real). Some are designed to empirically test specific hypotheses (e.g., in testing), while some focus on data that can be shared without privacy and legal concerns. We describe four challenges in sections below and 8 more in the appendix C.

### 4.1. Object discovery from video

Object discovery methods aim to decompose a scene into its constituent components and find object instance segmentation masks with minimal supervision. While recent models such as IODINE [36], MONet [12], GENESIS [28], and Slot Attention [59] succeed at decomposing simple scenes with uniform textures, decomposing dynamic scenes (i.e., videos) with high visual complexity and complex dynamics remains difficult. This challenge introduces five Multi-Object Video (MOVi) datasets, MOVi-A to -E (see Fig. 5), of increasing visual and dynamical complexity, aimed at testing the limits of existing object discovery approaches, enabling progress towards more realistic and diverse visual scenes.

We test two state-of-the-art video object discovery methods, SAVi [50] and SIMONe [45], for their ability to decompose videos into temporally consistent object masks (see Tab. 3). SAVi, which uses optical flow during training, performs better at decomposing moving objects, especially

| Method | MOVi-A | MOVi-B | MOVi-C | MOVi-D | MOVi-E |
|---|---|---|---|---|---|
| SAVi [50] | **82.0** ± 0.3 | **61.5** ± 0.3 | **47.0** ± 0.3 | 19.4 ± 8.0 | 2.7 ± 0.5 |
| SIMONe [45] | 61.8 ± 2.0 | 30.7 ± 3.3 | 19.8 ± 0.5 | **34.1** ± 0.7 | **34.9** ± 0.6 |
| SAVi + BBox | 95.3 ± 0.2 | 85.5 ± 0.2 | 73.5 ± 0.3 | 9.9 ± 1.4 | 7.5 ± 1.0 |

Table 3. **Object discovery –** Object segmentation performance, measured in terms of foreground ARI [36, 42] (FG-ARI↑) in %. We compare two recent state-of-the-art models, SAVi [50] (trained using optical flow) and SIMONe [45]. SAVi + BBox additionally receives object bounding boxes as cues in the first frame.

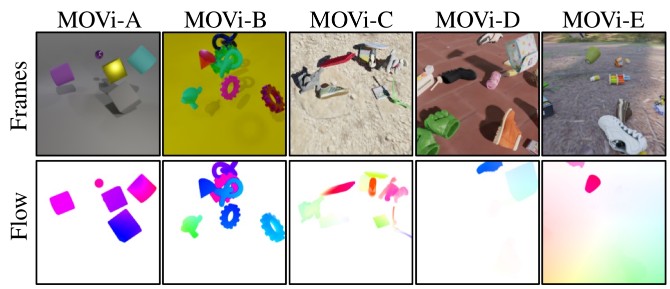


Figure 5. **Object discovery –** Dataset samples of MOVi of increasing visual complexity. MOVi-A uses objects inspired by CLEVR [44]. MOVi-B introduces additional primitive object types and colors. MOVi-C introduces real-world backgrounds and scanned 3D objects. MOVi-A to -C contain scenes of up to 10 moving objects (24 frames per video). MOVi-D & MOVi-E scenes have up to 23 objects, with only a small fraction of moving objects. In MOVi-E, the camera is moving in random directions.

when receiving bounding boxes for the first frame of the video. Both methods decline in performance as complexity increases with an exception for static objects in MOVi-D and -E, which are partially captured by SIMONe. Neither method can reliably decompose scenes in all five datasets.

### 4.2. Optical flow

Optical flow refers to the 2D motion from pixels in one frame to the next in a video. It is fundamental to video processing and analysis. Unlike high-level vision tasks, we cannot obtain reliable, ground-truth optical flow on generic real-world videos, even with human annotation. Optical flow is actually the first sub-field of computer vision to rely on synthetic data for evaluation [7].

Recent deep models, PWC-net [90], RAFT [92], and VCN [110], all rely on synthetic data for pre-training, like FlyingChairs [26]. However, FlyingChairs lacks photo-realism, uses synthetic chairs as the only foreground objects, and does not have general 3D motion. AutoFlow [89] learns rendering hyperparameters for generating a synthetic flow datasets, yielding large performance gains over FlyingChairs [89]. But AutoFlow adopts a simple 2D layered model, lacks 3D motion and realism in rendering. Our dataset addresses these shortcomings, as shown in Fig. 6.

We compare training RAFT on different datasets using the same training protocol [88, 89, 92]. As shown in Ta-

| | | Sintel | | KITTI-15 | |
|---|---|---|---|---|---|
| Dataset | Parameters | Clean | Final | AEPE | ER% |
| FlyingChairs (2D) | Manual | 2.27 | 3.76 | 7.63 | 38.5% |
| Kubric (3D) | Manual | **1.89** | 3.02 | 4.82 | 16.9% |
| AutoFlow (2D) | Learned | 2.08 | **2.75** | **4.66** | **15.3%** |

Table 4. **Optical flow** – Comparison of pre-training RAFT on different optical flow datasets (lower is better for all metrics).

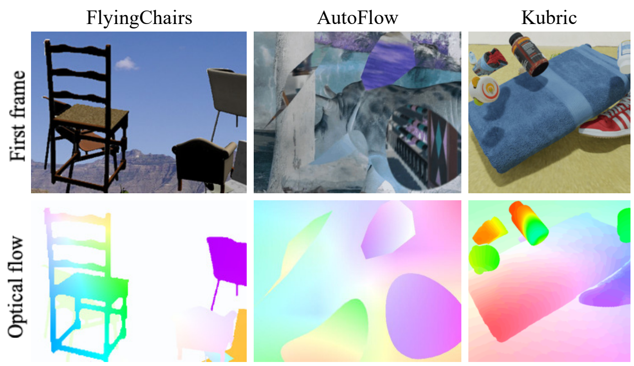


Figure 6. **Optical flow** – Chairs in FlyingChairs undergo 2D affine motion; random polygons in AutoFlow undergo nonrigid 2D motion; 3D objects in Kubric undergo 3D rigid-body motion.

| Frequency Cutoff | $10^{-0.5}$ | $10^{0}$ | $10^{0.5}$ | $10^{1}$ | $10^{2}$ |
|---|---|---|---|---|---|
| PSNR ↑ | **28.1** | 27.8 | 26.7 | 23.6 | 23.4 |
| Depth Variance↓ | 0.026 | 0.024 | 0.023 | 0.023 | **0.022** |

Table 5. **Texture-structure in NeRF** – Reconstruction error and depth variance for different texture frequency bands with NeRF on textured surface. Accuracy of color prediction improves as frequency of the texture becomes lower, while accuracy of the surface geometry degrades.

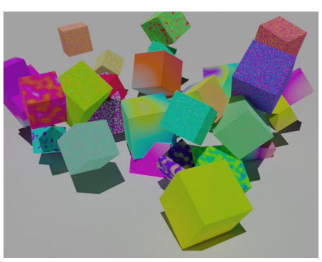

Figure 7. A NeRF dataset with procedural texture allows each pixel to be annotated with frequency information. This enables analysis of the frequency-structure relationship in the learned NeRF model.

ble 4, Kubric leads to significantly more accurate results than FlyingChairs when both use manually selected rendering hyperparameters, demonstrating the benefit of using 3D rendering. Kubric also performs competitively against AutoFlow. Note that this is not an apples-to-apples comparison, because the hyperparameters of AutoFlow have been learned to optimize the performance on the Sintel dataset [89]. These results suggest that learning hyperparameters for Kubric is likely to result in significant performance gains.

## 4.3. Texture-structure in NeRF

Neural radiance fields are inherently *volumetric* representations, but are commonly used to model the surfaces of

| Train data set | COCO + Active | COCO + Active + Synth |
|---|---|---|
| COCO [57] | 0.554 | 0.557 |
| Active [98] | 0.650 | 0.662 |
| Yoga | 0.391 | 0.427 |

Table 6. **Pose estimation** – results are improved out-of-domain by the addition of synthetic images of human models featuring poses outside the COCO domain; Keypoint Mean Average Precision (mAP) metric (higher is better)

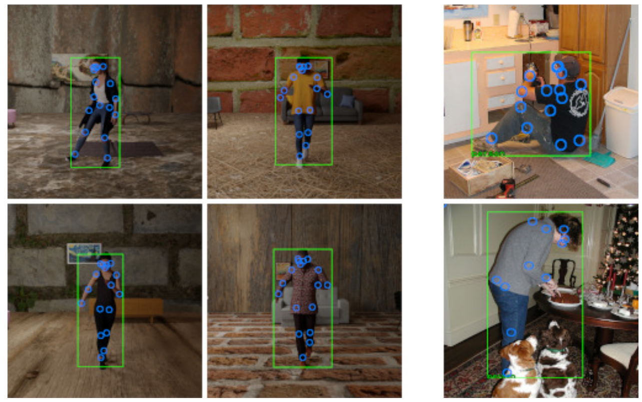

Figure 8. **Pose estimation** – fully annotated images from synthetic videos aimed at diversifying poses (left), motions, subjects and backgrounds featured in real-world annotated data sets, and (right) examples of COCO-equivalent images.

solid objects. These NeRF surface models are a result of the model trying satisfy a multi-view reconstruction problem: to reconstruct surface detail consistently from multiple views, those details must lie in a thin slice of the volume around the true surface. Note that not all surfaces will encourage NeRF to build a surface model. Surfaces with flat color may still be reconstructed as a non-solid volume. Hence, benchmarking NeRF methods according to how well they stay true to the actual surface depending on texture is an interesting aspect that is still unexplored.

To quantify this, we created synthetic scene containing flat surfaces, the textures of which are procedurally generated with blue noise to have varying spatial frequency. We annotate each pixel with the cutoff frequency of its texture and analyze the correlation between frequency, depth variance, and reconstruction error. We then train a NeRF model with this synthetic data. As shown in Table 5, we find that increasing frequencies are associated with lower depth variance, indicating better approximations to a hard surface, while also increasing the reconstruction error, showing that the network is less able to approximate the complex textures. It would be interesting to see how well future volumetric multi-view reconstruction methods would cope with this ambiguity and encourage hard surface boundaries.

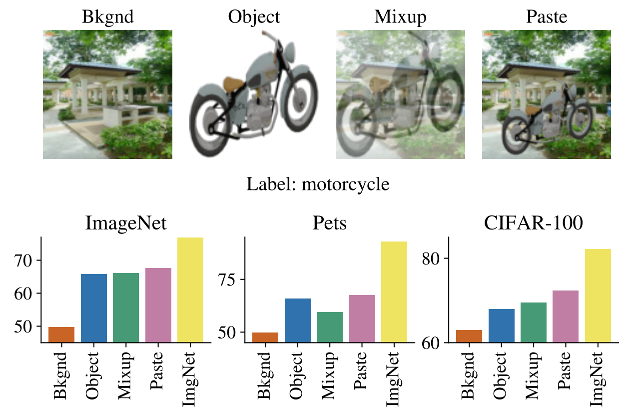


Figure 9. **Pre-training** a ResNet50 on synthetic Kubric data (top) and transferring it to standard bencharks (bottom) halves the gap between random pre-training (Bkgnd) and ImageNet pre-training.

### 4.4. Pose estimation

Pose-estimation-based interactive experience (e.g., Kinect) often feature human poses that remain under-represented in most data sets comprising user-generated pictures (e.g. COCO [57]), as picture-worthy poses present an obvious sampling bias. Simulated data can supplement real data with less aesthetic poses which are nonetheless common in real-life human motions. Here we improve MoveNet [98], a CenterNet [119] based pose inference CNN usually trained on COCO [57] and Active [98] (a proprietary data set with more diverse poses). As in Simpose [120], training batches mix real and synthetic data with an $80/20\%$ mixture. Unlike [120], synthetics do not provide additional labels (e.g., surface normals) but only contribute more diverse examples. As illustrated in Figure 8, the samples feature $41$ rigged RenderPeople models placed in a randomized indoor scene where background elements and textures come from BlenderKit and TextureHeaven. Human poses are extracted from dancing and workout ActorCore animations. While licensing terms of non-CC0 assets forbid data publication, the data set can be re-generated with our open source software by any owner of the same mesh and animation assets. Synthetic data improves keypoint Mean-Average-Precision (see Table 6), in domain (on COCO and Active), and out-of-domain (on Yoga, a test set of contorted poses comprising 1000 examples). Synthetic data are therefore now routinely used in our human-centric training procedures for still images as well as videos.

### 4.5. Pre-training visual representations

Ever since AlexNet [55], the entire field of computer vision has benefited immensely from re-using “backbones” pre-trained on large amounts of data [25,52,54,76]. However, recent work casts doubt on the continued use of datasets that consist of vast collections of photos from the internet [8,112]. One potential way forward, which completely circumvents the downsides of web-image based data, is to use rendered data. This has recently shown great success for face recognition [103], and we hypothesize that synthetic data could also eventually replace web images for pre-training general computer vision backbones. In order to evaluate the promise of such a setting, we perform a small pilot experiment. Kubric was used to render ShapeNet objects in various random poses on transparent backgrounds. We pre-train a ResNet-50 to predict the object’s category from images that combine the object with a random background image in various ways, as shown in Fig. 9 (top). We then transfer this pre-trained model to various datasets, following the protocol in [52]. Fig 9 (bottom) shows that this simple pilot experiment already halves the gap between random pre-training and pre-training on ImageNet, suggesting that this is a promising approach.

## 5. Conclusions

We introduce Kubric, a general Python framework complete with tools for generation at scale, integrating assets from multiple sources, rich annotations and a common export data format (SunDS) for porting data directly into training pipelines. Kubric enables the generation of high quality synthetic data, addressing many of the problems inherent in curating natural image data, and circumventing the expense of building task-specific, one-off pipelines. We demonstrate the effectiveness of our framework in 11 case studies with generated datasets of varying complexity for a range of different vision tasks. In each case, Kubric has substantially reduced the engineering effort to generate the required data and has facilitated reuse and collaboration. We hope that it will help the community by lowering the barriers to generating high-quality synthetic data, reduce fragmentation, and facilitate the sharing of pipelines and datasets.

**Limitations and future work**. While already tremendously useful, Kubric is still a work in progress and does not yet support many features of Blender and PyBullet. Notable examples include volumetric effects like fog or fire, soft-body and cloth simulations, and advanced camera effects such as depth of field and motion blur. We also plan to preprocess and unify assets from more sources, including the ABC dataset [51] or Amazon Berkeley Objects [17]. At present, Kubric requires substantial computational resources due to its reliance on a path-tracing renderer versus a rasterizing renderer. We hope to add support for a rasterizing backend, allowing users to trade-off speed and render quality.

We include a discussion on the potential societal impact and ethical implications surrounding the application of our system in Section A of the supplementary material.

## Acknowledgements

This work was supported by a UKRI Future Leaders Fellowship [grant number G104084].

## References


[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1597–1604. IEEE, 2009. 3

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014. 6

[3] Anonymous Authors. TAP-Net: Tracking any point in a video. *In Submission*, 2022. 5, 6

[4] Yuki M. Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. Pass: An imagenet replacement for self-supervised pretraining without humans. *NeurIPS Track on Datasets and Benchmarks*, 2021. 2

[5] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations?, 2019. 2

[6] S Baker, D Scharstein, JP Lewis, S Roth, MJ Black, and R Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92:1–31, 2011. 2

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994. 2, 6

[8] Abeba Birhane and Vinay Uday Prabhu. Large image datasets: A pyrrhic win for computer vision? In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 1536–1546. IEEE, 2021. 8

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2021. 2

[10] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient object detection: A benchmark. *IEEE transactions on image processing*, 24(12):5706–5722, 2015. 3

[11] Steve Borkman, Adam Crespi, Saurav Dhakad, Sujoy Ganguly, Jonathan Hogins, You-Cyuan Jhang, Mohsen Kamalzadeh, Bowen Li, Steven Leal, Pete Parisi, Cesar Romero, Wesley Smith, Alex Thaman, Samuel Warren, and Nupur Yadav. Unity perception: Generate synthetic data for computer vision, 2021. 2

[12] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. 6

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 2

[14] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D model repository. Dec. 2015. 5, 8

[15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 8

[16] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[17] Jasmine Collins, Shubham Goel, Achleshwar Luthra, Leon Xu, Kenan Deng, Xi Zhang, Tomas F Yago Vicente, Himanshu Arora, Thomas Dideriksen, Matthieu Guillaumin, and Jitendra Malik. ABO: Dataset and benchmarks for Real-World 3D object understanding. Oct. 2021. 8

[18] E Coumans and Y Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 4

[19] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 2

[20] Ingemar J Cox, Satish B Rao, and Yu Zhong. " ratio regions": a technique for image segmentation. In *Proceedings of 13th International Conference on Pattern Recognition*, 1996. 2

[21] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2

[22] Dawson-Haggerty et al. trimesh. 2

[23] M Denninger, M Sundermeyer, and others. Blenderproc. *arXiv:1911.01911*, 2019. 2, 3

[24] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d pose estimation: motion to the rescue. *NeurIPS*, pages 12949–12961, 2019. 2

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 8

[26] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 2, 6

[27] A Eftekhar, A Sax, and others. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. *arXiv:2110.04994*, 2021. 2

[28] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and

sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020. 6

[29] Deng-Ping Fan, Ming-Ming Cheng, Yun Liu, Tao Li, and Ali Borji. Structure-measure: A new way to evaluate foreground maps. In *Proceedings of the IEEE international conference on computer vision*, pages 4548–4557, 2017. 3

[30] Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2777–2787, 2020. 2, 3

[31] Blender Foundation. Blender 2.93.6 release candidate python api documentation, 2021. 3

[32] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE, 2018. 2

[33] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. *CVPR*, 2016. 2

[34] C Gan, J Schwartz, and others. ThreeDWorld: A platform for interactive Multi-Modal physical simulation. *arXiv:2007.04954*, 2020. 2, 3

[35] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 8

[36] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433, 2019. 6

[37] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[39] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020. 6

[40] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019. 2

[41] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020. 5, 2

[42] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. 6

[43] S James, P Wohlhart, M Kalakrishnan, and others. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *Proceedings of the*, 2019. 4

[44] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6, 4

[45] Rishabh Kabra, Daniel Zoran, Loic Matthey Goker Erdogan, Antonia Creswell, Matthew Botvinick, Alexander Lerchner, and Christopher P. Burgess. SIMONe: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *arXiv preprint arXiv:2106.03849*, 2021. 6

[46] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable deep reinforcement learning for Vision-Based robotic manipulation. June 2018. 4

[47] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1988. 2

[48] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proc. CVPR*, pages 3907–3916, 2018. 3, 4

[49] Kangyeol Kim, Sunghyun Park, Jaeseong Lee, Sunghyo Chung, Junsoo Lee, and Jaegul Choo. Animeceleb: Large-scale animation celebfaces dataset via controllable 3d synthetic models, 2021. 2

[50] Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. *arXiv preprint arXiv:2111.12594*, 2021. 6

[51] S Koch, A Matveev, Z Jiang, and others. ABC: A big cad model dataset for geometric deep learning. *Proceedings of the*, 2019. 8

[52] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V*, volume 12350 of *Lecture Notes in Computer Science*, pages 491–507. Springer, 2020. 8

[53] 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, 2019. 3

[54] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2661–2671. Computer Vision Foundation / IEEE, 2019. 8

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural net-

works. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. 8

[56] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Milos Hasan, Ravi Ramamoorthi, and Manmohan Chandraker. Openrooms: An end-to-end open framework for photorealistic indoor scene datasets. *CoRR*, abs/2007.12868, 2020. 2

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7, 8

[58] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 4

[59] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020. 6

[60] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021. 6

[61] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM transactions on graphics*, 33(4):1–12, July 2014. 3

[62] Khaled Mamou, E Lengyel, and AK Peters. Volumetric hierarchical approximate convex decomposition. In *Game Engine Gems 3*, pages 141–158. AK Peters, 2016. 5, 2

[63] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 8

[64] N Mayer, E Ilg, P Hausser, P Fischer, and others. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2

[65] J McCormac, A Handa, and others. Scenenet RGB-D: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *CVPR*, 2017. 2

[66] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 8

[67] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[68] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. CVPR*, 2020. 3

[69] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 1988. 2

[70] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 6

[71] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases, 2020. 2

[72] Xuebin Qin, Shida He, Zichen Zhang, Masood Dehghan, and Martin Jagersand. Bylabel: A boundary based semi-automatic image annotation tool. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. 2

[73] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020. 2, 3

[74] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7479–7489, 2019. 2, 3

[75] W Qiu and A Yuille. UnrealCV: Connecting computer vision to unreal engine. In *ECCV Workshops*. Springer, 2016. 2

[76] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 512–519. IEEE Computer Society, 2014. 8

[77] Google Research. Scanned objects dataset of common household objects, 2021. 5

[78] S R Richter, V Vineet, S Roth, and V Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2

[79] Fabrice Robinet, Rémi Arnaud, Tony Parisi, and Patrick Cozzi. gltf: Designing an open-standard runtime asset format. *GPU Pro*, 5:375–392, 2014. 5

[80] G Ros, L Sellart, J Materzynska, and others. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[81] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 2008. 2

[82] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene Representation Transformer: Geometry-Free Novel View

Synthesis Through Set-Latent Scene Representations. *CVPR*, 2022. 9

[83] 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. *CoRR*, abs/1904.01201, 2019. 3

[84] Max Schwarz and Sven Behnke. Stillleben: Realistic scene synthesis for deep learning in robotics. *CoRR*, abs/2005.05659, 2020. 3

[85] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304, 2011. 2

[86] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021. 3, 4, 8, 9

[87] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Proc. NeurIPs*, 2019. 3

[88] Deqing Sun, Charles Herrmann, Varun Jampani, Michael Krainin, Forrester Cole, Austin Stone, Rico Jonschkowski, Ramin Zabih, William T. Freeman, and Ce Liu. TF-RAFT: A tensorflow implementation of raft. In *ECCV Robust Vision Challenge Workshop*, 2020. 6

[89] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a bet @articleMayer2016-xo, title = "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation", author = "Mayer, N and Ilg, E and Hausser, P and Fischer, P and others", journal = "Proceedings of the", publisher = "openaccess.thecvf.com", year = 2016 ter training set for optical flow. In *CVPR*, 2021. 2, 6, 7

[90] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, June 2018. 6

[91] A Szot, A Clegg, and others. Habitat 2.0: Training home assistants to rearrange their habitat. *NeuRIPS*, 2021. 2

[92] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 6

[93] Thang To, Jonathan Tremblay, Duncan McKay, Yukie Yamaguchi, Kirby Leung, Adrian Balanon, Jia Cheng, William Hodge, and Stan Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. https://github.com/NVIDIA/Dataset_Synthesizer. 3

[94] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias, 2015. 2

[95] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011. 2

[96] James Traer and Maddie Cusimano. A perceptually inspired generative model of rigid-body contact sounds. https://www.dafx.de/paper-archive/2019/DAFx2019_paper_57.pdf. Accessed: 2021-11-17. 3

[97] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021. 8

[98] Ronny Votel, Yu-Hui Chen, and Na Li. Next-generation pose detection with movenet and tensorflow.js, 2021. 7, 8

[99] Chaoyang Wang, Simon Lucey, Federico Perazzi, and Oliver Wang. Web stereo video supervision for depth prediction from dynamic scenes. In *2019 International Conference on 3D Vision (3DV)*, pages 348–357. IEEE, 2019. 7

[100] Tinghuai Wang, Bo Han, and John Collomosse. Touchcut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding*, 2014. 2

[101] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019. 6, 7

[102] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*, 2018. 2

[103] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone, 2021. 2, 8

[104] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 1993. 2

[105] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2019. 2, 3

[106] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020. 2, 3

[107] F Xia, A R Zamir, Z He, A Sax, J Malik, and others. Gibson env: Real-world perception for embodied agents. *Proceedings of the*, 2018. 4

[108] Jiarui Xu and Xiaolong Wang. Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 6

[109] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *arXiv preprint arXiv:1612.00814*, 2016. 4

[110] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Advances in neural information processing systems*, pages 794–805, 2019. 6

[111] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T

Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *CVPR*, 2021. 4, 5

[112] Kaiyu Yang, Klint Qinami, Li Fei-Fei, Jia Deng, and Olga Russakovsky. Towards fairer datasets: filtering and balancing the distribution of the people subtree in the imagenet hierarchy. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 547–558. ACM, 2020. 8

[113] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv preprint arXiv:2106.12052*, 2021. 2

[114] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proc. CVPR*, pages 4578–4587, 2021. 3, 4, 8, 9

[115] Greg Zaal, Rob Tuytel, Rico Cilliers, James Ray Cock, Andreas Mischok, Sergej Majboroda, Dimitrios Savva, and Jurita Burger. Polyhaven: a curated public asset library for visual effects artists and game designers, 2021. 5, 3

[116] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021. 2

[117] Jia-Xing Zhao, Jiang-Jiang Liu, Deng-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. Egnet: Edge guidance network for salient object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8779–8788, 2019. 2, 3

[118] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-place scene labelling and understanding with implicit scene representation, 2021. 2

[119] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 8

[120] Tyler Zhu, Per Karlsson, and Christoph Bregler. Simpose: Effectively learning densepose and surface normals of people from simulated data. In *European Conference on Computer Vision*, pages 225–242. Springer, 2020. 8