Abstract-Rendering-Toolkit

Chenxi Ji, chenxij2@illinois.edu, 01/30/2026

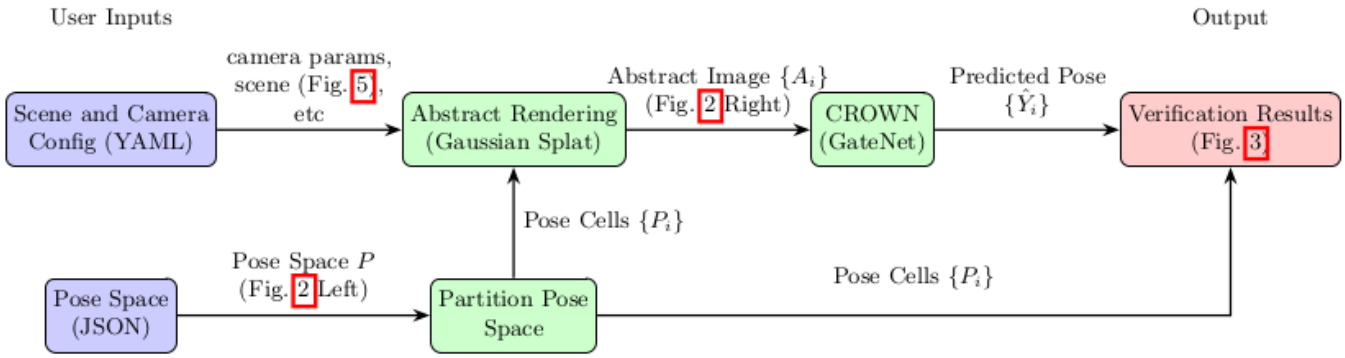
This repository provides a user-friendly implementation of **Abstract-Rendering**, which computes the set of images that can be rendered from a set of camera poses under a 3D Gaussian Scene, along with downstream applications such as classification, pose estimation, and object detection.

You can find more resources here:

- Paper (NeurIPS 2025)
- Project Website
- Presentation Slides (NeurIPS 2025)

Follow the steps below to set up the environment, gather scene data, and run the scripts.

Workflow



Setup

0. (Optional) Install Nerfstudio

The scene representation is required to follow the Nerfstudio data format. Therefore, installing Nerfstudio is recommended but not strictly required. You may either follow the installation commands provided in `nerfstudio_installation_commands.md` or refer to the official Nerfstudio installation guide (note that some steps on the website may be outdated).

1. Clone the Abstract-Rendering repository

Download the repository from GitHub:

```
cd ~
git clone --branch master
https://github.com/IllinoisReliableAutonomyGroup/Abstract-Rendering.git
```

2. Install auto_LiRPA

Install the neural network verification library *auto_LiRPA*, and symbolic link it under the Abstract-Rendering dictionary.

```
cd ~
git clone --branch van_verify_fix_six https://github.com/Verified-Intelligence/auto_LiRPA.git
cd ~/Abstract-Rendering
cd ln -s ~/Verified-Intelligence/auto_LiRPA auto_LiRPA
```

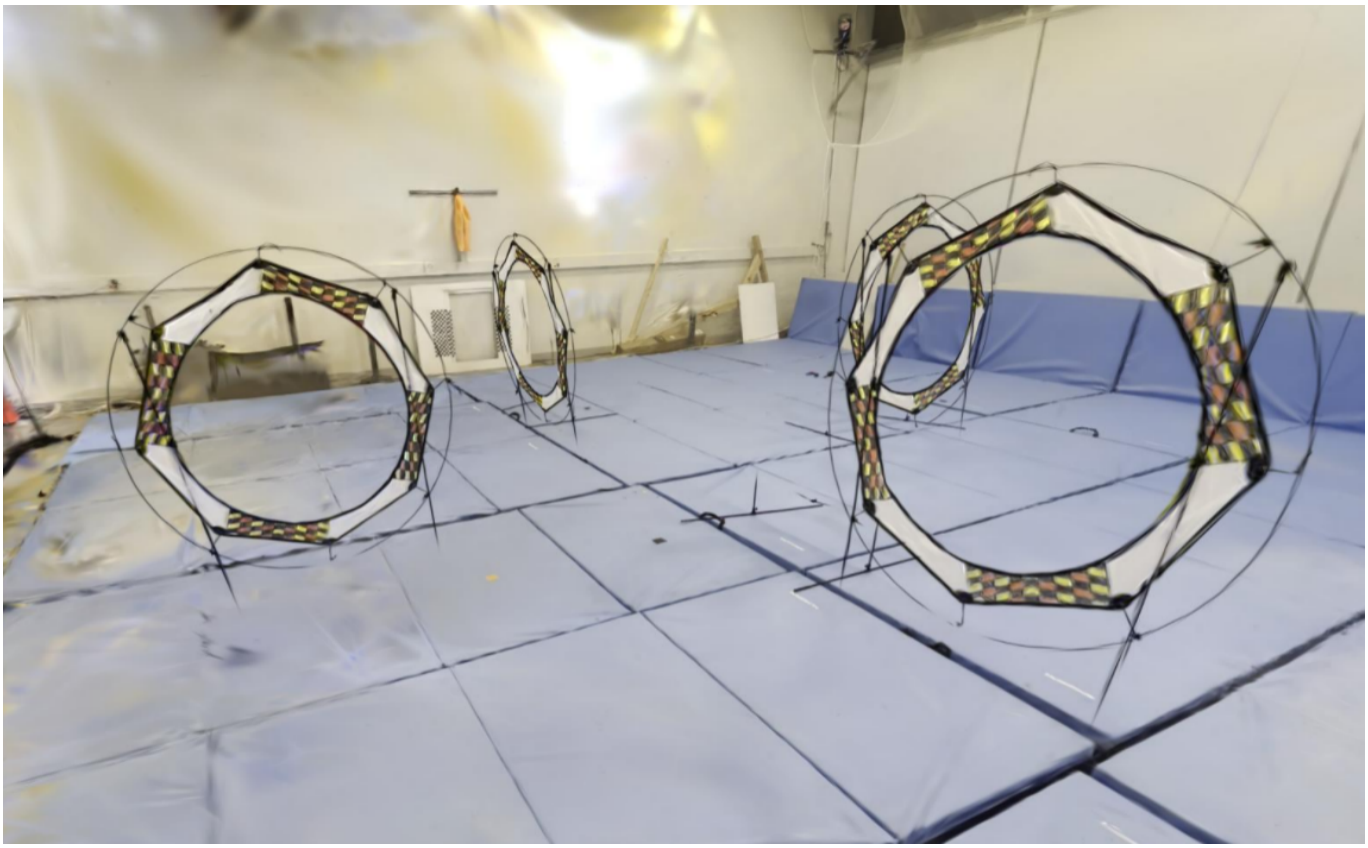
Note: This is a hosted in granted by Prof. Huan Zhang.

3. (Optional) Download Scene Data

You may either use your existing Nerfstudio data or download the pre-reconstructed Nerfstudio scenes and place them in the below dictionary structure.

```
~/Abstract-Rendering/nerfstudio/outputs/${case_name}/${reconstruction_method}/${datatime}/...
```

Below is visualization of scene *circle*.



Examples

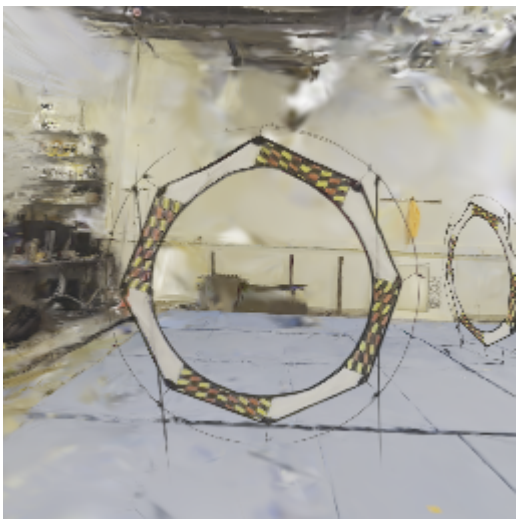
Note: The default GPU memory is 16GB. If you machine has less, please reduce the value of `gs_batch` in `config.yaml`.

Normal Rendering

You can use the command below to render images from a specified set of waypoints in a given scene (e.g. *circle*):

```
cd ~/Abstract-Rendering
export case_name=circle
python3 python3 scripts/render_gsplat.py --config
configs/${case_name}/config.yaml --odd configs/${case_name}/traj.json
```

The rendered image (`ref_######.png`) will be saved under `~/Abstract-Rendering/Outputs/RenderedIamges/${case_name}/${odd_type}`, for example:



Abstract Rendering

You can use the command below to generate abstract images from a specified set of waypoints in a given scene (e.g. *circle*):

```
cd ~/Abstract-Rendering
export case_name=circle
python3 scripts/abstract_gsplat.py --config
configs/${case_name}/config.yaml --odd configs/${case_name}/traj.json
```

The rendered images (`abstract_######.pt`) will be saved under `~/Abstract-Rendering/Outputs/AbstractIamges/${case_name}/${odd_type}`, and can be visualized by command, (e.g. *circle*):

```
cd ~/Abstract-Rendering
export case_name=circle
```

```
python3 scripts/vis_absimg.py --config configs/${case_name}/vis_absimg.yaml
```

The visualization of abstract image would be like

where the top-left subfigure shows sample concrete image from the pose cell; the bottom-left/right subfigure shows lower/upper bound abstract images; the top-right subfigure shows per-pixel difference between bounds as a heatmap.

Train Gatenet

```
cd ~/Abstract-Rendering
export case_name=circle
python3 scripts/train_gatenet.py --config configs/${case_name}/gatenet.yml
--samples configs/${case_name}/samples.json
```

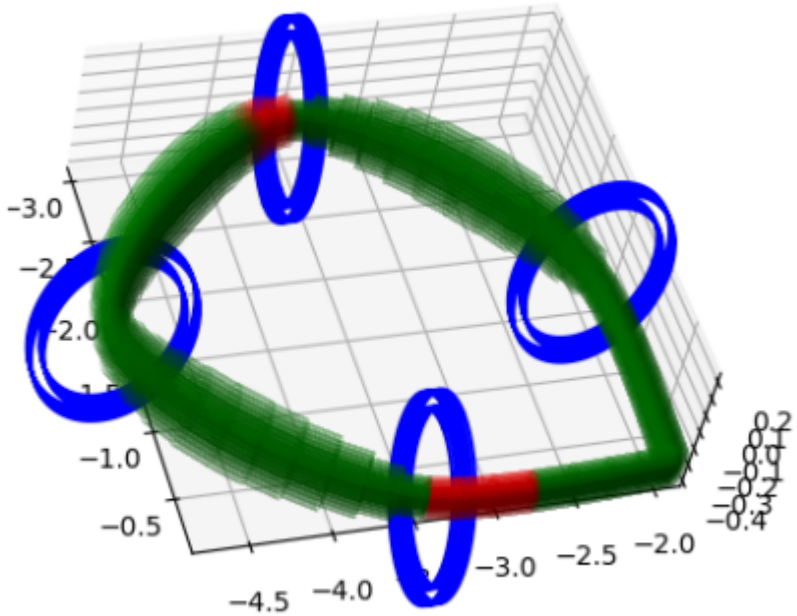
Certify Gatenet

```
cd ~/Abstract-Rendering
export case_name=circle
python3 scripts/certify_gatenet.py --config
configs/${case_name}/gatenet.yml
```

Visualize Certification Results

```
cd ~/Abstract-Rendering
export case_name=circle
python3 scripts/plot_gatenet.py --config configs/${case_name}/gatenet.yml -
-traj configs/${case_name}/traj.yaml
```

The visualization of Gatenet Verification is like:



where green indicates certified regions; red denotes potential violations; blue indicates gates.

Scripts (to be done)

`render_gsplat.py`:

`abstract_gsplat`:

`render_models.py`:

`utils_transform.py`:

`utils_alpha_blending.py`:

Citation

If you use this repository or the Abstract-Rendering toolkit in your work, please consider citing our NeurIPS 2025 splotlight poster:

BibTeX:

```
@inproceedings{ji2025abstractrendering,
  title     = {Abstract Rendering: Certified Rendering Under 3D Semantic
Uncertainty},
  author    = {Ji, Chenxi and Li, Yangge and Zhong, Xiangru and Zhang, Huan
and Mitra, Sayan},
  booktitle = {Advances in Neural Information Processing Systems (NeurIPS)
2025},
  year      = {2025},
  note      = {Poster},
  url       =
{https://mitras.ece.illinois.edu/research/2025/AbstractRendering_Neurips202
5.pdf}
}
```