Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему:

«ACTIVE VIEWPOINT PLANNING FOR EFFICIENT 3D RECONSTRUCTION»

Виконала: студентка 4-го року навчання
освітньої програми «Прикладна математика»,
спеціальності 113 Прикладна математика

Кононович Софія Борисівна

Керівник: Кузьменко Д. О.
ст. викладач

Рецензент:

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК_____
(підпис)

«_____» _____ 20___р.

Київ - 2026

Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

ЗАТВЕРДЖУЮ
Зав.кафедри математики,
доцент, кандидат фіз.-мат. наук
_____ *Чорней Р.К.*
(підпис)
“____”_____ 2025

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
для кваліфікаційної роботи
студентці 4-го курсу, факультету інформатики
Кононович Софії Борисівній

Тема: Active Viewpoint Planning for Efficient 3D Reconstruction / Активне планування оглядових точок для ефективної 3D-реконструкції
Зміст кваліфікаційної роботи:

1. Introduction
2. Theoretical Foundations
3. Methods
4. Experiments and Results
5. Conclusions

Дата видачі “____”_____ 2025 Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Графік підготовки кваліфікаційної роботи до захисту

Графік узгоджено «_____» _____ 2025р.

№ з/п	Перелік робіт	Термін виконання етапу	Підпис наукового керівника	Дата ознайомлення наукового керівника	Примітка
1.	Отримання та ознайомлення з темою кваліфікаційної роботи.	01.10.2025			
2.	Робота з науковою літературою, опис основних означень.	20.01.2026			
3.	Написання коду для практичної частини.	01.03.2026			
4.	Перевірка й отримання результатів усіх експериментів.	15.03.2026			
5.	Завантаження чорнової версії кваліфікаційної роботи.	15.03.2026			
6.	Робота над текстовим оформленням теоретичної частини та одержаних результатів.	01.04.2026			
7.	Попередній захист кваліфікаційної роботи.	02.04.2026			
8.	Фінальна версія кваліфікаційної роботи .	20.05.2026			
9.	Захист кваліфікаційної роботи.	28.05.2026			

Науковий керівник: Кузьменко Д.О.

Виконавець кваліфікаційної роботи: Кононович С.Б.

Contents

1 Introduction **5**

2 Theoretical Foundations **7**

2.1 Neural Radiance Fields 7

2.1.1 The 3D Scene Reconstruction Problem 7

2.1.2 NeRF Architecture and Representation 7

2.1.3 Volume Rendering 8

2.1.4 Evaluation Metrics 9

2.2 Reinforcement Learning for Navigation 10

2.2.1 Markov Decision Processes in Embodied AI 11

2.2.2 The Next-Best-View Problem and Information Gain 12

2.2.3 Proximal Policy Optimization 12

3 Methods **14**

3.1 Environment and Simulator 14

3.1.1 Simulation Environment 15

3.1.2 Dataset Selection 16

3.1.3 Grid-Based Coverage Tracking and NavMesh Integration . . 16

3.2 Active Policy Design 17

3.2.1 Reward Function Design 17

3.2.2 Observation Space 19

3.3 Reconstruction Pipeline 20

3.4 Evaluation Protocol 21

3.4.1 Perception Metrics 21

3.4.2 Navigation Metrics 22

4 Experiments and Results **23**

4.1 Experimental Setup 23

4.1.1 Scene Selection and Evaluation Budget 23

4.1.2 Computational Resources 23

4.1.3 Software Environments 24

4.2 Experimental Design 24

4.2.1 External Baseline 24
4.2.2 Ablation Study 25
4.3 Experimental Results 26
4.3.1 Training Convergence 26
4.3.2 Quantitative Results 27
4.3.3 Trajectory Analysis 28
4.4 Generalization 29

5 Conclusions **30**

A PPO Training Hyperparameters **37**

B Qualitative Reconstruction Results **37**

Abstract

Efficient data acquisition is a key factor in neural 3D reconstruction, where redundant viewpoints can increase computational cost without improving reconstruction quality. We propose a reinforcement learning-based active viewpoint policy that optimises image collection for NeRF reconstruction under a fixed sensing budget. The agent is trained to maximise scene coverage while penalising redundant observations and inefficient trajectories, and learns to terminate data collection as information gain saturates. Evaluated on indoor scenes from the Replica dataset, our method achieves 33.33 dB PSNR while reducing the number of views by 25% (from 80 to 60), with a proportional reduction in scene acquisition time. Under the same sensing budget, it outperforms a coverage-only policy (33.05 dB) and a heuristic baseline (18.79 dB). The learned policy generalises to unseen scenes without retraining. These results indicate that learned data acquisition can improve both efficiency and reconstruction quality as an upstream component of 3D pipelines.

1 Introduction

Neural 3D reconstruction methods, such as Neural Radiance Fields (NeRF) [1], have significantly improved the quality of scene representation from sparse image observations. In recent years, the demand for high-quality 3D models has grown rapidly [2]. These approaches enable high-fidelity view synthesis and serve as a foundation for applications in robotics, augmented reality, and digital twin construction. However, their performance depends strongly on the quality and diversity of input viewpoints. In practice, image acquisition is often performed using heuristic trajectories or passive capture strategies, which can lead to redundant observations and inefficient use of resources.

Active viewpoint planning addresses this challenge by selecting informative camera poses to maximize scene coverage. Classical approaches formulate this as a Next-Best-View (NBV) problem, using geometric or information-theoretic criteria to guide exploration [3, 4, 5]. Recent advancements have integrated these principles into neural reconstruction, combining viewpoint selection with active representation learning and data collection [6, 7]. However, many of these methods rely on hand-crafted heuristics or prioritize coverage alone, without explicitly optimizing the efficiency of collected observations.

Reinforcement learning (RL) provides a flexible framework for learning exploration strategies in embodied environments, with prior work focusing on navigation, mapping, and task completion [8, 9]. While these works highlight the importance of viewpoint selection for downstream quality, the problem of learning policies that explicitly reduce redundancy while maintaining reconstruction fidelity remains relatively underexplored.

In this work, we address this gap by learning an active viewpoint policy that optimizes data acquisition for neural reconstruction. Our approach trains a Proximal Policy Optimization (PPO) agent [10] to maximize coverage while penalizing redundant viewpoints and inefficient trajectories. To optimize hardware utilization, we implement an asymmetric resource strategy: the navigational policy operates on a vectorized observation space on the CPU, while GPU resources are reserved for photorealistic rendering in Habitat-Sim [11] and high-resolution NeRF reconstruction. The resulting policy produces compact, informative im-

age sets that improve reconstruction quality under a fixed sensing budget. We summarize our contributions as follows:

- We propose a reinforcement learning-based framework for active viewpoint selection that explicitly balances coverage and redundancy.
- We integrate a precision coverage tracking mechanism with the simulator's NavMesh to ensure reward signals are grounded in true navigable scene geometry.
- We integrate the learned policy with a NeRF-based reconstruction pipeline and evaluate its impact on reconstruction quality.
- We demonstrate that the proposed method improves PSNR while reducing the number of captured views, and generalizes to unseen scenes.

2 Theoretical Foundations

2.1 Neural Radiance Fields

This section provides the theoretical basis of NeRF-based 3D reconstruction, introducing the classical formulation of the problem and the neural rendering approach.

2.1.1 The 3D Scene Reconstruction Problem

3D Reconstruction is the process of generating a 3-dimensional scene representation from a set of 2-dimensional inputs like images, video etc. We can formulate the classical 3D reconstruction problem [12, 1, 13] as: given a set of 2D observations $\{\mathbf{I}_i\}_{i=1}^{N}$ captured from known or estimated camera poses $\{\mathbf{P}_i\}_{i=1}^{N}$, recover a representation of the underlying 3D scene $\mathcal{S}$. Early approaches relied on Structure-from-Motion (SfM) [14] and Multi-View Stereo (MVS) [15] to detect sparse keypoints, match them across views, and triangulate a point cloud, which could subsequently be meshed and textured. These pipelines, despite robustness, often suffer from incomplete coverage in some regions and do not originally support novel-view synthesis.

Neural scene reconstruction methods address these limitations by formulating view synthesis as a photometric optimisation problem, minimising the error between rendered and observed images directly from posed RGB inputs, as detailed in Section 2.1.2.

2.1.2 NeRF Architecture and Representation

Neural Radiance Field (NeRF) is a technique for generating synthesised novel views from a sparse set of input images [1, 16]. Introduced by Mildenhall et al. at ECCV 2020, NeRF demonstrated that a static scene can be represented as a continuous 5D function and rendered at novel viewpoints with improved photorealism.

Formally, NeRF defines a function

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \longrightarrow (\mathbf{c}, \sigma), \tag{1}$$

where $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ is a 3D spatial location, $\mathbf{d} = (\theta, \phi) \in \mathbb{S}^2$ is the viewing direction expressed in spherical coordinates, $\mathbf{c} = (r, g, b) \in [0, 1]^3$ is the emitted RGB colour, and $\sigma \geq 0$ is the volume density, which acts as a differential opacity [1].

The weights Θ are optimised end-to-end solely from posed RGB images [1]. To represent high-frequency geometric detail, both position and direction are first lifted into a higher-dimensional space via a sinusoidal positional encoding [1, 17]:

$$\gamma(p) = \big(\sin(2^0\pi p),\, \cos(2^0\pi p),\, \ldots,\, \sin(2^{L-1}\pi p),\, \cos(2^{L-1}\pi p)\big), \tag{2}$$

where L controls the number of frequency bands. The encoded inputs are processed by a fully connected Multi-Layer Perceptron (MLP). To be precise, the first branch outputs volume density σ from position alone, and a second branch outputs view-dependent colour $\mathbf{c}$ conditioned on both position and direction [1]. The network parameters Θ are then optimised by minimising the photometric reconstruction loss over all camera rays:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \big\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \big\|_2^2, \tag{3}$$

where $\hat{C}(\mathbf{r})$ is the rendered colour (Section 2.1.3) and $C(\mathbf{r})$ is the ground-truth pixel colour.

Since this optimisation depends entirely on the set of posed input images, the choice of captured viewpoints directly determines the quality of the resulting reconstruction.

2.1.3 Volume Rendering

To understand why viewpoint quality matters, we must examine how NeRF converts its MLP outputs into a rendered image. The theoretical basis of NeRF is classical volume rendering, first formalised for computer graphics [18]. A camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is cast from camera origin $\mathbf{o}$ in direction $\mathbf{d}$. The expected colour $C(\mathbf{r})$ along the ray between near and far bounds t_n and t_f is given by the *volume rendering integral*:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma\big(\mathbf{r}(t)\big)\, \mathbf{c}\big(\mathbf{r}(t), \mathbf{d}\big)\, \mathrm{d}t, \tag{4}$$

where the accumulated transmittance $T(t)$ represents the probability that the ray travels from t_n to t without hitting any particle:

$$T(t) = \exp\left(- \int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)\, \mathrm{d}s \right). \tag{5}$$

In practice, the integral (4) is approximated by stratified sampling of K points $\{t_k\}_{k=1}^{K}$ along each ray [1]. Let $\delta_k = t_{k+1} - t_k$ be the distance between adjacent samples. The discrete colour estimate is

$$\hat{C}(\mathbf{r}) = \sum_{k=1}^{K} T_k \, \alpha_k \, \mathbf{c}_k, \tag{6}$$

where the discrete transmittance and alpha values are

$$T_k = \exp\left(- \sum_{j=1}^{k-1} \sigma_j \, \delta_j \right), \qquad \alpha_k = 1 - \exp(-\sigma_k \, \delta_k). \tag{7}$$

This formulation is fully differentiable, enabling gradient-based optimisation of the MLP parameters via Equation (3).

2.1.4 Evaluation Metrics

In this work we evaluate reconstruction quality using three complementary perceptual and signal-level metrics that are standard in the novel-view-synthesis literature [1, 19].

Peak Signal-to-Noise Ratio (PSNR) quantifies the pixel-level fidelity between a rendered image $\hat{\mathbf{I}}$ and its ground-truth counterpart $\mathbf{I}$, both of size $H \times W$. It is derived from the Mean Squared Error (MSE):

$$\mathrm{MSE} = \frac{1}{HW} \sum_{i,j} \big\| \hat{\mathbf{I}}_{ij} - \mathbf{I}_{ij} \big\|_2^2, \tag{8}$$

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}} \right) \quad [\mathrm{dB}], \tag{9}$$

where $\mathrm{MAX} = 1$ for normalised pixel values. Higher PSNR indicates lower distortion. Typical NeRF results on standard benchmarks achieve 28–33 dB [1]. A known limitation of PSNR is that it operates per-pixel and does not capture structured perceptual similarity.

Structural Similarity Index(SSIM) [20] measures perceived similarity along three axes: luminance l, contrast c, and structure s. For two image patches $\mathbf{x}$ and $\mathbf{y}$:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \underbrace{\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}}_{l(\mathbf{x},\mathbf{y})} \cdot \underbrace{\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}}_{c(\mathbf{x},\mathbf{y})} \cdot \underbrace{\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}}_{s(\mathbf{x},\mathbf{y})}, \tag{10}$$

where μ_x, μ_y are local means, σ_x^2, σ_y^2 are local variances, σ_{xy} is the cross-covariance, and C_1, C_2, C_3 are small stability constants [20]. The full-image SSIM is obtained by averaging over all patches, the values range from -1 to 1, with 1 indicating perfect structural agreement.

Neither PSNR nor SSIM correlates strongly with human perceptual judgement [19]. **Learned Perceptual Image Patch Similarity** (LPIPS) addresses this by measuring the ℓ_2 distance between deep feature activations extracted by a pre-trained convolutional network ϕ:

$$\mathrm{LPIPS}(\mathbf{x},\mathbf{y}) = \sum_{\ell} \frac{1}{H_\ell W_\ell} \sum_{h,w} \left\| w_\ell \odot \left(\hat{\phi}_\ell(\mathbf{x})_{hw} - \hat{\phi}_\ell(\mathbf{y})_{hw} \right) \right\|_2^2, \tag{11}$$

where $\hat{\phi}_\ell$ denotes channel-normalised activations at layer ℓ, w_ℓ are learned linear weights, and H_ℓ, W_ℓ are spatial dimensions [19]. Lower values of LPIPS indicate higher perceptual similarity, as it is a distance metric. Together, PSNR, SSIM, and LPIPS provide a complementary picture of reconstruction quality.

2.2 Reinforcement Learning for Navigation

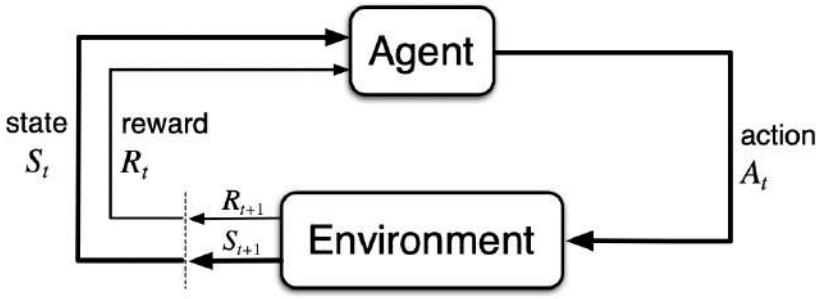

Figure 1: The agent–environment interaction in reinforcement learning [21].

Reinforcement learning (RL) is a branch of machine learning distinct from supervised and unsupervised learning. Rather than learning from labelled examples or

discovering structure in data, an RL agent learns by interacting with an environment to achieve a goal [21]. The core idea is that the agent must be able to sense the state of the environment to some extent and must be able to take actions that affect the state [21, 22] in order to maximize the cumulative reward it receives in the long run.

One of the main challenges in reinforcement learning is the need to balance *exploration*, which prioritizes gathering the information, and *exploitation*, which is greedy search. This trade-off is particularly relevant for active viewpoint planning. The agent must decide whether to revisit well-understood areas of the scene or explore unobserved regions that may introduce greater reconstruction improvement. In our work, this balance is managed implicitly through the PPO policy gradient framework and the design of the reward function, discussed in Sections 2.2.3 and 3.2.1 respectively.

2.2.1 Markov Decision Processes in Embodied AI

Embodied AI requires an agent to take actions, interact with an environment, and learn from active perception [11]. The standard mathematical framework for such sequential decision-making is the **Markov Decision Process** (MDP). A finite MDP can be described as a tuple (S, A, T, R, γ), where S is the state space, A is the action space, $T(s, a, s')$ is the transition function, R is the reward function, and $\gamma \in [0, 1)$ is the discount factor [21].

The agent's goal is to maximise the expected discounted cumulative return $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The reward signal defines this goal precisely: on each timestep, the environment sends the agent a single scalar reward, and it is critical that the reward function truly indicates what we want accomplished [21].

The MDP framework assumes the **Markov property**, meaning the future is independent of the past given the present state captured by the joint transition distribution:

$$p(s', r \mid s, a) = \Pr\{S_{t+1} = s',\ R_{t+1} = r \mid S_t = s,\ A_t = a\}. \tag{12}$$

In practice, an agent rarely has access to the full environment state. A Partially Observable MDP (POMDP) extends this framework by replacing direct state access with an observation $o \in \Omega$ from an observation function $O : S \times A \rightarrow$

$\Delta(\Omega)$ [22]. Crucially, POMDPs make no distinction between actions that change the world and actions that gather information [22]. This is directly relevant to AVP as each movement of the agent both changes its position in the scene and reduces uncertainty about unobserved regions.

We formulate our problem as a POMDP. The agent cannot observe the complete scene and instead receives a compact observation vector summarising local depth statistics and pose information and learns from it (Section 3.2.2).

2.2.2 The Next-Best-View Problem and Information Gain

The central question in exploration is: given what you know about the world, where should you move to gain as much new information as possible [4]? Early approaches answer this with frontier-based exploration, selecting boundaries between known and unknown space, however, these methods do not consider future consequences.

The problem of minimizing the views required to cover an object for 3D reconstruction is known as the View Planning Problem [2]: find a minimum subset $\mathcal{T} \subset S$ of viewpoints such that the scene Ω is sufficiently covered [2]:

$$v^* =_{v \in \mathcal{V}} \mathcal{I}(v) \quad \text{s.t.} \quad |\mathcal{T}| \leq B, \tag{13}$$

where $\mathcal{I}(v)$ is the information gain at viewpoint v and B is the sensing budget.

Solving VPP optimally is NP-hard, and the greedy algorithm is the best known polynomial-time approximation [23]. In order to find a better solution, we cast VPP as a Markovian Decision Process and solve the MDP in RL framework. As the result, instead of hand-crafting a heuristic, the agent learns a policy that accounts for long-term information gain.

2.2.3 Proximal Policy Optimization

Having framed viewpoint selection as an MDP, we need a method to solve it. A policy π_θ is trained to maximise the expected cumulative return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t R_t \right], \tag{14}$$

Policy gradient methods optimise $J(\theta)$ directly via the policy gradient theorem [21]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \hat{A}_t\right], \tag{15}$$

where $\hat{A}_t = Q^\pi(s_t, a_t) - V^\pi(s_t)$ is the advantage function, which is the relative benefit of action a_t over the expected value of state s_t.

Proximal Policy Optimization (PPO) [10] stabilises training by clipping the probability ratio between new and old policy $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\, 1{-}\epsilon,\, 1{+}\epsilon)\hat{A}_t\right)\right], \tag{16}$$

where ϵ controls the maximum policy change per update [10].

We select PPO for our task because the coverage reward is non-stationary, making on-policy learning appropriate. PPO operates efficiently with compact, low-dimensional observation vectors, enabling efficient resource utilization. Additionally, the simple first-order updates make ablation comparisons clean and reproducible.

3 Methods

In this chapter, we describe the technical framework and the multi-staged pipeline developed for active reconstruction, as shown in Figure 2. Our primary task involves utilizing a scene simulator where a trained agent autonomously captures viewpoints within a 3D space. Our methodology integrates high-performance 3D simulation, reinforcement learning for viewpoint selection, and NeRF for spatial synthesis. The goal of the system is to transit from raw spatial data to a refined, navigable, and reliable 3D model. This model provides the foundation for efficient scene reconstruction and can be further utilized for active viewpoint planning tasks.

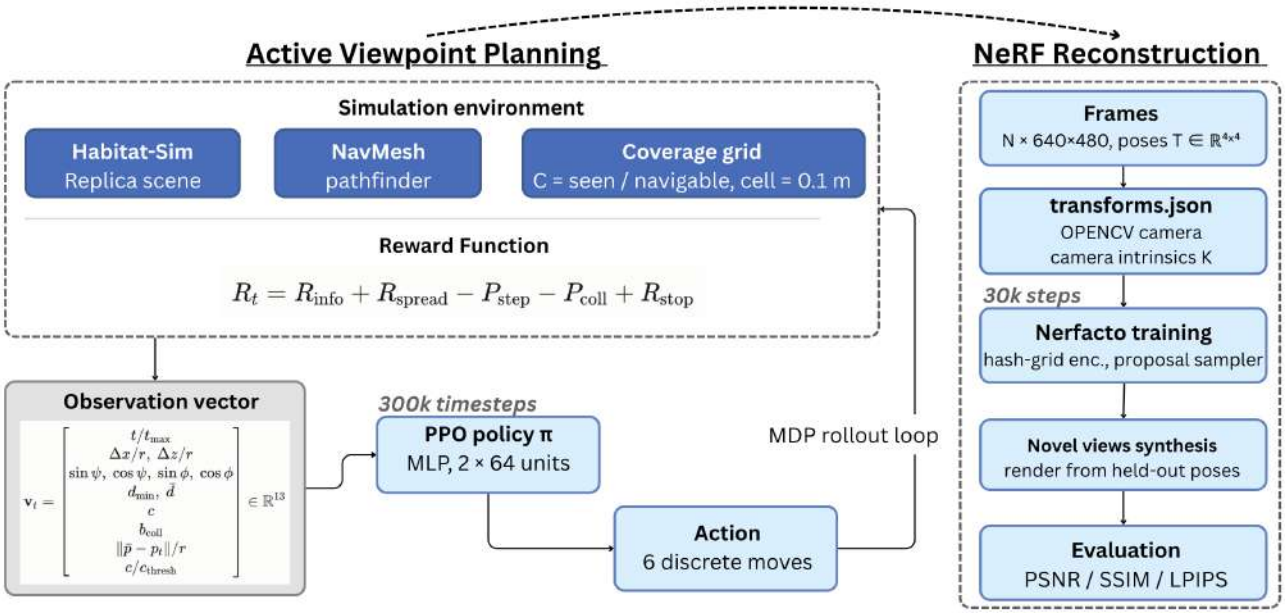

Figure 2: Overview of the proposed active viewpoint planning pipeline. Left: the RL training loop inside the simulation environment, showing the MDP rollout between the PPO policy and Habitat-Sim. Right: the NeRF reconstruction stage, from captured frames and camera poses through nerfacto training to metric evaluation against Habitat-Sim ground truth.

3.1 Environment and Simulator

In order to develop an autonomous agent capable of efficient 3D reconstruction, a controlled yet photorealistic environment is required. Simulation of the scene provides a safe, scalable, and computationally efficient alternative to real-world

robotics. Additionally, thousands of trials can be performed without extensive hardware or safety risks.

3.1.1 Simulation Environment

Habitat enables training embodied agents (virtual robots) in flexible, high-performance 3D simulation with multiple sensors and generic 3D dataset handling [11]. We utilized the Habitat-Sim Python API to manage the interaction between the agent and the 3D scene. This simulator is selected for the work due to its computational efficiency, as it can render photorealistic frames at thousands of frames per second (FPS) by leveraging shared memory to expose rendered data as Python tensors [11]. This architecture allows for a tight feedback loop between the agent's actions and its visual observations.

In our implementation, we utilized the following architectural abstractions provided by the Habitat API [11]:

- **Environment** is the top-level wrapper that abstracts the simulator's complexities, providing a standardised interface for resetting scenes and stepping through actions.
- **Task** extends the simulator's observation and action space with task-specific ones [11]. It allows us to extend the standard observation space with task-specific metrics, such as our custom coverage grid.
- **Action Space** is a discrete set of movements (Forward, Turn, Pitch, Stop) that the agent uses to navigate the 3D space.

We wrap Habitat-Sim in a custom **Gymnasium Environment** (`AVPEnv`) implementing the Gymnasium interface [24], which exposes a standardised `reset()` / `step()` API to the PPO trainer. The wrapper manages a 2D navigable floor grid for coverage tracking, a dual-resolution sensor pipeline (160×120 for policy training, 640×480 saved to disk for NeRF), and a discrete action space of six actions: move forward, turn left, turn right, look up, look down, and stop. This design keeps all scene-specific logic encapsulated, allowing the same policy to be evaluated on unseen rooms by simply changing the scene path at instantiation.

A key technical decision in our setup was the optimisation of hardware resources. While Habitat-Sim utilises the GPU for high-throughput rendering of the 3D scenes, we designed the agent's core to be computationally efficient. Instead of using a heavy Convolutional Neural Network (CNN) to process raw pixels during training, we utilised a vector-based observation space. By feeding the agent a 13-element vector (containing depth statistics, coordinates, and yaw/pitch), we were able to run the reinforcement learning (PPO) training on the CPU. This approach significantly reduced GPU memory utilisation and reserved those resources for the final high-resolution NeRF reconstruction.

3.1.2 Dataset Selection

For the scene geometry, we used the **Replica Dataset**, a collection of 18 highly photo-realistic 3D indoor scene reconstructions at room and building scale [25]. Replica is chosen over other datasets because it sets a high standard for texture and geometry resolution, including HDR information and realistic lighting [25]. This level of realism is essential for NeRF-based reconstruction, as the neural model requires consistent, high-quality visual data to accurately synthesize the scene from a sparse set of viewpoints.

3.1.3 Grid-Based Coverage Tracking and NavMesh Integration

In order to measure the success of a performed reconstruction and track exploration process, we implement a Grid-Based Coverage Tracker that maps 3D depth observations into a 2D occupancy grid, adapting the occupancy grid framework of Elfes [26] from obstacle detection to visual coverage.

A critical component of the tracker is the Navigation Mesh (NavMesh), an abstract data structure used in artificial intelligence applications to aid agents in pathfinding through complicated spaces [27]. The NavMesh was integrated into the simulator to define the reachable area and simplify complex 3D geometry into navigable polygonal surfaces. We integrated the NavMesh via the Habitat-Sim PathFinder API [28] to establish the ground truth of reachable space within the room. This ensures the agent is only evaluated on its ability to scan areas where a robot could physically stand, ignoring the volume of walls or furniture.

We discretised the 3D environment into a 2D occupancy grid G, where each cell represents a $0.1\,\text{m} \times 0.1\,\text{m}$ area of the floor plane. The true coverage ratio C represents the percentage of the reachable room that has been successfully captured by the agent's sensor, calculated as the intersection of observed cells and the navigable mask:

$$C = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} O_{i,j} \cdot N_{i,j}}{\sum_{i=1}^{M} \sum_{j=1}^{N} N_{i,j}} \tag{17}$$

where M, N are the dimensions of the global grid, $O_{i,j} \in \{0, 1\}$ is a binary indicator equal to 1 if cell (i, j) has been intersected by at least one camera ray, and $N_{i,j} \in \{0, 1\}$ is the NavMesh mask equal to 1 if the cell lies within the navigable bounds defined by the simulator [28].

This grid allows the agent to distinguish between new information and redundant views, which is the basis for the redundancy penalty reward discussed in Section 3.2.1. While coverage measures the quantity of observed space, it does not account for the efficiency of the trajectory. To evaluate the spatial distribution of the captured camera poses, we report the Average Nearest Neighbour Distance (ANND). A lower ANND indicates high spatial clustering (redundancy), whereas a higher ANND suggests a more diverse and efficient set of viewpoints:

$$\text{ANND} = \frac{1}{n} \sum_{k=1}^{n} \min_{j \neq k} \|\mathbf{p}_k - \mathbf{p}_j\| \tag{18}$$

where n is the total number of captured frames, and $\mathbf{p}_k$, $\mathbf{p}_j$ are the 3D position vectors of the k-th and j-th camera poses.

3.2 Active Policy Design

The main objective of active policy in this work is to transform the navigation from a stochastic random walk into goal-oriented exploration. We frame the algorithm as an MDP and train a policy π to maximise the expected cumulative reward $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ through interaction with Habitat-Sim.

3.2.1 Reward Function Design

The reward function is the core of our agent, as it directs the robot toward informative regions while penalising inefficient behaviour. Our reward logic is based

on the fundamental definition by Yamauchi, where exploration is the act of moving through an unknown environment while building a map that can be used for subsequent navigation [4]. In order to gain the most new information about the world, it is sufficient to move to the boundary between open space and uncharted territory [4]. Our reward function explicitly encourages the agent to seek out these frontiers. Mathematically, we implement this as a multi-component function:

$$R_t = R_{\text{info}} + R_{\text{spread}} - P_{\text{step}} - P_{\text{collision}} - P_{\text{redundancy}} + R_{\text{stop}}. \tag{19}$$

Following Charrow et al. [29], the **information gain** term rewards the rate of new coverage rather than raw coverage:

$$R_{\text{info}} = \frac{\alpha}{1 + \gamma\, v_{\text{cell}}} \, (\Delta c)^{\beta}, \tag{20}$$

where Δc is newly observed navigable cells, v_{cell} is prior visits to the current cell, and $\beta < 1$ introduces sublinear scaling. The saturation term discourages revisiting as the first visit yields full reward, the second 0.67, and so on. This structure forces the agent to navigate the exploitation–exploration trade-off, deciding whether to "improve the map in its local vicinity, or try to explore other parts of the environment" [29].

The **exploration spread** reward encourages the agent to maintain distance from its previously visited centroid [4]:

$$R_{\text{spread}} = \lambda_s \min\big(\|\mathbf{p}_t - \bar{\mathbf{p}}\|,\, r_{\max}\big), \tag{21}$$

where $\bar{\mathbf{p}}$ is the mean of all previously visited positions and $r_{\max}$ clips the maximum spread reward. This reward pushes the agent away from already-explored regions without requiring an explicit frontier map computation.

To ensure that the agent follows useful paths for real-world robotics, we include a **collision penalty** $P_{\text{collision}}$. At each step, the agent's displacement is measured. If the `move_forward` action is selected but the actual movement is below a small threshold (0.05 m), the agent is considered to have collided with geometry, and a fixed penalty is applied. Over training, this teaches the policy to avoid forward motion when facing an obstacle and instead favour turning actions to reorient. In real-world deployment such contact would risk sensor damage or task failure, so explicit collision-avoidance behaviour is essential [29].

The **step cost** P_{step} is a small constant applied at every step. It encourages the agent to prioritise the rate of discovery over time and provides the implicit pressure that makes the stop action attractive once coverage is sufficient.

A **redundancy penalty** $P_{\text{redundancy}}$ is applied to penalize the repeated capture of near-identical frames. At each step, the agent's current position and orientation are compared against all previously visited positions. If any past viewpoint lies within proximity threshold d_{red} and its yaw and pitch both fall within angular tolerance δ, the viewpoint is classified as redundant and a fixed value is deducted from the reward.

To avoid the Zeno-like paradox where information gain decreases geometrically as full coverage is approached [4], we implement a **discrete stop action**. When the agent fires this action after achieving a navigable coverage ratio $\geq \tau$, it receives a bonus:

$$R_{\text{stop}} = B_s + \lambda_e \, (T - t), \tag{22}$$

where B_s is a fixed termination bonus, λ_e scales the incentive for early stopping, T is the step budget, and t is the current step. This threshold is selected to allow task completion in the fewest steps possible while avoiding the diminishing-returns regime near full coverage. If the stop action is fired before the threshold is reached, the episode continues with a small penalty to ensure the agent does not permanently avoid exploring the stop action during training. All hyperparameters are set empirically and held fixed across scenes.

3.2.2 Observation Space

To achieve high-efficiency training without the computational overhead of processing raw RGB images, we use a 12-dimensional vectorized observation space. The observation vector $\mathbf{v} \in \mathbb{R}^{12}$ is structured as shown in Table 1.

By utilising this vectorised representation rather than raw pixel input, PPO training runs entirely on CPU at approximately 200–280 FPS, while the GPU is reserved exclusively for Habitat-Sim's photorealistic rendering during rollout and for NeRF reconstruction at inference time. This separation of compute responsibilities was intentional: the 12-dimensional vector captures all information the policy needs to navigate without requiring a convolutional network that would

Table 1: Observation vector components.

Index	Feature	Description
0	$t/t_{\max}$	Mission progress
1, 2	$\Delta x/r,\ \Delta z/r$	Relative displacement from episode start, normalised by room diameter r
3, 4	$\sin\psi,\ \cos\psi$	Yaw orientation (continuous encoding)
5, 6	$\sin\phi,\ \cos\phi$	Pitch orientation (continuous encoding)
7	$d_{\min}$	Minimum depth (nearest obstacle)
8	$\bar{d}$	Mean depth (scene complexity)
9	c	Coverage ratio $\in [0, 1]$
10	b_{coll}	Binary collision flag
11	$\|\bar{\mathbf{p}} - \mathbf{p}\|/r$	Normalised distance from visited centroid

compete with the renderer for GPU memory.

3.3 Reconstruction Pipeline

The final stage of the research focuses on the transition from image collection to 3D synthesis. Once the active policy has collected its viewpoint sequence, these images are processed through a neural reconstruction engine provided by Nerfstudio, a modular framework designed to simplify the development and deployment of Neural Radiance Fields [30]. In our pipeline, the output of each rollout (640×480 RGB frames and 4×4 camera-to-world pose matrices saved as `.txt` files) is converted into a `transforms.json` file following the OpenCV camera model convention, which Nerfstudio takes as input via its `nerfstudio-data` dataparser.

We selected the `nerfacto` model, which combines several state-of-the-art techniques to balance reconstruction speed and fidelity. The scene is represented as a continuous 5D function mapping each 3D point (x, y, z) and viewing direction (θ, ϕ) to an emitted colour and volume density [1]. The reconstruction process

consists of marching camera rays through the scene, sampling points along each ray, querying the neural network for colour and density at each point, and accumulating these values via classical volume rendering to produce a 2D image [1].

Two components make `nerfacto` particularly well-suited to our setting:

- **Hash-Grid Encoding** [31] allows the model to train in minutes rather than hours by using a specialised data structure to store scene information.
- **Proposal Sampler** [32] concentrates ray samples near actual surfaces such as walls and furniture rather than empty space, improving reconstruction quality in the bounded indoor scenes provided by the Replica dataset.

The **Dual-Resolution Sensor Pipeline** operates at 160×120 during PPO training to maximise simulation efficiency, as the policy operates on a 12-dimensional vector and is invariant to image resolution. During rollout, a second dedicated HD sensor captures 640×480 frames from the same physical camera pose for NeRF reconstruction. This design ensures the policy's navigational behaviour is identical between training and deployment and, most importantly, provides `nerfacto` with the high-frequency texture detail it requires to reconstruct sharp edges and surfaces accurately [1].

3.4 Evaluation Protocol

To verify the quality of the reconstruction and the efficiency of the navigation, we utilise two categories of metrics. For all evaluated methods, the collected images and camera poses are passed to a `nerfacto` model trained for a fixed 30,000 optimisation iterations, with all other training hyperparameters held constant across conditions.

3.4.1 Perception Metrics

We evaluate reconstruction quality using PSNR, SSIM, and LPIPS as defined in Section 2.1.3. For each method, we render 20 novel views from a fixed set of held-out test poses generated directly from Habitat-Sim ground truth, and compute all metrics against the ground-truth renders. This directly connects the active policy's viewpoint selection to final reconstruction fidelity.

3.4.2 Navigation Metrics

To evaluate the active viewpoint policy, on each rollout we measure:

- **Coverage Ratio (C):** defined in Section 3.1.2 (Equation 17), bounded to $[0, 1]$ and serving as the primary signal for the stop action threshold.
- **Average Nearest Neighbour Distance (ANND):** defined in Section 3.1.2 (Equation 18), used to evaluate spatial diversity of the collected viewpoints.
- **Images Saved (N):** the total number of frames captured before episode termination. For the full optimised policy this reflects when the agent voluntarily stopped; for ablation baselines it equals T. Lower N with equivalent C indicates higher efficiency.
- **Efficiency Index (EI):** defined as $\text{EI} = C/N$, capturing the trade-off between coverage and image count. Higher EI means more coverage per captured frame.
- **Coverage per View (CpV):** a training-time metric computed as $\text{CpV} = (C/\text{average episode length}) \times 100$, tracking how efficiently the policy converts steps into new coverage during training.

4 Experiments and Results

4.1 Experimental Setup

4.1.1 Scene Selection and Evaluation Budget

Room_0 from the Replica dataset was chosen as the primary training environment due to its moderately complex furniture layout which requires non-trivial navigation. Larger scenes such as `apartment_0` were excluded as their spatial scale requires substantially more steps to achieve comparable coverage fractions. Room_2 is used to assess policy generalisation on unseen data with no retraining.

A fixed budget of 80 frames was selected based on preliminary experiments with the coverage-driven RL policy. The task was for the coverage-maximising agent to achieve full coverage satisfying the commonly adopted threshold of 30 dB for perceptually acceptable NeRF reconstruction [1]. Budgets below 60 frames resulted in coverage below 80% and PSNR dropping under this threshold, while budgets above 80 produced diminishing returns in both metrics. As a result, the 80-frame budget was empirically selected to represent the minimum sufficient allocation for maximum performance and for future viewpoint reduction with the proposed policy.

4.1.2 Computational Resources

Reinforcement learning training, NeRF optimisation, and high-resolution rendering required significant GPU acceleration. All experiments were performed using a single NVIDIA RTX A5000 GPU with 24 GB of VRAM, which was necessary to handle the memory-intensive hash-grid encoding of the `nerfacto` model without downscaling. We utilised the RunPod cloud platform for on-demand computing, supported by a 100 GB network volume located in the EU-SE-1 data centre. The system was configured with 30 GB of system RAM and 8 virtual CPUs.

The NeRF reconstruction process of 30,000 iterations required approximately 60 - 90 minutes per scene to achieve geometric convergence and texture clarity. Evaluating the held-out test subset takes approximately 2 minutes per method. The RL policy (PPO) was trained for 300,000 time steps, taking approximately 30 minutes for 80 viewpoints setup to reach a stable reward plateau. Training

hyperparameters are listed in Appendix A, Table 4. All ablation variants use identical settings.

4.1.3 Software Environments

To organize the research pipeline and manage conflicting dependencies, we established two distinct Python virtual environments:

- **Habitat-RL Environment (Python 3.9)** is configured with `gymnasium` [24], `stable-baselines3`, and `habitat-sim` [11] to manage agent training and navigation.
- **Nerfstudio Environment (Python 3.11)** is focused on the reconstruction backend, utilising PyTorch with CUDA 11.8 to enable hardware-accelerated neural rendering [30].

Alongside the main frameworks, the following supporting libraries were utilised: NumPy, SciPy, pandas, matplotlib, Pillow (PIL), and numpy-quaternion.

4.2 Experimental Design

To demonstrate the effectiveness of the proposed solution, the evaluation is structured in two parts: an external baseline comparison and an internal ablation study. To ensure a fair comparison, all methods operate under a fixed maximum budget of 80 viewpoints. This budget was selected empirically as sufficient to achieve near-complete coverage of a Replica room-scale scene while remaining within a practical sensing limit. For each rollout, the collected images and camera poses are passed to a NeRF model. Below we describe each method and quantitative results follow in Section 3.2.

4.2.1 External Baseline

The external baseline implements a reactive, heuristic trajectory without a learned policy. At each step, the agent reads the depth value at the centre pixel of the current frame. If the distance to the nearest obstacle exceeds 1.0 m, the agent moves forward, and otherwise it randomly turns left or right with equal

probability. This kind of approach is simple obstacle-avoidance strategy common in early exploration literature [4]. In our work, it represents unplanned, passive capture. The action space is intentionally restricted to three actions, such as forward, turn left, and turn right, and all 80 viewpoints are utilised regardless of the coverage achieved. Formally, the action selection at step t is:

$$a_t = \begin{cases} \texttt{move\_forward} & \text{if } d_{\text{centre}} > d_{\text{thresh}}, \\ \texttt{turn\_left or turn\_right} & \text{otherwise (uniform random)}, \end{cases} \tag{23}$$

where d_{centre} is the depth reading at the image centre and $d_{\text{thresh}} = 1.0\,\text{m}$.

4.2.2 Ablation Study

The ablation study is divided into incremental components and results in the proposed reward function. Thus, we evaluate the contribution of each design decision. All variants use the same PPO policy architecture (300,000 timesteps, Room_0) with the same observation vector and action space described in Section **??**.

For **Ablation 1 (Coverage Only)** the reward function comes as a simple coverage signal:

$$R = \alpha_1 \, \Delta c_t - P_{\text{collision}}, \tag{24}$$

where Δc_t is the number of navigable floor cells newly observed at step t and $P_{\text{collision}}$ penalizes contact with geometry. The agent's purpose is to maximize the fraction of the scene, and, therefore, the full scene, not focusing on efficiency and diversity.

Ablation 2 (Coverage and Redundancy) is aimed to not only maximise coverage alone but to cut on redundant views. The reward function extends Ablation 1 with a redundancy penalty and a spread logic:

$$R = \alpha_2 \, \Delta c_t + \lambda_r \min(\|p_t - \bar{p}\|, r_{\max}) - P_{\text{redundancy}} - P_{\text{collision}}, \tag{25}$$

where $\bar{p}$ is the mean of all previously visited positions and $P_{\text{redundancy}}$ is applied when the current viewpoint is spatially and angularly similar to a prior observation (within d_{red} and angular tolerance δ in both yaw and pitch). This addresses the limitation we have observed for coverage-only policy, as it tends to cluster views in already-explored regions without explicit discouragement.

The full reward function for the **Proposed Policy** is described in Section 3.2.To summarise, it combines all reward components introduced in the ablation variants with a learned stop action, allowing the agent to terminate data collection once sufficient coverage is reached. This is the only variant capable of returning fewer than 80 views.

4.3 Experimental Results

4.3.1 Training Convergence

Figures 3 and 4 demonstrate the training dynamics of three policy variants across fixed timesteps. In subsequent qualitative analysis, Ablation 2 is omitted from trajectory visualisations and plots as it represents an intermediate design step and its contribution is fully captured in Table 2.

Figure 3 reports Coverage per View (CpV) over training. All variants show a consistent upward trend, indicating that the policy progressively learns to convert steps into new coverage more efficiently. The coverage-only variant plateaus around 1.1% CpV at approximately 150,000 timesteps. The coverage and redundancy variant reaches a slightly higher plateau of approximately 1.2% CpV at 200,000 timesteps, confirming that penalising redundant views encourages more efficient exploration. The proposed policy achieves the highest CpV of approximately 1.4%, continuing to improve steadily throughout the full 300,000 timestep budget, which indicates that the additional reward components require more experience to learn effectively.

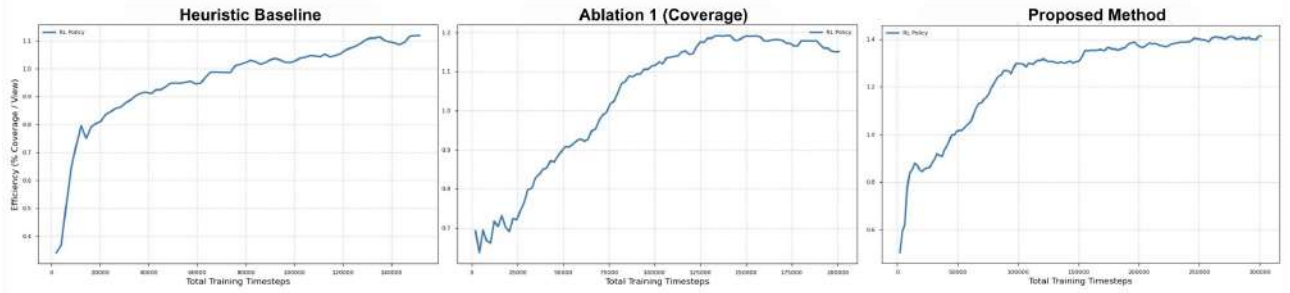

Figure 3: Coverage per View (CpV) training curves for (left) Ablation 1 – Coverage Only, (centre) Ablation 2 – Coverage and Redundancy, and (right) Proposed policy. All variants show improving efficiency over training, with the full policy achieving the highest final CpV.

Figure 4 shows the mean episode reward (Ep Rew Mean) for all three variants. All curves rise monotonically from negative initial values toward a stable positive plateau, confirming that the policy is learned under all three configurations. The proposed policy starts with the most negative initial reward, reflecting its stricter penalty structure, but converges to high final value (≈45).

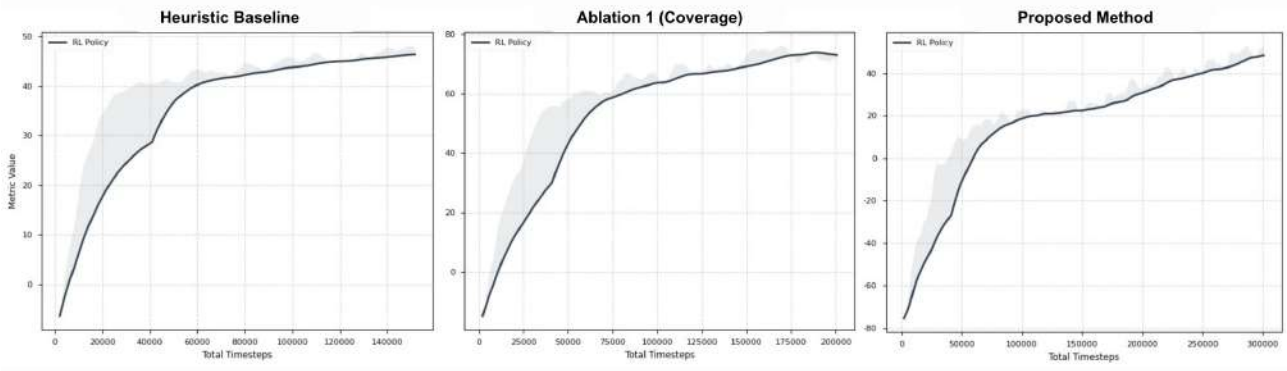

Figure 4: Mean episode reward (Ep Rew Mean) for (left) Ablation 1, (centre) Ablation 2, and (right) the Proposed policy

4.3.2 Quantitative Results

Table 2 compares the proposed method against a heuristic baseline and two ablation variants across reconstruction quality (PSNR, SSIM, LPIPS) and data acquisition efficiency (number of views, coverage).

The heuristic baseline covers only 26% of the scene and produces poor reconstruction quality (18.79 dB PSNR, LPIPS = 0.230), confirming that reactive trajectories yield highly redundant and uninformative observations. Switching to a coverage-driven RL policy raises scene coverage to 94% and substantially improves reconstruction fidelity (33.05 dB PSNR, LPIPS = 0.034).

Adding a redundancy penalty (Ablation 2) does not further improve PSNR or LPIPS as coverage drops slightly to 89% and LPIPS rises to 0.042. This suggests that penalising redundancy alone, without a corresponding spread incentive, can reduce viewpoint diversity rather than increase it.

The proposed full model resolves this by combining coverage, spread, and redundancy objectives with a learned stop action. It captures only 60 images, **25% fewer** than the 80-frame budget, directly reducing scene acquisition time to around **23 minutes**. Despite fewer views, it achieves the highest PSNR (33.33 dB)

Table 2: Results on Room_0. Proposed achieves the best PSNR and LPIPS with 25% fewer views.

Method	Views	Cov.	PSNR↑	SSIM↑	LPIPS↓
Heuristic	80	26%	18.79	0.893	0.230
RL (Coverage)	80	94%	33.05	0.947	0.034
RL (Cov.+Redund.)	80	89%	32.99	**0.948**	0.042
Proposed	**60**	88%	**33.33**	0.942	**0.027**

and lowest LPIPS (0.027) among all conditions. SSIM is marginally lower than Ablation 2 (0.942 vs. 0.948), likely due to the reduced number of views. The improvement in PSNR and LPIPS indicates that the selected viewpoints capture sharper, more geometrically informative observations.

These results suggest that explicitly modelling the trade-off between coverage, spatial diversity, and redundancy leads to more efficient data collection. The proposed method reduces both image count and acquisition time without sacrificing reconstruction quality, improving it across most metrics.

4.3.3 Trajectory Analysis

While maximizing coverage alone leads to strong improvements over heuristic baselines, it does not prevent the collection of redundant views, as visible in the clustered trajectories of Figure 5. It shows top-down views of agent positions (markers) and facing directions (arrows) within the Room_0 floor plane, with X and Z denoting the two horizontal spatial axes in metres. Each rollout starts from a randomly sampled navigable point, so the absolute coordinate ranges differ between methods while remaining within the same scene.

The results highlight the importance of balancing coverage and redundancy in active data acquisition. Incorporating a redundancy penalty encourages spatial diversity, resulting in more informative observations and higher reconstruction quality. The learned stop action further improves efficiency by avoiding diminishing returns near full coverage. This allows the agent to reduce the number of collected views without degrading reconstruction performance.

The efficiency gains are further reflected in reconstruction quality: the proposed method produces the sharpest novel view synthesis with the fewest captured

views, as shown in Appendix B.

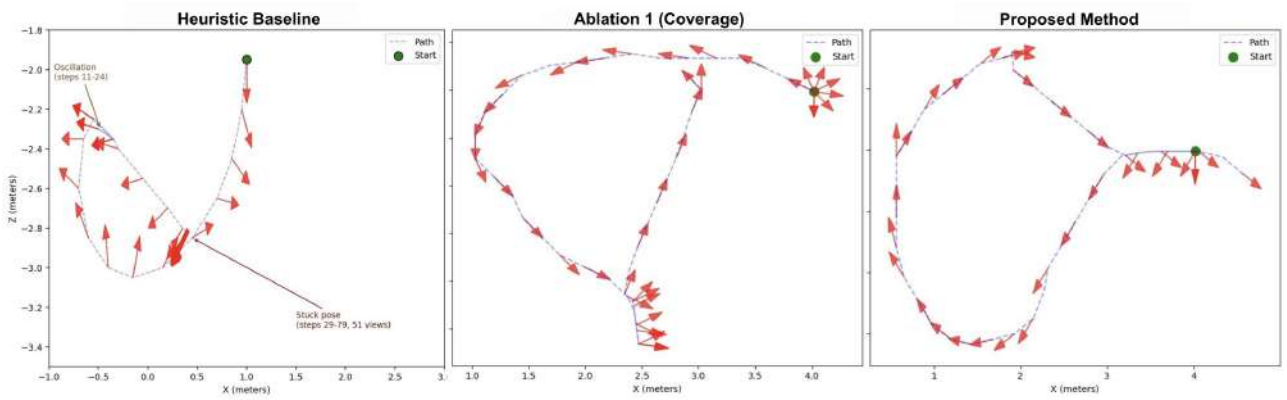

Figure 5: Top-down coverage maps (left to right): Heuristic, RL Coverage, and Proposed. The proposed policy distributes viewpoints more uniformly across the navigable space, avoiding the spatial clustering visible in the heuristic and coverage-only conditions.

4.4 Generalization

Table 3: Generalisation results. The policy is trained on Room_0 and evaluated on Room_2 without retraining.

Scene	Views	Cov.	PSNR↑	LPIPS↓
Room_0 (train)	60	88%	33.33	0.027
Room_2 (unseen)	44	93%	29.43	0.056

We evaluate the learned policy on Room_2, an unseen scene with a comparable navigable area to the training environment, without retraining. The policy achieves 93% coverage with 44 views (efficiency index 0.0212 vs. 0.0146 on Room_0) while maintaining spatial diversity (ANND 0.069), confirming that the spread objective transfers across layouts. Reconstruction quality is summarised in Table 3. Reconstruction quality reaches 29.43 dB PSNR and 0.922 SSIM. The PSNR drop is consistent with scene-specific appearance differences rather than a generalisation failure.

5 Conclusions

Our work investigates the role of active data acquisition in neural 3D reconstruction and shows that viewpoint selection can significantly influence downstream reconstruction quality. While prior approaches often treat image collection as a passive or heuristic process, our results demonstrate that a learned policy can produce more informative observations, improving reconstruction fidelity while reducing the number of required views.

A key strength of the proposed approach lies in explicitly modeling the trade-off between coverage and redundancy. By incorporating both factors into the reward design, the agent learns to prioritize spatial diversity and avoid duplicate observations, resulting in more efficient data collection. The inclusion of a stop action further enables the policy to operate under a practical sensing budget, terminating exploration once additional views provide limited benefit. Importantly, these improvements are achieved without modifying the underlying reconstruction model, suggesting that data acquisition itself is a meaningful optimization target within 3D pipelines. Qualitative inspection of rendered novel views confirms this, with the proposed method producing more high-quality reconstructions despite capturing 25% fewer images. The policy also generalises to an unseen Replica scene without retraining, indicating that the learned strategy is scene-agnostic rather than memorised.

At the same time, several limitations remain. First, the policy is trained and evaluated entirely in simulation, and its performance on real robotic platforms may be affected by domain gaps in perception, noisy state estimation, and the need to map discrete navigation actions to continuous robot control. Second, the reward function relies on hand-designed components, which may not fully capture the relationship between viewpoint selection and reconstruction quality. Third, the reconstruction stage is treated as a fixed downstream process, without feedback to the policy during training.

These limitations point to several directions for future work. An important extension is sim-to-real transfer, enabling deployment on physical robotic platforms. Another direction is the use of learned or differentiable objectives that directly reflect reconstruction quality, potentially allowing tighter coupling between view-

point selection and 3D modeling. Finally, integrating active data acquisition into end-to-end trainable 3D systems remains an open problem, where perception, planning, and reconstruction are jointly optimized.

Overall, our findings suggest that improving how data is collected can be as impactful as improving how it is modeled, and that active viewpoint planning is a promising component in the development of more efficient and scalable 3D reconstruction systems.

Acknowledgements

We would like to acknowledge the use of a large language model assistant (Claude Sonnet 4.6, Anthropic) during the preparation of this work. Its use was strictly limited to formatting, grammar correction, text organisation, and code style improvements, and was thoroughly tested by author. All ideas, analyses, experimental design, implementation, and intellectual content are solely our own.

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[2] Daryl Peralta, Joel Casimiro, Aldrin Michael Nilles, Justine Aletta Aguilar, Rowel Atienza, and Rhandley Cajote. Next-best view policy for 3d reconstruction. arXiv preprint arXiv:2008.12664, 2020.

[3] Clodagh Connolly. The determination of next best views. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1985.

[4] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997.

[5] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030, 1999.

[6] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[7] Cheng Jin et al. Nerf-nav: Visual navigation with nerf representations. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[8] Devendra Singh Chaplot, Saurabh Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[9] Santhosh K. Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[11] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[12] Tanveer Younis and Zhanglin Cheng. Sparse-view 3D reconstruction: Recent advances and open challenges. arXiv preprint, 2024.

[13] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3D scene reconstruction with the manhattan-world assumption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[14] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, 2015.

[16] Lin Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020.

[17] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[18] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Computer Graphics*, 18(3):165–174, 1984.

[19] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[22] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[23] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.

[24] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium. `https://github.com/Farama-Foundation/Gymnasium`, 2023.

[25] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797, 2019.

[26] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[27] Wikipedia contributors. Navigation mesh. `https://en.wikipedia.org/wiki/Navigation_mesh`, 2024. Accessed 2024.

[28] Meta AI. Habitat-sim pathfinder api. `https://aihabitat.org/docs/habitat-sim/habitat_sim.nav.html`, 2023. Accessed 2024.

[29] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[30] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022.

[32] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

A PPO Training Hyperparameters

Table 4 lists the hyperparameters used to train the PPO agent across all experimental conditions. All ablation variants use identical settings and only the reward function differs between conditions. Most values follow the Stable-Baselines3 defaults, which provide robust performance across a wide range of tasks. Two values were tuned through preliminary experiments: the entropy coefficient was increased from 0.0 to 0.02 to encourage broader action exploration during early training, particularly for the discrete stop action, and the batch size was raised from 64 to 512 to stabilise gradient estimates given the relatively long episode horizon (80 steps).

Table 4: PPO training hyperparameters.

Hyperparameter	Value
Algorithm	PPO (Stable-Baselines3)
Total timesteps	300,000
Learning rate	3e-4
Entropy coefficient	0.02
Clip range	0.2
n_steps	2048
n_epochs	10
Batch size	512
GAE λ	0.95
Discount γ	0.99

B Qualitative Reconstruction Results

Figure 6 shows novel view synthesis from the same held-out camera pose across three conditions. The qualitative differences reinforce the quantitative results in Table 2: the heuristic trajectory produces severe geometric distortion consistent with its low scene coverage, while both RL coverage and proposed method recover scene structure. The proposed method produces the sharpest reconstruction despite using 25% fewer input views, confirming that viewpoint diversity compensates for reduced image count.

Figure 6: Novel view synthesis from the same held-out camera pose. Heuristic (18.79 dB PSNR) fails to reconstruct scene geometry. RL Coverage (33.05 dB) recovers structure but retains blur. Proposed (33.33 dB) produces the sharpest reconstruction with 25% fewer views.