# Extended Abstract

**Motivation** Novel view generation is an important problem that has a wide range of applications in many fields of work. Fields like robotics and self-driving cars require fast and efficient representations of scenes to calculate safe physical movement, while other fields like virtual reality and augmented reality can take advantage of solutions to create more immersive experiences for entertainment and education. Historically, the most successful methods for novel view generation have been using algorithms that make point cloud reconstructions using SfM and MVS. These methods are a tried and tested method that uses geometric information to gather scene data. However, this method lacks the ability to recreate the scene in a way that actually looks like the original, only the geometry. This is where newer methods like NeRF and 3D Gaussian Splatting have an edge. While requiring more computational resources and training time, their ability to perform scene reconstruction in a online environments makes them well suited for problems that involves dynamic views.

**Method and Implementation** Our approach uses a combination of reinforcement learning and computer vision to create an agent that can explore a scene in a way that optimizes the performance of the two computer vision models we are testing, 3D Gaussian Splatting and Neural Radiance Fields. On the DRL side, we will be training a PPO algorithm, with additional experimentation of various RL methods to improve the training process. These methods include behavioral cloning to initialize the model, multi-modal rewards based on both the vision model metrics as well as proxy metrics that we use in place of the full metric calculations, and hyperparameter adjustments. We also use a multi-step pipeline, with different training iterations using unique rewards to improve the overall model performance. For the environment, we use Habitat-Sim, a 3D environment rendering simulator that allows us to manipulate a seeing agent in a preset 3D scene. We use our own implementation of PPO for this project, as well as our own agent class. On the computer vision side, we use the original paper's implementation of 3D Gaussian Splatting with some modifications to the COLMAP feature matching pipeline by using sequential matching as opposed to the exhaustive matching since we are using a video-like input in the form of an agent's rollout. For NeRF, we use a mostly vanilla version of Instant Neural Graphics Primatives, a faster version that has some image quality improvements and performance speedups compared to the original NeRF implementation. To evaluate our model, we will be using pixel signal to noise ratio (PSNR) of the generated images.

**Results** Our baseline algorithm, which trains PPO using a novelty-based reward calculated using the pose novelty and image novelty, achieves a low PSNR of 11.82. The first of our two iterations upon these initial weights, SIFT keypoint-based rewards, achieves a PSNR of 37.26. Finally, our alternative method, COLMAP feature-based rewards combined with the PSNR metrics obtained a PSNR of 25.68. Since we were attempting to optimize the performance of our CV models, the most obvious method was to use the PSNRs outputted by the models themselves. However, that turned out to be not very effective. The problem we found was that because this metric was unable to get a per-step reward, sparsely rewarding PPO only for the last step of a training epoch, PPO did not learn very effectively. This is on top of the fact that this method takes much longer to train and is computationally much more expensive than the SIFT keypoints-based reward.

**Discussion** While we are happy with the performance that our agent allowed the models to achieve, we have some limitations. The first is that due to time and computational constraints, we were unable to run on a variety of scenes, and had to stick with a relatively simple scene. This was disappointing because the whole point of using novelty-based rewards instead of a frontier-exploration type map was that it was more effective in larger scenes. Furthermore, the number of iterations that the full model was deployed on to get rewards was relatively small, meaning that COLMAP + PSNR could potentially have a higher PSNR, but couldn't reach it.

**Conclusion** Novel view generating models such as 3DGS and Instant-NGP are an interesting model to optimize for in terms of agent performance, as they require a balance of view novelty and "redundancy" (so that feature matching between images can be done). By combining novelty and matching keypoint counts, we ended up with a model that has decent performance. In the future, we would like to apply our model to a larger dataset like HM3D to see how generalizable it is, and whether it would be easy to fine tune it to retain performance.

# Optimizing 3D Scene Agent Exploration for Novel View Generation

**Brian Lee**
Department of Computer Science
Stanford University
bjlee25@stanford.edu

**Allison Tee**
Department of Computer Science
Stanford University
ateecup@stanford.edu

## Abstract

We implement a reinforcement learning-based approach to optimize 3D scene exploration for novel view generation using two novel view generation models: 3D Gaussian Splatting (3DGS) and Instant Neural Graphics Primitives (Instant-NGP). Using Habitat-Sim for simulated environments, we train an agent using Proximal Policy Optimization (PPO) to navigate scenes in ways that maximize view quality. Due to the sparsity and cost of directly using PSNR as a reward, we experiment with alternative training strategies, including multi-stage rewards with pose/image novelty, SIFT keypoint matches, and COLMAP-based proxy metrics. Our experiments show that SIFT-based rewards significantly outperform baseline novelty metrics and even the more costly PSNR-based fine-tuning approach, achieving an average PSNR of 37.26 for novel views. These findings suggest that combining view novelty with keypoint matchability results in more effective exploration behavior, and future work will test generalizability to larger datasets such as HM3D.

## 1 Introduction

3D scene reconstruction is an important problem that has a wide range of applications in many fields of work. Fields like robotics and self-driving cars require fast and efficient reconstruction for safe physical movement, while other fields like virtual reality and augmented reality can take advantage of solutions to create more immersive experiences for entertainment and education.

Historically, the most successful methods for reconstruction have been point cloud reconstructions using SfM and MVS. These methods are a tried and tested method that uses geometric information to gather scene data. However, this method lacks the ability to recreate the scene in a way that actually looks like the original, only the geometry. This is where newer methods like NeRF and 3D Gaussian Splatting have an edge. While requiring more computational resources and training time, their ability to perform scene reconstruction in a online environments makes them well suited for problems that involves dynamic views.

### 1.1 NeRF

Neural Radiance Fields (NeRF) Mildenhall et al. (2020) is a method for novel view synthesis that represents a 3D scene as a continuous volumetric field encoded by a neural network. Given a 3D position and a viewing direction, NeRF predicts the color and volume density at that point. These predictions are integrated along camera rays using volumetric rendering to produce photorealistic images from arbitrary viewpoints. NeRF has demonstrated impressive results in rendering complex scenes with high visual fidelity, especially in terms of view-dependent effects such as reflections and lighting changes.

Despite its rendering quality, NeRF has several limitations. Training is computationally expensive and typically requires hours on a GPU for a single scene. Rendering is also slow, as it involves ray marching and evaluating the neural network many times per pixel. Furthermore, NeRF lacks explicit geometry and is difficult to edit or integrate into standard 3D workflows. However, its flexibility and differentiable nature make it an important baseline for evaluating novel view generation and reconstruction methods.

### 1.2 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) Kerbl et al. (2023) is a recent technique that represents a scene using a set of anisotropic 3D Gaussians, each with attributes like position, color, opacity, and scale. These Gaussians are directly rendered onto the image plane using a fast differentiable rasterization pipeline. Unlike NeRF, which models the scene implicitly through a neural network, 3DGS provides an explicit, geometry-aware representation of the scene, enabling real-time rendering and interactive editing.

One of the key strengths of 3DGS is its efficiency. Training can be completed in minutes, and rendering can be done in real time, making it suitable for real-world applications such as AR/VR and robotics. However, because 3DGS does not model view-dependent appearance in its basic form, it can sometimes fall short of NeRF's photorealism in specific scenes. Even so, its speed and practicality make it a compelling alternative for scene reconstruction, especially in scenarios where rendering latency and sample efficiency are important.

## 2 Related Work

We initially came to this idea through a paper by Lei et al. on GaussNav Lei et al. (2025). GaussNav combines 3DGS and agentic movement to optimize for object finding in cluttered scenes. Their work builds on another paper by Li et al. on Semantic Gaussians, a way of segmenting a view generated by Gaussian splatting using traditional 2D encoders combined with embeddings of a 3D semantic network Li et al. (2024). While their work was relevant in combining 3DGS and agentic movement, we differ from their work in two main ways. The first is that while GaussNav is specifically training for object detection and navigation, our model aims for a higher-level understanding of the scene as a whole through an agent's optimization of these vision models. The second is that, on the reinforcement learning side, GaussNav uses frontier-based exploration to have the model explore parts of the environment that haven't been explored yet by keeping track of a global map. We were inspired by this idea, and used a similar method that mimicked this idea of unknown-exploration by using a novelty-based reward, which made a global map unnecessary.

The second paper we build off of is AutoNeRF by Marza et al. Their work is very similar to ours in that they are also using an agent to gather images to train an implicit representation-based novel view generation algorithm. However, their agent exploration strategy, like AutoNeRF, is based on frontier-based exploration. It keeps track of an explicit top-down semantic map at each step, updating it with each action the agent takes. Furthermore, their final model is evaluated using the accuracy of the semantic map as opposed to the reconstructions themselves, which is different from our approach.

## 3 Method

Our approach uses a combination of reinforcement learning and computer vision to create an agent that can explore a scene in a way that optimizes the performance of the two computer vision models we are testing, 3D Gaussian Splatting and Neural Radiance Fields.

On the computer vision side, we use the original paper's implementation of 3D Gaussian Splatting with some modifications to the COLMAP feature matching pipeline by using sequential matching as opposed to the exhaustive matching since we are using a video-like input in the form of an agent's rollout. For NeRF, we use a mostly vanilla version of Instant Neural Graphics Primatives, a faster version that has some image quality improvements and performance speedups compared to the original NeRF implementation. To evaluate our model, we will be using pixel signal to noise ratio (PSNR) of the generated images.

On the DRL side, we will be training our agent using proximal policy optimizaiton (PPO), with additional experimentation of various RL methods to improve the training process. We use our own

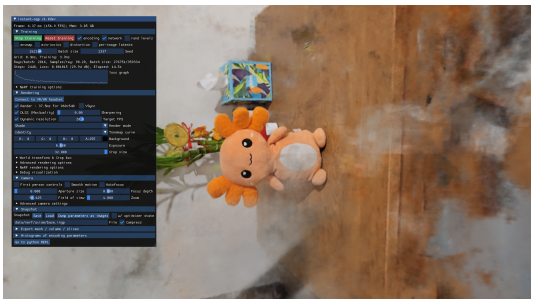


Figure 1: NeRF Sample using 50 images of Axiom Axolotl

basic implementation of PPO for this project, as well as our own agent class. This was because Habitat-Sim's RL environment counterpart, Habitat-Lab, had complex dependency issues that seemed very common based on feedback on their official page. For our agent, we implemented basic action selection by sampling logits, and created a customizable reward function so that we could test various ones. This reward function would be dense rewards that PPO traditionally excels at using. In our harness that ran the PPO and initialized the agent, we also added sparse reward integration for an additional reward.

We initially experimented with behavioral cloning by creating an "expert trajectory." However, we did not see a noticeable difference between runs that used BC and runs that didn't. We hypothesize that this was because we had provided expert trajectories in the form of a predetermined action sequence, but because of random scene initialization, the trajectories were being rendered unhelpful at best, or actively damaging at worst. While we looked into further experimentation, we decided that we would rather look into alternative RL methods.

Our main contribution comes in the form of the various rewards we tested. As a baseline, we implemented a novelty-based reward at each step of the rollout, calculated by combining the image novelty and pose novelty:

$$\text{Reward} = 0.5 \cdot r_{\text{pose}} + 0.5 \cdot r_{\text{image}}$$

$$r_{\text{pose}} = \begin{cases} 1.0 & \text{if } \|\mathbf{p}_{\text{new}} - \mathbf{p}_{\text{prev}}\| \geq \text{novelty\_radius for all previous poses} \\ 0.0 & \text{otherwise} \end{cases}$$

$$r_{\text{image}} = \begin{cases} 1.0 & \text{if SSIM}(\text{gray}_{\text{new}}, \text{gray}_{\text{prev}}) \leq 0.95 \text{ for all previous images} \\ 0.0 & \text{otherwise} \end{cases}$$

Then, we fine tuned the weights from training on the above rewards using two different rewards. The first reward we tried was a combination of COLMAP and PSNR metrics. While the view generation part of the models are quite fast (a few seconds for each novel view), we found that the training process was quite long ( 10 minutes per epoch). This meant that if we wanted to use the PSNR of the models, the training time would be quite long. As such, we combined it with a faster proxy in the form of COLMAP. Every 25 epochs, we would run the model training pipeline, but in every other epoch, we would instead use the number of registered images COLMAP ended up with when doing feature matching. While this itself also took some time ( 1 to 2 minutes per epoch), it was much faster than using the full model. We also ensured that the two reward types, full and proxy, were normalized to be around 1.0 so that we wouldn't get one overpowering the other, or other problems with gradient calculations.

The second reward we tried was a much simpler one that didn't involve any actual model metrics. We used SIFT to calculate the number of matching keypoints as a measure of the quality of the rollout's matchability. We used the OpenCV library SIFT implementation. The benefit to this was that, while the first method would result in a much more accurate metric of model performance (since

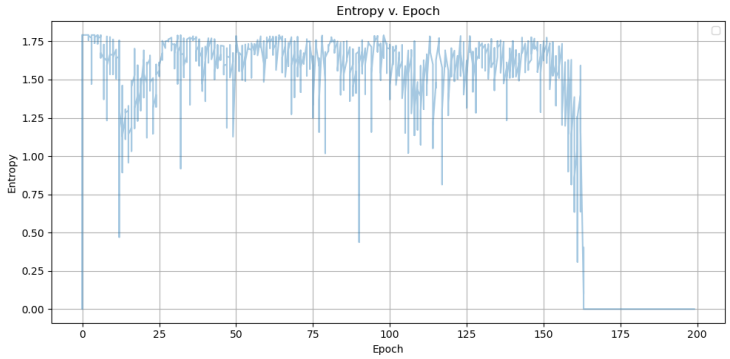
we directly used the metrics themselves), it was a sparse reward; we couldn't reward each action of the rollout directly, and could only evaluate the rollout as a whole, which meant that the agent would have difficulty learning exactly which actions in a rollout were good and bad. This could potentially be remedied by more epochs, but the first method also takes more time to run, so we hypothesized that the SIFT method would have comparable performance to the first. We then planned to use SIFT combined with the first method's rewards as well.

## 4 Experimental Setup

We ran our scripts and simulation on a local installation of Ubuntu 24.02 Python 3.9.21. The machine has an Intel i9 12900K with 64 GB of RAM, and an Nvidia RTX 4090 24 GB VRAM. We set up a Miniconda environment that used modified dependencies of Habitat-Sim, instant-ngp, 3DGS, and COLMAP to support all installations. We used the Skokloster Castle scene that came with Habitat-Sim as our simulation environment.

## 5 Results

After training our initial baseline, we found that the policy would relatively consistently collapse before the end of training.




The above figure shows the longest run without collapse for our original model, most runs collapsed at epochs less than 10. To combat this, we introduced an entropy bonus that would reward more exploration:

$$\text{PolicyLoss} = -\min(\text{surr}_1, \text{surr}_2) - \alpha \cdot \text{Entropy}$$

We experimented with $\alpha$ until we found one that saw entropy decrease but not collapse, and got $\alpha = 0.01$ to be effective.

We also modified the COLMAP + PSNR run to use minibatch updates.

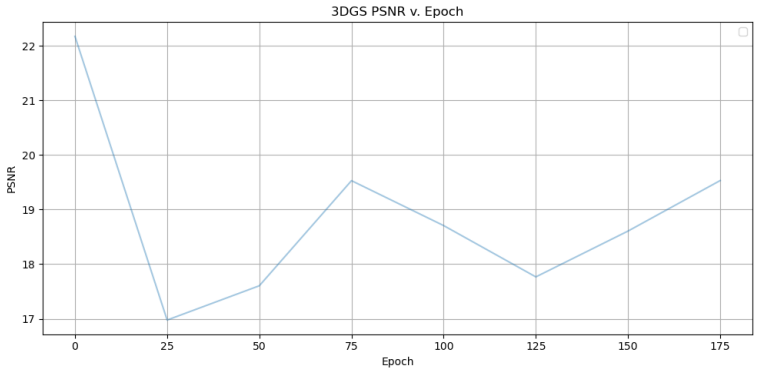


Table 1: Performance Comparison

| Method | PSNR (3DGS) | PSNR (Instant-NGP) |
|---|---|---|
| Baseline | 11.82 | 8.53 |
| COLMAP + PSNR | 25.68 | 24.02 |
| SIFT Keypoints | 37.26 | 33.59 |

### 5.1 Quantitative Evaluation

Our final baseline algorithm, which trains PPO using a novelty-based reward calculated using the pose novelty and image novelty, achieves a low PSNR of 11.82. The first of our two iterations upon these initial weights, SIFT keypoint-based rewards, achieves a PSNR of 37.26. Finally, our alternative method, COLMAP feature-based rewards combined with the PSNR metrics of 3DGS / Instant-NGP obtained a PSNR of 25.68 (3DGS for evaluation). We found that 3DGS consistently outperformed Instant-NGP, which looking back, was to be expected considering it does outperform Instant-NGP on many popular scene datasets anyway.

### 5.2 Qualitative Analysis

(a) Ground Truth (b) Rendered Image

Figure 2: Comparison between ground truth image and rendered image using SIFT-based exploration (best performing generation).

(a) Ground Truth (b) Rendered Image

Figure 3: Comparison between ground truth image and rendered image using SIFT-based exploration (worst performing generation).

(a) Ground Truth (b) Rendered Image

Figure 4: Comparison between ground truth image and rendered image using COLMAP + PSNR rewards.

While most rendered novel images for SIFT look like the best performing one, we did find some rare instances of visual artifacts in the scene, suggesting that the model still has some places to improve. For the COLMAP + PSNR model, there were no clear visual artifacts in any of the other scenes that were a good point of comparison.

Since we were attempting to optimize the performance of our CV models, the most obvious method was to use the PSNRs outputted by the models themselves. However, that turned out to be not very effective. The problem we found was that because this metric was unable to get a per-step reward, sparsely rewarding PPO only for the last step of a training epoch, PPO did not learn very effectively. This is on top of the fact that this method takes much longer to train and is computationally much more expensive than the SIFT keypoints-based reward. We had initially planned to then take the model of the SIFT keypoints-based reward and fine tune it with the rewards of COLMAP + PSNR, but given that SIFT was already achieving such high performance, it was unnecessary. Therefore, the

## 6 Discussion

While we are happy with the performance that our agent allowed the models to achieve, we have some limitations. The first is that due to time and computational constraints, we were unable to run on a variety of scenes, and had to stick with a relatively simple scene. This was disappointing because the whole point of using novelty-based rewards instead of a frontier-exploration type map was that it was more effective in larger scenes. Furthermore, the number of iterations that the full model was deployed on to get rewards was relatively small, meaning that COLMAP + PSNR could potentially have a higher PSNR, but couldn't reach it.

## 7 Conclusion

Novel view generating models such as 3DGS and Instant-NGP are an interesting model to optimize for in terms of agent performance, as they require a balance of view novelty and "redundancy" (so that feature matching between images can be done). By combining novelty and matching keypoint counts, we ended up with a model that has decent performance. In the future, we would like to apply our model to a larger dataset like HM3D to see how generalizable it is, and whether it would be easy to fine tune it to retain performance. When reflecting on the possible applications of our work, while it definitely could be useful in the industries we described above, we believe that it is most useful in finding a proxy measure for agentic novel image generation quality in the form of keypoints.

## 8 Team Contributions

- **Group Member 1: Brian Lee** Brian worked on implementing the basic PPO and logging, and helped work on the reward design and debugging the dependencies for the vision algorithms. Brian also worked on the paper and the poster.
- **Group Member 2: Allison Tee** Allison took the lead in setting up the vision algorithms and code that allowed all components of the vision algorithm to be used in PPO for reward and training, and helped design which rewards to use. Allison also worked on the paper and edited the poster.

**Changes from Proposal** We pivoted from doing agentic object search with 3DGS to more general scene representation / novel view generation with 3DGS and instant-NGP.

## References


Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. In *ACM SIGGRAPH Conf. Proc.* `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/` arXiv:2306.03816.

Xiaohan Lei, Yinhao Liang, Qi Xu, Xinshuo Wang, Yuchen Li, Roberto Martín-Martín, Li Fei-Fei, and Silvio Savarese. 2025. GaussNav: Gaussian Splatting for Visual Navigation. *arXiv preprint arXiv:2403.11625* (2025). `https://arxiv.org/abs/2403.11625` to appear in ICRA 2025.

Yuxuan Li, Fanbo Wu, Yichao Jiang, and Ziwei Deng. 2024. Semantic Gaussians: Open-Vocabulary Scene Understanding on 3D Gaussian Splatting Scenes. *arXiv preprint arXiv:2402.05149* (2024). `https://semantic-gaussians.github.io/`

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*. 405–421. `https://doi.org/10.1007/978-3-030-58452-8_24`