BlenderProc

Synthetic Data Generation for Computer Vision

Comprehensive Tutorial

July 23, 2025

Contents

1 Introduction

BlenderProc is a synthetic data generator, it helps the process of training a neural network training by reducing the amount of manual data collection required. This tool allows for easy modifications of data such as noise creation(new objects and different background).

1.1 Pros

- Reduces manual data collection requirements
- Enables easy data variation and modification
- Accelerates neural network training processes
- Provides controlled testing environments

1.2 Cons

- Performance degradation due to sim2real gap
- Does not replicate the real world, might be faulty
- Requires time and attention, stipe learning curve

2 Installation and Setup

2.1 Environment Setup

To avoid conflicts and damage, create a dedicated conda environment:
Click here to follow the official Anacona installation guide

```
# Create new environment with Python 3.10
$ conda create --name blenderproc python=3.10


# Activate the environment
$ conda activate blenderproc

# To exit the environment (when needed)
$ conda deactivate

# Go inside your blenderproc folder inside the environment

cd your/path/to/env
```

2.2 BlenderProc Installation

2.2.1 Method 1: pip Installation

```
$ pip install blenderproc
```

Since i had some problem with pip i suggest to use git to install the full repository

2.2.2 Method 2: Git Installation

If pip installation fails, use the git method:

```
# Clone the repository
$ git clone https://github.com/DLR-RM/BlenderProc


# Navigate to the folder
$ cd BlenderProc

# Install all the dependencies required
$ pip install -e .
```

2.3 Quick Start

I higly recommend to start with the example they provide. **Note:** BlenderProc will automatically install Blender 4.2. Installation time depends on network connection speed.

```
$ blenderproc quickstart
```

2.4 Viewing Results

Check the generated example output:

```
$ blenderproc vis hdf5 output/0.hdf5
```

3 Dataset Downloads

BlenderProc provides access to various datasets.

3.1 CC Textures Dataset

Download over 1500 materials from cc0textures.com:

3.2 Available Datasets

- `ccTextures` - Textures
- `blenderkit` - Materials and models
- `haven` - Textures and models from polyhaven.com
- `pix3d` - IKEA dataset from its superset
- `scenenet` - Scene datasets
- `matterport` - 3D environment scans

4 Basic BlenderProc Script Structure

4.1 Essential Components

A basic BlenderProc script has five main components:

1. Scene Setup
2. Noise Adding

3. Light Addition
4. Camera Alignment
5. Output Rendering

4.2 Complete Basic Script

```python
import blenderproc as bproc
import numpy as np

# Initialize blenderproc
bproc.init()

# Load an object
objs = bproc.loader.load_obj("path/to/model.obj")

# Randomize object pose to add noise to the scene
for obj in objs:
    obj.set_location(np.random.uniform(-1, 1, size=3))
    obj.set_rotation_euler(np.random.uniform(0, np.pi*2, size=3))

# Add light to your scene
light = bproc.types.Light()
light.set_location([2, -2, 2])
light.set_energy(np.random.uniform(100, 500))

# Define camera poses with transformation matrix
cam_pose = bproc.math.build_transformation_mat(
    [0, -3, 1], [np.random.uniform(0.1, 0.5), 0, 0]
)
bproc.camera.add_camera_pose(cam_pose)

# Render the output
data = bproc.renderer.render()
bproc.writer.write_coco_annotations("output/", data, "scene")
```

5 Script Components Explained

5.1 Scene Setup

```python
import blenderproc as bproc
import numpy as np

# Initialize blenderproc
bproc.init()

# Load an object
objs = bproc.loader.load_obj("path/to/model.obj")
```

The scene setup initializes BlenderProc and loads 3D objects that will be used in the synthetic dataset.

5.2 Noise Addition

```python
# Randomize object pose to add noise to the scene
for obj in objs:
    obj.set_location(np.random.uniform(-1, 1, size=3))
    obj.set_rotation_euler(np.random.uniform(0, np.pi*2, size=3))
```

Noise addition randomizes object positions and rotations to simulate real-world variability, improving model accuracy when applied to actual scenarios.

5.3 Lighting

```
light = bproc.types.Light()
light.set_location([2, -2, 2])
light.set_energy(np.random.uniform(100, 500))
```

Adding light is an important step, it helps emulate real-world light such as the sun or a lamp

5.4 Camera Alignment

Camera alignment is the process of telling blenderproc what the camera should see and where it should be in the scene.

```
cam_pose = bproc.math.build_transformation_mat(
    [0, -3, 1], [np.random.uniform(0.1, 0.5), 0, 0]
)
bproc.camera.add_camera_pose(cam_pose)
```

5.5 Output Rendering

```
data = bproc.renderer.render()
bproc.writer.write_coco_annotations("output/", data, "scene")
```

The final step renders the scene and exports the data in COCO annotation format for use in machine learning.

6 Debugging

BlenderProc provides an excellent debugging feature to check the "Behind the scenes":

```
$ blenderproc debug yourfile.py
```

This allows real-time visualization of script execution and scene construction.

7 How to create noise

7.1 Prerequisites

Download the code from the GitHub repository with this command:

```
$ git clone https://github.com/Faboohh/BlenderprocTutorial.git
```

and prepare the following datasets:

7.2 CC Textures Dataset

```
$ blenderproc download cc_textures path/to/folder/where/to/save/materials
```

Warning: This is a large dataset (1900 folders) that may take 45+ minutes to download.

7.3 IPD Dataset Setup

We are going to need an even larger dataset, which is the IPD dataset. This is going to take a couple of hours to download depending on internet connection and speed

```
# Create dataset folder
$ mkdir bpd_datasets && cd bpd_datasets

# Create IPD folder
$ mkdir ipd && cd ipd

# Set environment variable in the CLI (if you restart the CLI without having
    downloaded the files redo this command)
$ export SRC=https:https://huggingface.co/datasets/bop-benchmark/ipd/tree/main

# Download required files

$ wget $SRC/ipd_base.zip
$ wget $SRC/ipd_models.zip
```

7.4 File Organization

After downloading and extracting the ZIP files:

1. Extract both ZIP files
2. Move folders from `ipd_models.zip` into the main `ipd` folder
3. Download `main_v2.py` from the project repository
4. Place `main_v2.py` in `examples/dataset/bop_object_physics_positioning/`

7.5 Running the Advanced Script

```
$ blenderproc run examples/datasets/bop_object_physics_positioning/main_v2.py \
  /data/bpc_datasets ipd resources/cctextures output_main_v2
```

Important: Ensure all file paths are correct. The IPD parameter should target the folder; objects will be selected randomly.

After running the command two folders with different outputs will get generated, inside them there will be images containing a background from cctextures and random objects from the ipd dataset

8 Troubleshooting

8.1 Git Installation Error

If you encounter the error:

```
The git executable must be specified in one of the following ways...
```

This indicates that Git is not installed or not globally accessible. Install Git using:

```
$ sudo apt install git
```

8.2 Common Issues and Solutions

- **Path Issues:** Always verify that file paths are correct and accessible
- **Environment Conflicts:** Use the dedicated conda environment
- **Memory Issues:** Large datasets may require significant RAM and storage
- **Network Timeouts:** Dataset downloads may fail due to network issues; retry if necessary

9 Conclusion

BlenderProc provides a powerful framework for generating synthetic training data for computer vision applications. While the Sim2Real gap presents challenges, the tool's flexibility and extensive dataset support make it valuable for accelerating machine learning model development.

The key to successful synthetic data generation lies in:

- Proper noise and variation introduction
- Realistic lighting and camera setup
- Appropriate dataset selection and preparation
- Iterative refinement based on model performance

Regular debugging and visualization using BlenderProc's built-in tools will help ensure optimal results and identify potential issues early in the development process.