Supplementary Materials for Hestia: Voxel-Face-Aware Hierarchical Next-Best-View Acquisition for Efficient 3D Reconstruction

Supplementary Materials

In this supplementary material, we present the theoretical grounding for treating voxels as cubes in Sec. S1, additional quantitative results in Sec. S2, further qualitative results in Sec. S3, more real-world demonstration results in Sec. S4, details of the reward design in Sec. S5, the novelty of the proposed components in Sec. S6, the impact of spurious correlations in Sec. S7, dataset details and preparation in Sec. S8, the real-world system setup and associated costs in Sec. S9, training and testing details in Sec. S10, and limitations in Sec. S11. We hope readers understand that the main paper is limited to 8 pages, and therefore, we have included 11 additional sections in the supplementary material. We have made every effort to link the main paper with the supplementary material to improve the reading experience.

S1. Theoretical Grounding

Although treating a voxel as a cube rather than a point straightforwardly avoids overlooking surface geometry, we formalize the benefit through the following *constrained example* from a theoretical perspective. Consider a scene composed of k unit cubes and a 1–ray camera that emits a single ray per sample, where each ray is assumed to intersect one of the cubes in the scene. We contrast two sampling rules:

- **Scenario 1 (voxel as a point).** Continue sampling until every cube has been intersected by at least one ray.
- **Scenario 2 (voxel as a cube).** Continue sampling until every face of every cube has been intersected by at least one ray.

Our goal is to compare the expected face visibility achieved after the scenarios terminate. Hitting each of the k cubes at least once can be treated as a classical coupon–collector problem [2], whose expectation is:

$$k\Big(1+\frac{1}{2}+\cdots+\frac{1}{k}\Big)\approx k\ln k. \tag{S1}$$

Every ray that hits a cube intersects one of its six faces uniformly at random, so each ray can be viewed as a draw from:

$$N \;=\; 6k \tag{S2}$$

distinct faces. After n rays, the probability that a specific face is still unseen is:

$$\Big(1-\frac{1}{6k}\Big)^{n} \;\approx\; e^{-\frac{n}{6k}}, \qquad 6k \gg 1. \tag{S3}$$

Hence, through Eqs. (S1) and (S3), we know that after Scenario 1 stops, the ratio of the expected non-visible faces is:

$$e^{-\frac{k\ln k}{6k}} = k^{-\frac{1}{6}}. \tag{S4}$$

If $k = 8000$, roughly 22.3% of the faces remain unseen for Scenario 1, while Scenario 2 can cover all the faces. This theoretical result further motivates treating voxels as cubes rather than points when designing next-best-view policies.

S2. Benchmark Details

In this section, we present the detailed coverage ratio (CR), Chamfer Distance (CD), and area under the coverage ratio curve (AUC) from Tab. 1, broken down by each object position setting in Tabs. S1 to S3. Hestia outperforms other baselines across all object position settings in the OmniObject3D [23], Objaverse [4], and Houses3K [15] test splits. Moreover, Hestia is the only methods that demonstrate robust performance across all object position settings on all three datasets, with less than a 1% coverage ratio difference across different object configurations. For efficient 3D reconstruction, Fig. S1 shows that Hestia outperforms other methods by nearly 10% and 5% in the first five and fifteen captures, respectively. This efficiency is especially important in real-world power-constrained scenarios, where a robot or agent may quickly exhaust its battery.

S3. Qualitative Results

In this section, we present additional qualitative results for OmniObject3D [23], Objaverse [4], and Houses3K [15] in Figs. S2 to S6. Fig. S2 shows that Hestia's viewpoints successfully reconstruct the point clouds of real-world scanned objects, while other baselines often miss parts and perform less robustly across various object shapes. Specifically, other baselines miss the starfish's arms, the sofa's front or bottom, the plant's pot, the statue's head or stand, the table's surface, and the durian's flesh. All

Method	OmniObject3D Test														
	(0, 0)			(4, 4)			(4, -4)			(-4, 4)			(-4, -4)		
	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑
NeU-NBV [9]	88	24	83	73	51	70	75	75	70	72	53	69	76	38	71
ScanRL [15]	87	21	80	79	37	76	84	21	78	72	47	71	76	35	72
MACARONS [7]	86	19	75	93	13	86	91	14	83	94	10	88	80	29	69
GenNBV [3]	92	12	88	92	14	86	91	15	83	94	10	89	94	9	88
GenNBV (Rep.) [3]	93	10	87	94	9	89	92	12	85	94	10	88	92	11	85
Hestia (Ours)	97	4	94	97	4	93	97	5	92	97	4	93	96	5	93

Table S1. **CR (%) / CD (cm) / AUC (%) comparison on the OmniObject3D test set.** Hestia outperforms other methods and is more robust across different object position settings.

Method	Objaverse Test														
	(0, 0)			(4, 4)			(4, -4)			(-4, 4)			(-4, -4)		
	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑
NeU-NBV [9]	88	20	83	76	41	74	77	31	72	77	41	74	78	31	72
ScanRL [15]	87	18	82	80	31	78	82	23	77	77	37	76	80	27	76
MACARONS [7]	88	17	79	91	16	86	91	15	83	75	42	70	78	30	64
GenNBV [3]	94	11	89	90	15	85	89	17	82	93	12	89	91	13	86
GenNBV (Rep.) [3]	94	11	88	92	12	88	90	15	82	92	12	88	90	14	84
Hestia (Ours)	96	7	93	96	7	92	96	6	91	96	7	93	96	8	91

Table S2. **CR (%) / CD (cm) / AUC (%) comparison on the Objaverse test set.** Hestia outperforms other methods and is more robust across different object position settings.

Method	Houses3K Test														
	(0, 0)			(4, 4)			(4, -4)			(-4, 4)			(-4, -4)		
	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑	CR↑	CD↓	AUC↑
NeU-NBV [9]	85	26	80	73	52	71	81	30	77	72	53	70	80	32	76
ScanRL [15]	86	21	79	77	40	75	88	19	82	72	50	69	81	28	76
MACARONS [7]	87	20	78	92	16	87	93	14	85	71	51	66	81	30	68
GenNBV [3]	94	10	89	90	18	83	91	16	84	94	12	89	93	12	89
GenNBV (Rep.) [3]	94	12	88	95	10	90	94	13	88	94	11	89	91	14	88
Hestia (Ours)	97	5	96	97	5	93	97	7	93	97	5	94	97	7	93

Table S3. **CR (%) / CD (cm) / AUC (%) comparison on the Houses3K test set.** Hestia outperforms other methods and is more robust across different object position settings.

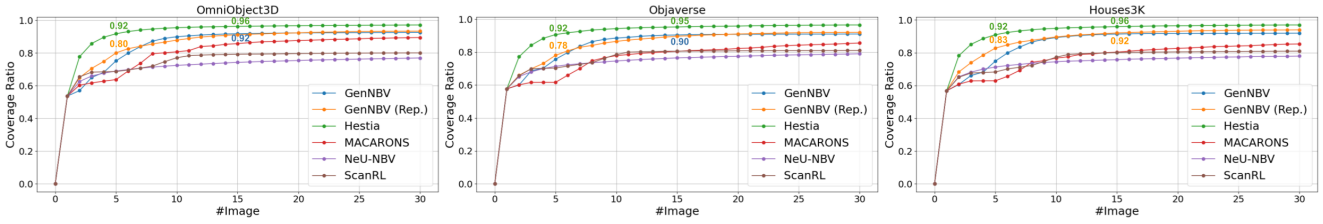

Figure S1. **CR curves on three datasets.** Hestia outperforms prior approaches by nearly 10% and 5% in the first five captures and the first fifteen captures, respectively. The efficiency is particularly significant in real-world power-constrained scenarios, where a robot or agent may run out of battery in a short time.

these diverse missing parts are captured by Hestia. Fig. S3 shows that Hestia's viewpoints robustly cover various object shapes, while other baselines often miss finer details or parts underneath. Specifically, other methods miss the wooden stand's legs, the Lego man's face or arms, the lamp's lampshade or neck, the underside of the wooden log, and the tree's leaves. In contrast, Hestia successfully captures all these diverse and challenging parts. Fig. S4 shows that Hestia's viewpoints can effectively capture building-like structures, while other methods often miss features such as pillars, roof soffits, or windows. Fig. S5 shows that Hestia can capture the complex scene well. Fig. S6 shows that Hestia is the only method that achieves consistent performance across different object position settings. These visualization results validate that our proposed core components, including dataset choice, observation design, action space, reward calculation, and learning scheme, form a significant foundation for the tasks and thereby bring a non-marginal impact.

S4. Real-World Results

In this section, we present real-world images captured using Hestia operating within the drone system (see Sec. S9). Fig. 5 demonstrates that Hestia performs well in real-world object-centric scenes, even when the depth camera is unavailable, for both shifted and non-shifted cases. Notably, without a depth camera to synchronize the virtual and real world, we manually set up three viewpoints, which are deliberately placed close to each other (e.g., the red boxes in Fig. 5). In addition, it is reasonable that some next-best viewpoints appear similar because we use a multi-view depth predictor [6, 22] to convert RGB images into depth maps. Therefore, it is common for certain viewpoints to overlap in order to obtain depth and update the input state. Fig. S7 shows that Hestia robustly handles various real-world object shapes. These results demonstrate that Hestia surpasses the prior works [3, 7, 9, 12, 14, 15, 17, 19, 25], which have not been validated in real-world environments. In addition, Hestia is suitable for use as a viewpoint predictor for real-world applications.

S5. Reward Design

In this section, we review the design of our reward function. The reward function in Hestia is formulated as

$$r_t = R(s_t, a_t) = r_{\text{coverage}}(s_t, a_t) + r_{\text{constraint}}(s_t, a_t). \quad \text{(S5)}$$

To promote the exploration of previously unseen surfaces, we define the positive reward as

$$r_{\text{coverage}}(s_t, a_t) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{6} \left(F_t^{i,j} - F_{t-1}^{i,j} \right) \cdot M_{\text{col}}}{N \cdot 6} \cdot 0.3 \quad \text{(S6)}$$

where $F_t^{i,j}$ and $F_{t-1}^{i,j}$ denote the visibility status of the j-th face of the i-th voxel at time t and $t-1$, respectively. The variable $M_{\text{col}} \in \{0, 1\}$ acts as a collision indicator, set to 0 when a collision occurs, thereby nullifying any potential reward for unsafe actions. This reward is computed based on the increment in newly visible voxel faces at the current step. By focusing on the increment rather than the accumulated visibility, the agent is better able to associate rewards with the immediate effects of its actions. To discourage unsafe or invalid decisions, we define a penalty as

$$r_{\text{constraint}}(s_t, a_t) = \begin{cases} -0.01, & \text{if } r_{\text{coverage}}(s_t, a_t) = 0, \\ & \text{or } a_t[2] > H_t, \\ & \text{or } a_t \in \text{non-free voxels}, \\ 0, & \text{otherwise.} \end{cases} \quad \text{(S7)}$$

In particular, a negative reward is assigned if the agent fails to reveal any new faces, attempts to move above the maximum allowable flight height H_t, or selects a viewpoint located within non-free voxels. To ensure a balance between positive and negative rewards, the positive reward is scaled by a factor of 0.3. This weighting is based on the observation that the maximum face ratio is 1 and each episode ends after 50 steps. As a result, the total possible positive reward is approximately $0.3 \times 1 = 0.3$, which roughly aligns with the maximum overall value of the negative penalty $0.01 \times 50 = 0.5$. If the episode ends earlier, such as after 30 steps, this reward structure maintains a perfect balance.

S6. Novelty Justification

This section elaborates on the novelty of Hestia. Hestia is a generalizable next-best-view planner that can predict five-degree-of-freedom viewpoints and model a drone as an agent. Therefore, we mainly focus on comparing methods that also predict five-degree-of-freedom viewpoints or assume a drone as an agent. Unlike prior approaches, Hestia systematically addresses the next-best-view task by introducing core components such as dataset selection, observation design, action space formulation, reward computation, and learning schemes. Together, these elements form a comprehensive and unified foundation for the planner (see Sec. 4.7). As shown in Tab. S4, compared to online-learning methods [7, 11, 12, 14, 17, 19, 25], Hestia avoids the need to sample candidate views or perform online optimization. This results in greater flexibility in viewpoint prediction and supports real-time inference. Additionally, in comparison to generalizable methods [3, 9, 15], Hestia is trained on a significantly larger and more diverse dataset, enabling it to generalize robustly to a wide variety of object shapes during testing. Hestia is also the only method that consistently performs well under different object configurations, as demonstrated in Tabs. S1 to S3. These advantages stem from the key innovations in our design, including treating

Figure S2. **Qualitative comparison on OmniObject3D [23].** Hestia's viewpoints reconstruct the point clouds of real-world scanned objects accurately, while other baselines exhibit less robustness across various object shapes, often missing parts in the reconstructed point clouds.

voxels as cubes rather than points, employing a hierarchical structure to manage the complexity of the action space, and using a greedy learning scheme to mitigate spurious correlations. Notably, the purpose of Hestia's hierarchical structure is to address the high-dimensional continuous action search space problem in reinforcement learning-based generalizable next-best-view planning, which is fundamentally different from traditional methods that use hierarchical structures to move along frontiers. One of the most recent works [3] still lacks the designs we propose.

S7. Spurious Correlation

In this section, we present the spurious correlation caused by future positive rewards in the task. Spurious correlation has been widely observed across various tasks [8, 10]. In our task, we find that using a large discount factor and future goal rewards can lead to false associations between current actions and their rewards. This creates an illusion for the reinforcement learning agent that the current action is beneficial, even when there is no information gain (e.g., empty views as shown in Fig. S8) resulting from the cur-

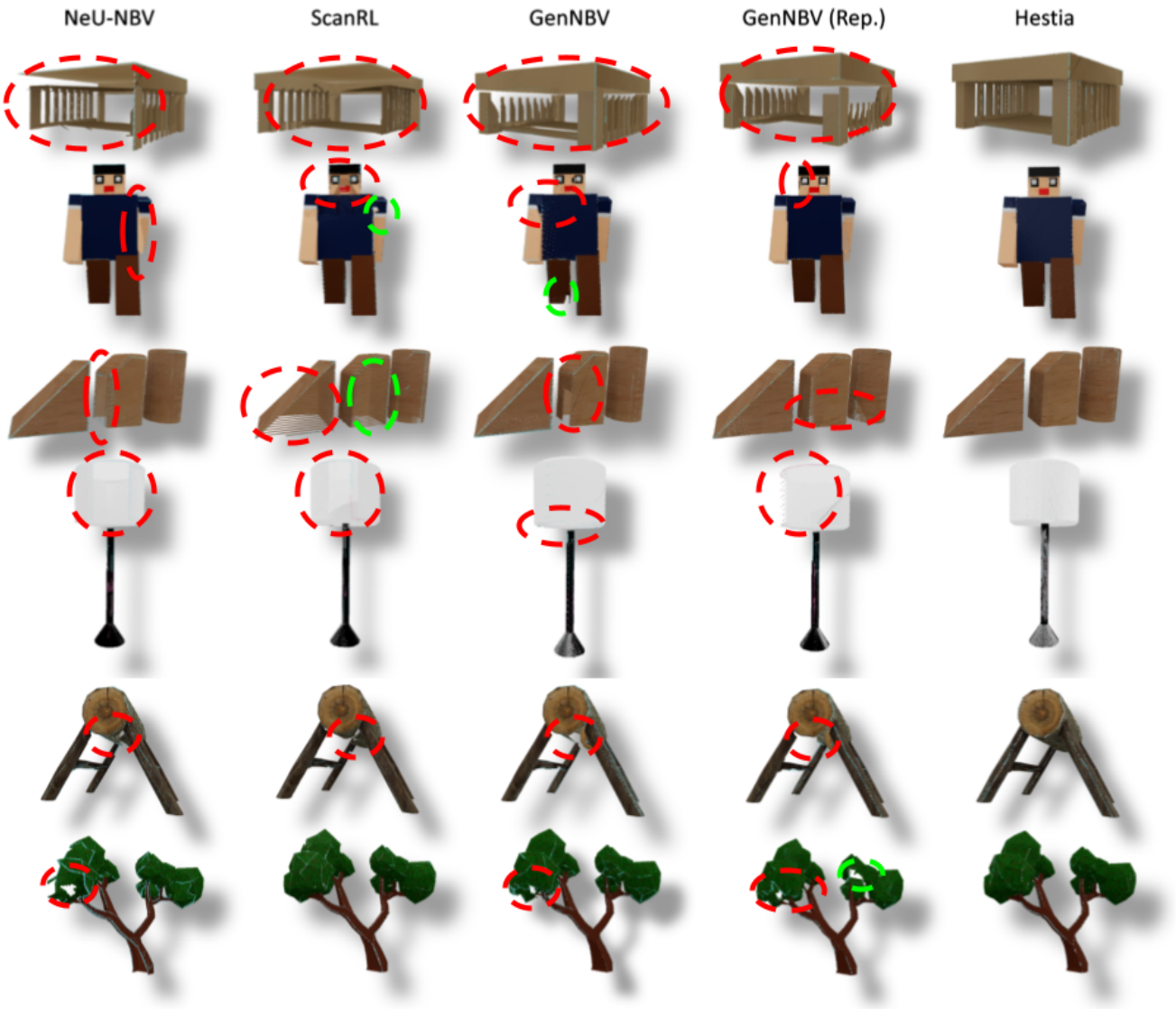

Figure S3. **Qualitative comparison on Objaverse [4].** Hestia's viewpoints successfully reconstruct diverse and complex object shapes, while other baselines exhibit less robustness, often missing self-occluded regions or parts that require bottom-up viewpoints.

rent action. Enabling the close-greedy design mitigates this issue as shown in Fig. S8.

S8. Datasets

This section briefly introduces the three main datasets used in the Hestia benchmark: Houses3K [15], Objaverse [4, 5], and OmniObject3D [23].

S8.1. Introduction to Datasets

Houses3K Dataset. Houses3K [15] is designed for next-best-view policy learning. The dataset contains 600 distinct buildings, each rendered with five texture variants, yielding a total of 3,000 FBX models. Many buildings feature challenging self-occlusions, such as roof soffits, that can be fully observed only from bottom-up viewpoints (see Fig. S9). However, because the dataset includes only a single cube-like object category (e.g., buildings), its diversity is limited, which may hinder the ability of next-best-view policies trained on Houses3K to generalize to other structures or everyday objects.

Objaverse Dataset. Objaverse [4] is one of the largest open 3D datasets, containing more than 800,000 shapes across at least 18 high-level categories, including furniture, vehicles, animals, and plants (see Fig. S9). Each category is further divided into several subcategories. The dataset's scale and diversity make it particularly well-suited for foundation

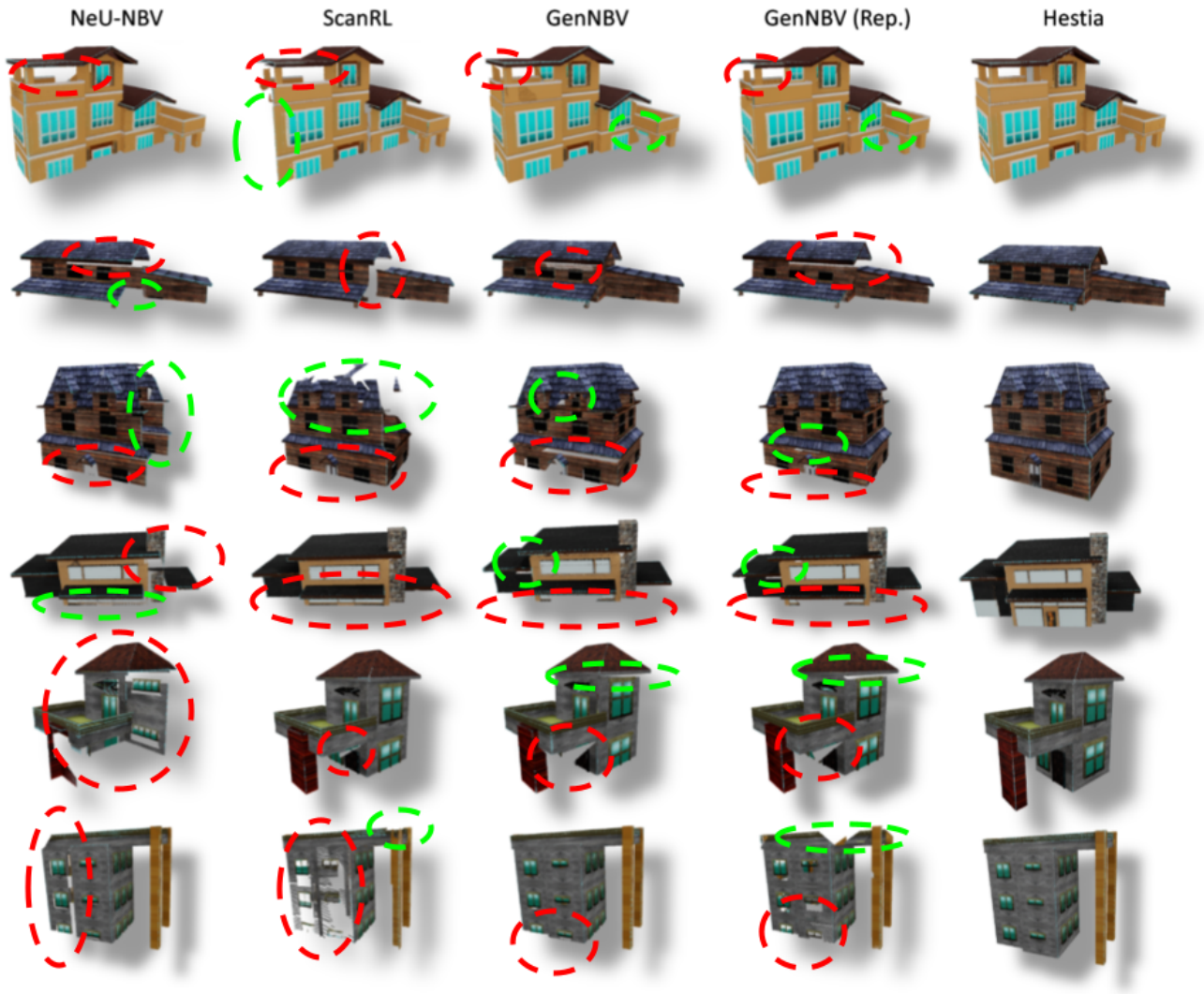

Figure S4. **Qualitative comparison on Houses3K [15].** Compared to the baselines, the point clouds reconstructed from the depth maps collected by Hestia capture finer details, such as roof soffits, pillars, and windows, particularly in self-occluded areas.

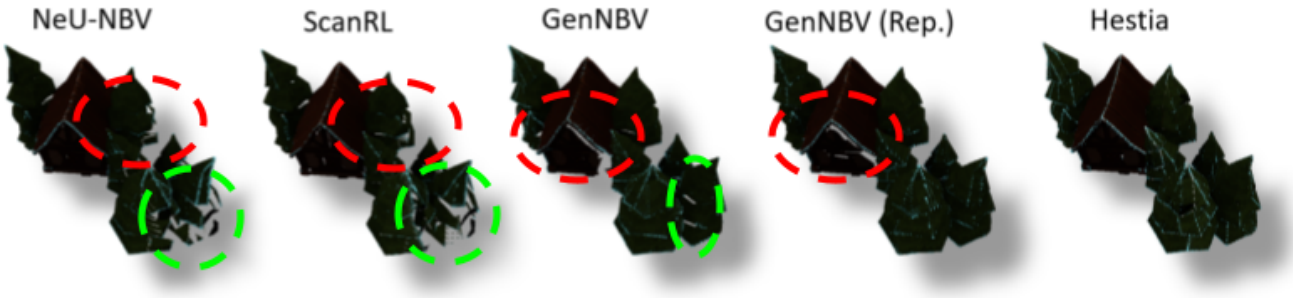

Figure S5. **Reconstruction on a complex scene.** Hestia captures the scene well.

model research, especially for 3D generative models. To the best of our knowledge, we are the first to introduce Objaverse for next-best-view policy learning. Its large-scale and diverse object coverage enables training next-best-view

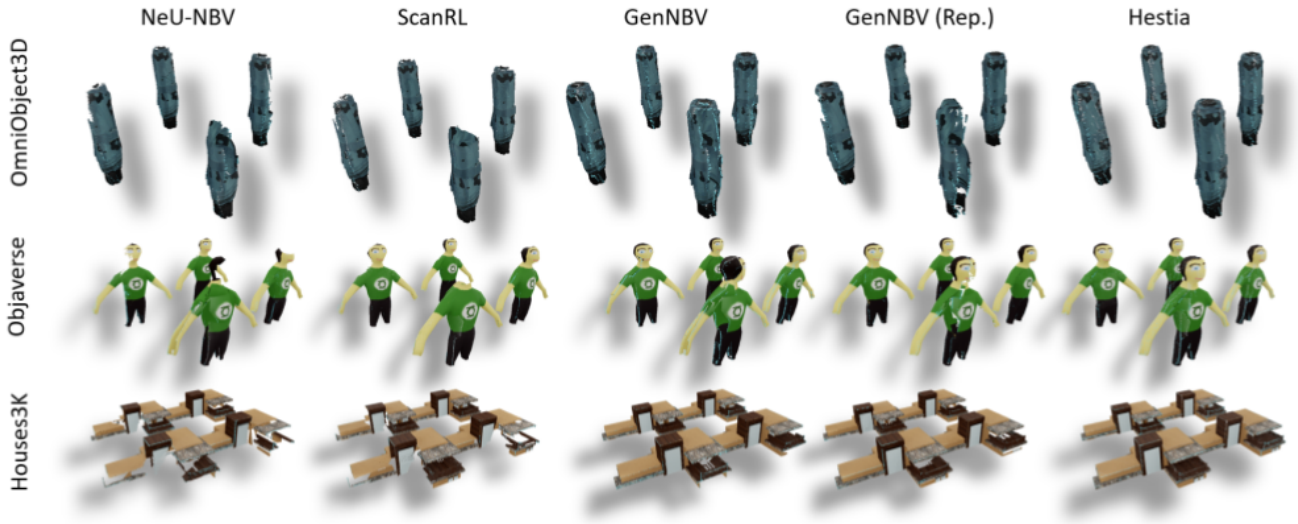

Figure S6. **Qualitative comparison of objects at the four corners.** Hestia performs robustly across different object position settings, while other baselines fail to maintain consistent performance across positions.

Method	No Cand. View	No Online Learn.	Voxel as Cube	Hier. Str.	Greedy	Div. Data.	Robust for Shift.	Real-World Demo.
Active3D [12]	✗	✗	–	✗	–	✗	–	✗
NeRF-En [19]	✗	✗	–	✗	–	✗	–	✗
ActiveNeRF [14]	✗	✗	–	✗	–	✗	–	✗
NeurAR [17]	✗	✗	–	✗	–	✗	–	✗
EfficientView [24]	✗	✗	–	✗	–	✗	–	✓
UnGuide [11]	✗	✗	–	✗	–	✗	–	✓
SEER [20]	✗	✗	–	✓	–	✗	–	✓
ActiveRMAP [25]	✗	✗	–	✗	–	✗	–	✗
MACARONS [7]	✗	✗	–	✗	–	–	–	✗
NeU-NBV [9]	✗	✓	–	✗	–	✗	✗	✗
ScanRL [15]	✓	✓	–	✗	✗	✗	✗	✗
GenNBV [3]	✓	✓	✗	✗	✗	✗	✗	✗
Hestia	✓	✓	✓	✓	✓	✓	✓	✓

Table S4. **Comparison of learning-based next-best-view methods.** Compared to online learning methods, Hestia achieves real-time inference speed. Compared to generalizable methods that predict five-degree-of-freedom viewpoints or assume a drone as the agent, Hestia exhibits robustness across different object position settings and demonstrates feasibility in real-world object-centric scenes.

policies that perform robustly across a wide range of categories and shapes.

OmniObject3D Dataset. OmniObject3D [23] is a high-quality 3D object dataset collected through real-world scanning, consisting of approximately 6,000 objects across more than 190 categories (see Fig. S9). Unlike synthetic datasets, OmniObject3D captures real-world geometry and texture details using high-resolution 2D and 3D sensors. It provides accurate geometry and realistic material properties, making it commonly used for evaluating real-world transferability in vision tasks such as novel-view synthesis. In this paper, we introduce OmniObject3D for benchmark purposes.

S8.2. Dataset Preparation

We propose using Objaverse [4, 5] as the training dataset to ensure a diverse range of shapes during training (see Figs. S10 and S11). To achieve this, we filter out large meshes and download the remaining mesh files from Objaverse [4, 5], resulting in a dataset comprising 120,000 shapes. For each shape, we generate the occupancy grid

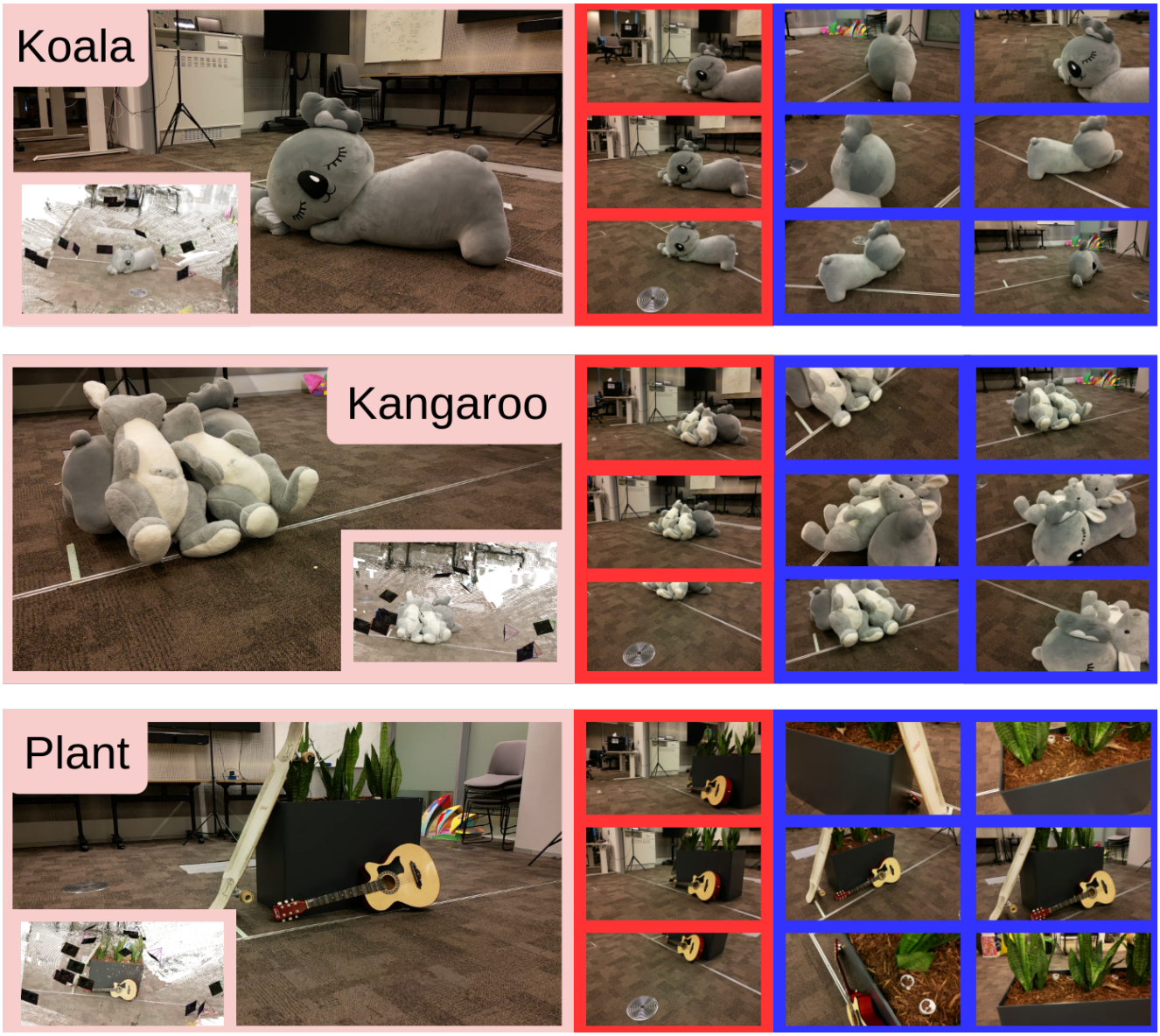

Figure S7. **Real-world demonstration of various object-centric scenes.** Hestia operates in real-world object-centric scenes starting from three initial viewpoints (red rectangles) and predicts the next-best viewpoints for capture (blue rectangles for the first six views), even when the depth camera is unavailable. Point cloud reconstruction results are shown on the left, with black rectangles representing the camera poses.

and point cloud using Open3D [26]. To remove invisible voxels and points, we perform a breadth-first search (BFS) starting from external free voxels, retaining only reachable occupancy voxels and points as the ground truth. The visible faces of each voxel are identified by examining the occupancy states of neighboring voxels. We apply PCA [1] to reduce the point cloud to three components and use k-means clustering [13] to group the 120,000 shapes into 30,100 clusters. The cluster centers of 30,000 clusters are designated as training data, while the remaining cluster centers are used for testing. The same procedure is used to create 256 training samples and 100 test samples from the Houses3K dataset [15] for our benchmark. Our processed training set is two orders of magnitude larger than those used in previous studies [3, 9, 15] and includes at least 18 more categories than prior datasets [3, 15]. As for OmniObject3D [23], due to limited storage space, we randomly select one shape per category for evaluation, resulting in approximately 200 test samples.

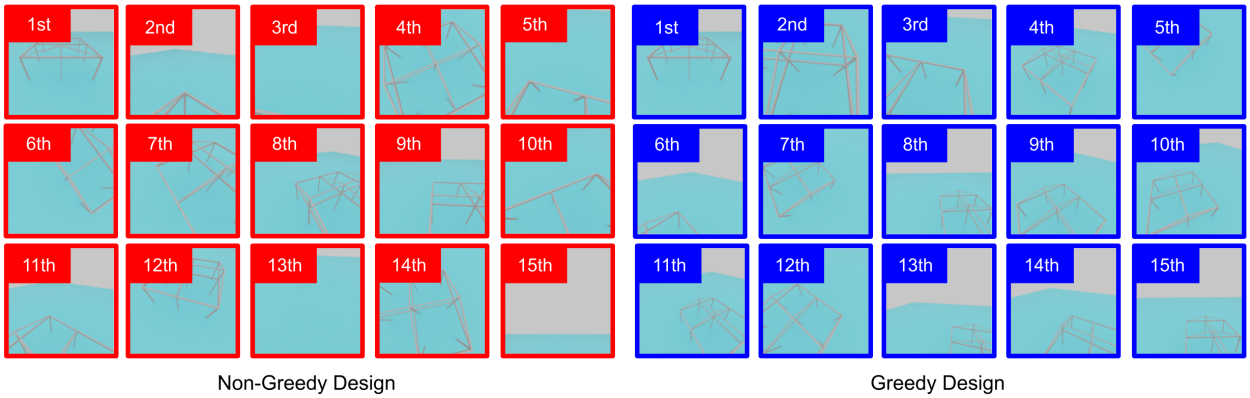

Figure S8. **Spurious correlation.** In this study, spurious correlation refers to the assignment of future positive rewards to current non-beneficial actions, resulting in suboptimal viewpoint predictions. For instance, the third, thirteenth, and fifteenth viewpoints are empty in the non-greedy design, while enabling the close-greedy design alleviates this issue.

Figure S9. **Sample data from Houses3K [15], Objaverse [4], and OmniObject3D [23].** Houses3K features building shapes with challenging self-occlusions, such as roof soffits. Objaverse includes a diverse range of object shapes. OmniObject3D contains high-quality real-world 3D scans.

S9. Real-World Drone System

This section includes the setup and pseudo code of the real-world drone system we used. By integrating Hestia into the real-world drone system (see Fig. S12), Hestia demonstrates its feasibility under practical scenarios.

S9.1. Real-World System Overview

To demonstrate Hestia's feasibility in a real-world environment, we use a real-world system (see Fig. S13), where a drone equipped with an RGB camera moves to the next-best viewpoint predicted by Hestia to capture images of an object. The system uses four HTC Lighthouse base stations and a Crazyflie 2.1 for localization and transmits images to the ground control station via wireless commu-

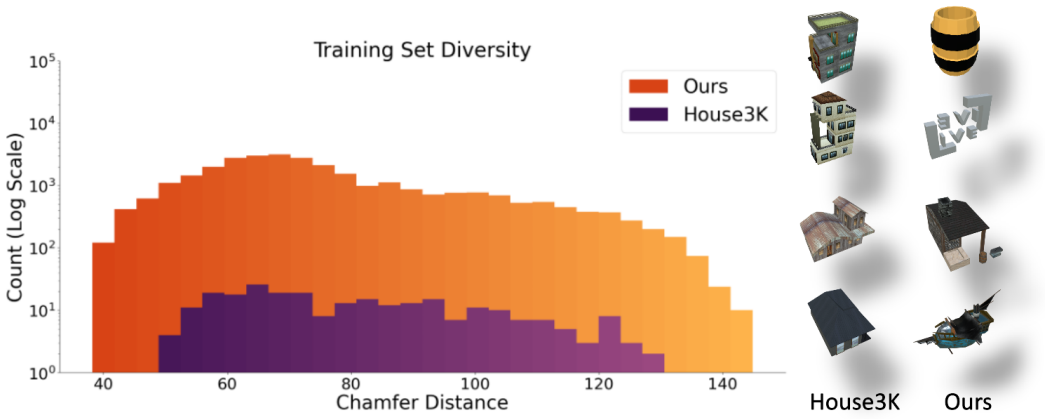

Figure S10. **More diverse and large-scale training set.** The chamfer distance measures the discrepancy between the point cloud of the training data and the sphere point cloud. The logarithmic scale of the count represents the number of shapes within each distance range. The right portion displays sample shapes with the same chamfer distance, shown side by side for each dataset. The wider chamfer distance range, higher number of shapes per chunk, and varied shape categories demonstrate that our training data, processed from Objaverse [4, 5], are more comprehensive and large-scale compared to Houses3K.

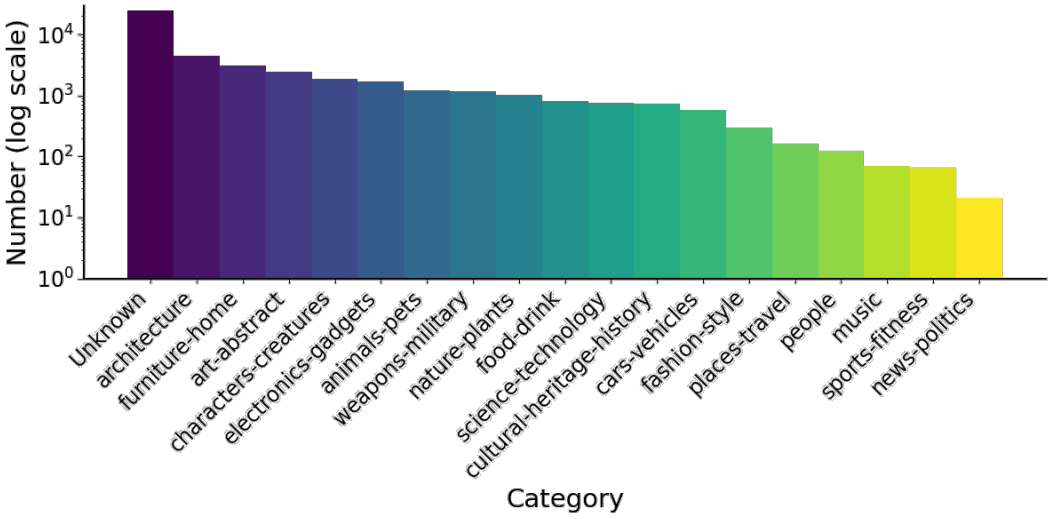

Figure S11. **Dataset distribution.** Our training dataset, processed from Objaverse [4, 5], includes a wide range of categories and is not limited to cubic-like shapes (e.g., buildings).

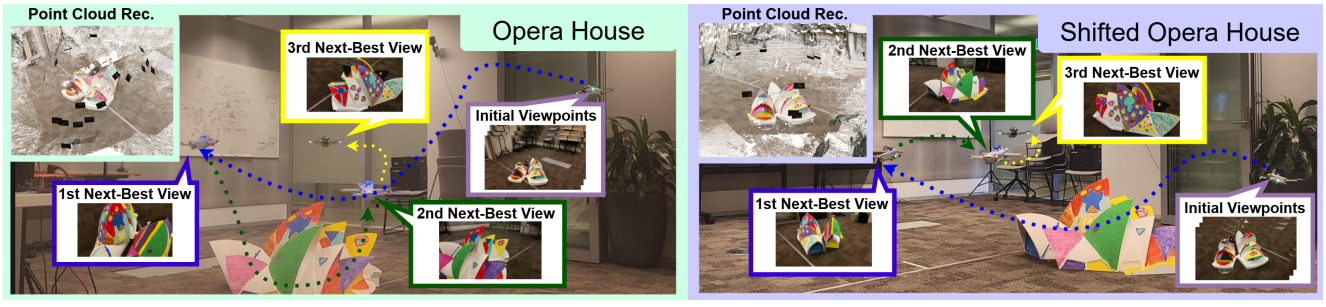

Figure S12. **Real-world drone system.** Hestia is a generalizable next-best-view planner that is feasible for real-world deployment. *Please refer to our demonstration video for further details.*

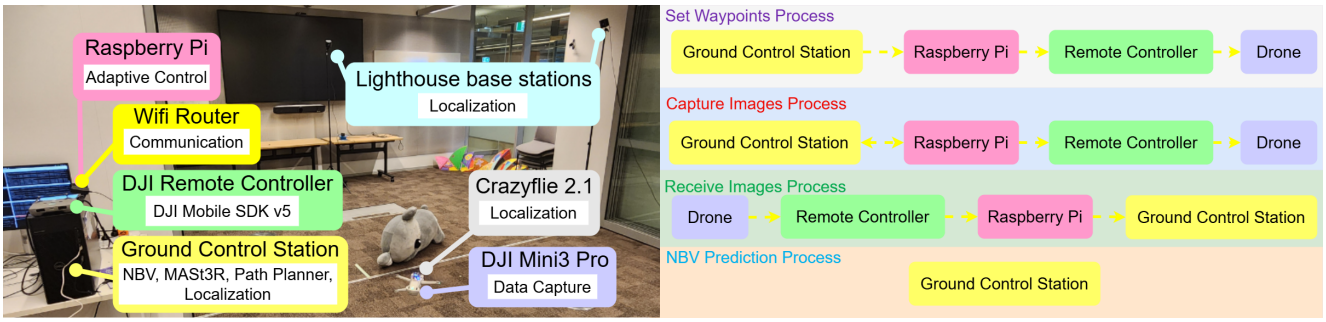

Figure S13. **Overview of the real-world drone system.** The system uses a drone with an RGB camera for data capture, Lighthouse base stations and Crazyflie for localization, and a Wifi router for wireless communication.

Algorithm 1 Main

Require: A drone system $\mathcal{D} = \{\mathcal{D}_{\text{gs}}, \mathcal{D}_{\text{pi}}, \mathcal{D}_{\text{ad}}, \mathcal{D}_{\text{ct}}, \mathcal{D}_{\text{drone}}\}$, **where:**

$\mathcal{D}_{\text{gs}}$: Ground station
$\mathcal{D}_{\text{pi}}$: Raspberry Pi
$\mathcal{D}_{\text{ad}}$: Android phone
$\mathcal{D}_{\text{ct}}$: Remote controller
$\mathcal{D}_{\text{drone}}$: Drone

$\mathcal{X} \leftarrow [x_1, x_2, x_3]$ ▷ Initial viewpoints
for $x \in \mathcal{X}$ **do**
  $\mathcal{W} \leftarrow [x]$ ▷ Update waypoints
  `Set_Waypoints(`$\mathcal{W}$`)` ▷ Move the drone (Alg. 2)
  $i \leftarrow$ `Capture_Image()` ▷ Capture an image (Alg. 3)
  `Receive_Image(i)` ▷ Transmit image (Alg. 4)
end for
for $k \in \{1, 2, \ldots, K\}$ **do**
  $\mathcal{W} \leftarrow$ `NBV_Prediction()` ▷ Predict NBV (Alg. 5)
  `Set_Waypoints(`$\mathcal{W}$`)` ▷ Move to NBV (Alg. 2)
  $i \leftarrow$ `Capture_Image()` ▷ Capture an image (Alg. 3)
  `Receive_Image(i)` ▷ Transmit image (Alg. 4)
end for

nication. MASt3R [6] is integrated to convert RGB images into pointmaps (e.g., depth images), and three initial viewpoints are set for real-world and virtual-world synchronization [21]. Fig. S13 and Alg. 1 illustrate four key processes of the system. Specifically, the drone captures images at three initial viewpoints, where the *set waypoints process* navigates the drone using a heuristic trajectory planner based on prior knowledge of the environment. Then, the *capture image process* commands image capture, and the *receive image process* transmits the image to the ground station. After capturing the initial viewpoints, the *nbv prediction process* predicts the next-best viewpoint based on the collected data. Four processes repeat until sufficient data is collected. For more details, please refer to Sec. S9.3.

S9.2. Real-World System Setup

The environment size of our object-centric scenes (e.g., an opera house) is approximately 2.6 m × 2.6 m × 2 m. To prevent the drone from exceeding the HTC Lighthouse base station range, the maximum height H_t is restricted to 1.5 m. Additionally, in the nearest collision-free voxel module, voxels below 0.4m are marked as occupied to avoid potential counterforces between the floor and the drone's quadrotor. In this system, we deploy the DJI Mini 3 Pro as the primary aircraft model.

We integrate the DJI Mobile SDK v5 to enable remote control and command transmission from the base station to the UAV. This software development kit provides developers with comprehensive control capabilities over the UAV. The SDK is embedded in an Android application

package, where we develop a custom application capable of broadcasting aircraft data via the User Datagram Protocol (UDP) wireless network protocol. The broadcast data is captured using a Raspberry Pi 4, which runs a ROS 2 node designed to receive UDP packets and convert them into ROS 2-compatible messages. On the same Raspberry Pi 4, we implement an adaptive trajectory planning technique. This method evaluates a pre-generated library of feasible offline trajectories, allowing the UAV to navigate autonomously by selecting the most appropriate path based on real-time conditions. Integrating these components ensures a reliable flow of data and commands, enabling efficient autonomous navigation for the UAV. We utilize the Crazyflie v2.1, a nano UAV equipped with a Lighthouse Positioning Deck, to achieve precise localization within the experimental environment. To integrate its capabilities with the primary aircraft, we remove the propellers and motors of the Crazyflie and securely mount it on top of the DJI Mini 3 Pro. This setup enables the Crazyflie to serve as a localization beacon, providing accurate positional data for the main UAV within the tracking range of the Lighthouse base stations. The positioning deck on the Crazyflie captures localization data using infrared signals from the Lighthouse system. This data is transmitted wirelessly via the Crazyradio 2.0 module, which connects to a base station. The base station, running the Crazyswarm 2.0 package on the ROS 2 framework, processes and publishes the localization data in real time. This setup facilitates autonomous navigation and precise positioning of the main UAV by continuously updating its coordinates within the experimental space.

The system is constructed entirely from publicly available, low-cost hardware. The total cost of the additional components, including the Crazyflie v2.1 (approximately $200), the Lighthouse Positioning Deck (around $100), and the Crazyradio 2.0 module (about $50), is approximately $350. When combined with the DJI Mini 3 Pro, which costs around $800, the total system cost remains significantly lower than that of conventional localization solutions. This cost-effective design, combined with open-source software such as ROS 2 and the Crazyswarm package, provides a reliable and accessible prototype for UAV-based data collection.

S9.3. Real-World System Processes

The flowchart of the system in the right part of Fig. S13, highlights four main sub-processes: *Set Waypoints Process*, *Capture Image Process*, *Receive Image Process*, and *NBV Prediction Process*. Each sub-process is represented by distinct colors in the diagram and described as follows:

- **Setting Waypoints (Alg. 2)**: Waypoints for the drone are pre-configured and stored on the ground station. These waypoints are transmitted to the drone through a communication channel comprising a Raspberry Pi, an Android mobile phone connected to the remote controller, and a remote controller. The drone navigates to each waypoint to align with the predicted NBV.

Algorithm 2 Set Waypoints

Require: Waypoints $\mathcal{W}$, and a drone system $\mathcal{D}$

$\mathcal{D}_{\text{gs}} \xrightarrow{\mathcal{W}} \mathcal{D}_{\text{pi}} \xrightarrow{\mathcal{W}} \mathcal{D}_{\text{ad}} \xrightarrow{\mathcal{W}} \mathcal{D}_{\text{ct}} \xrightarrow{\mathcal{W}} \mathcal{D}_{\text{drone}}$ ▷ Send waypoints to the drone
for $w \in \mathcal{W}$ **do**
    $\mathcal{D}_{\text{drone}}$.`move_to`$(w)$ ▷ Move to waypoint
end for

- **Capturing Images (Alg. 3)**: Upon reaching a waypoint, the Raspberry Pi retrieves the drone's real-time position from the ground station. Once the waypoint is confirmed, the ground station sends a "capture image" command. The drone then captures the image using adjusted camera parameters.

Algorithm 3 Capture Image

Require: A drone system $\mathcal{D}$

while $\mathcal{D}_{\text{drone}}$.loc $\not\approx$ $\mathcal{D}_{\text{pi}}$.x **do** ▷ Wait until location matches NBV
    Continue
end while
$\mathcal{D}_{\text{pi}} \xrightarrow{\text{NBV\_reached}} \mathcal{D}_{\text{gs}}$ ▷ Notify ground station
$i \leftarrow \mathcal{D}_{\text{drone}}$.`capture()` ▷ Capture image
return i ▷ Return image

- **Receiving Images (Alg. 4)**: After capturing an image, the drone transmits the image along with its real-time pose back to the ground station. These images are used to iteratively update the reconstruction model.

Algorithm 4 Receive Image

Require: Image i and a drone system $\mathcal{D}$

$\mathcal{D}_{\text{drone}} \xrightarrow{i} \mathcal{D}_{\text{ct}} \xrightarrow{i} \mathcal{D}_{\text{ad}} \xrightarrow{i} \mathcal{D}_{\text{pi}} \xrightarrow{i} \mathcal{D}_{\text{gs}}$ ▷ Transmit image to ground station
$\mathcal{D}_{\text{gs}}$.`save`(i) ▷ Save image
$\mathcal{D}_{\text{gs}}$.`save`(x') ▷ Save real-time position

- **Predicting the NBV (Alg. 5)**: The ground station processes the captured image and the drone's pose to predict the next-best-view using the NBV module. The newly determined viewpoint is then sent to the drone to continue the data collection process.

Algorithm 5 NBV Prediction

Require: A drone system $\mathcal{D}$

1: $\mathcal{I} \leftarrow [i_1, \ldots, i_n]$ ▷ Load images
2: $\mathcal{X}^{'} \leftarrow [x_1^{'}, \ldots, x_n^{'}]$ ▷ Load positions
3: $\mathcal{G} \leftarrow \mathcal{D}_{\text{gs}}.\texttt{MASt3R}(\mathcal{X}', \mathcal{I})$ ▷ Compute grid
4: $x \leftarrow \mathcal{D}_{\text{gs}}.\texttt{pred\_NBV}(\mathcal{G}, i_n, x_n^{'}, h)$ ▷ Predict NBV
5: $\mathcal{W} \leftarrow \mathcal{D}_{\text{gs}}.\texttt{generate\_waypoints}(x)$ ▷ Generate waypoints
6: **return** $\mathcal{W}$ ▷ Return waypoints

Alg. 1 provides an overview of the entire process. Initially, the drone visits three pre-defined viewpoints to synchronize the real-world and virtual-world data (lines 1-7). Following these initial captures, the NBV module predicts subsequent viewpoints based on the collected data (lines 8-13), guiding the drone iteratively until sufficient data is acquired for reconstruction. Additionally, the MASt3R module is integrated into the ground station to convert RGB images into pointmaps (e.g., depth images). By combining these components, the system enables efficient and intelligent data collection, demonstrating the potential of drones as autonomous agents for scalable and versatile real-world scenarios.

S10. Training and Testing Details

This section includes the details of the training and testing. We employ PPO [18] from stable-baselines3 [16] as the reinforcement learning framework for Hestia. The grid resolution g is set to 20, and h and w are set to 300. The initial learning rate is 3×10^{-4} and is decayed by a factor of 2 every 500,000 iterations starting from 2,000,000 iterations until reaching 4,000,000, for a total of 5,000,000 training iterations. During training, H_t is randomly sampled from the initial viewpoint height, up to a maximum of 10 meters. For testing, H_t starts at 10 meters, is reduced to 5 meters during the last 10 to 5 steps, and further decreases to 2 meters in the final 5 steps. A training episode ends and the scene resets either when the number of captured images reaches 50 or when the target face coverage ratio of 0.9 is achieved. The complete training process takes approximately 24 hours on an NVIDIA RTX A6000 GPU.

Our network architecture is lightweight, with only 4.9 million parameters (see Tab. 3), which is approximately half the size of a standard ResNet-18 model, which has around 11.7 million parameters. The proposal network consists of three 3D convolutional layers, each followed by a Leaky ReLU activation. This design progressively downsamples the 3D grid before further downstream operations. It is then followed by a 3D self-attention layer, again paired with a Leaky ReLU activation, to expand the receptive field. Finally, the network applies a reparameterization trick module, composed of linear layers, to generate the output distribution parameters. This design allows the look-at point to be sampled from a distribution rather than predicted deterministically. The grid encoder consists of three 3D convolutional layers, each followed by batch normalization and a Leaky ReLU activation. After encoding, trilinear interpolation is applied to each encoded grid feature, followed by feature concatenation. The image encoder is composed of three 2D convolutional layers, each followed by batch normalization and Leaky ReLU activation. The encoded features are then flattened and passed through a linear layer with Leaky ReLU activation. For the policy network, we adopt the default model provided in stable-baselines3. For more details about the network architecture, please refer to the code provided in the supplementary materials. The hierarchical design first predicts the look-at point, followed by the camera position. This design prioritizes the look-at point, as the primary objective in this task is to determine where to look rather than where to fly. It also resembles how a human pilot controls a drone during data capture, focusing first on the target of observation before planning the flight path.

S11. Limitations

Although the quantitative and qualitative results (see Sec. 4.3 and Sec. S3) demonstrate a nearly comprehensive point cloud reconstruction, there are still some failure cases of Hestia (see Fig. S14). Hestia may occasionally fail to capture finer 3D structures, such as the window frames of the first-row house shown in Fig. S14. It may also fail to reconstruct highly self-occluded parts, such as the pillar of the second-row house. In addition, Hestia sometimes struggles to capture bottom-up views that require extreme vertical viewing angles, for example, the Lego man's right hand and the underside of the pillar. Moreover, it may struggle to reconstruct shapes with fine details over coarse surfaces, such as tiny parts of the broccoli and anise. Adopting a multi-resolution grid structure or integrating octree-based methods to enhance the voxel grid resolution could be a potential future step to mitigate these issues.

In addition to the above limitations, we hope that Hestia will not be misused for other types of next-best-view (NBV) tasks. In this study, we found that a close-greedy training scheme can effectively mitigate spurious correlations and is well-suited to our problem definition (see Sec. S7, Sec. 3, and Sec. 4). However, the next-best-view problem is a broad research topic with varying objectives. This finding may not generalize to other NBV tasks, such as next-best-view for object tracking or next-best-view for human aesthetics, where long-term planning is more critical.

Due to hardware limitations (e.g., the absence of an RGB-D camera), Hestia cannot fully exhibit its potential in the real-world drone system. However, since we use a depth estimator to convert RGB images into depth maps, this lim-

Figure S14. **Failure cases of Hestia.** Hestia may occasionally fail to capture finer 3D structures, highly self-occluded parts, nearly vertical bottom-up views, and small details on coarse object surfaces.

itation represents a trade-off rather than a fundamental constraint. Our experiments conducted in NVIDIA IsaacLab demonstrate the full capability of Hestia, while the real-world application highlights Hestia's robustness when a depth sensor is unavailable. Furthermore, due to drone regulations, the real-world application of Hestia is conducted indoors using an indoor GPS system (e.g., HTC Lighthouse base stations). Drone policies vary across countries, and obtaining outdoor flight approvals can take up to a year in our region. Additionally, outdoor trials require significant funding, such as renting a safe test site measuring approximately 100 meters by 100 meters. As a future step, we plan to test Hestia outdoors to further validate its performance. Another future step is to extend Hestia to a multi-agent setting for large-scale outdoor scanning (e.g., city-scale) under power-constrained scenarios.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 8

[2] Arnon Boneh and Micha Hofri. The coupon-collector problem revisited—a survey of engineering problems and computational methods. *Stochastic Models*, 13(1):39–66, 1997. 1

[3] Xiao Chen, Quanyi Li, Tai Wang, Tianfan Xue, and Jiangmiao Pang. Gennbv: Generalizable next-best-view policy for active 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16436–16445, 2024. 2, 3, 4, 7, 8

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023. 1, 5, 7, 9, 10

[5] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 5, 7, 10

[6] Bardienus Pieter Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. *CoRR*, 2024. 3, 11

[7] Antoine Guédon, Tom Monnier, Pascal Monasse, and Vincent Lepetit. Macarons: Mapping and coverage anticipa-

tion with rgb online self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 940–951, 2023. 2, 3, 7

[8] A CAUSAL INTERPRETATION. Spurious correlation: A causal interpretation herbert a. simon. *Causal Models in the Social Sciences*, page 5, 1971. 4

[9] Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popović. Neu-nbv: Next best view planning using uncertainty estimation in image-based neural rendering. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11305–11312. IEEE, 2023. 2, 3, 7, 8

[10] Younghyun Kim, Sangwoo Mo, Minkyu Kim, Kyungmin Lee, Jaeho Lee, and Jinwoo Shin. Discovering and mitigating visual biases through keyword explanation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11082–11092, 2024. 4

[11] Soomin Lee, Le Chen, Jiahao Wang, Alexander Liniger, Suryansh Kumar, and Fisher Yu. Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields. *IEEE Robotics and Automation Letters*, 7(4):12070–12077, 2022. 3, 7

[12] Kevin Lin and Brent Yi. Active view planning for radiance fields. In *Robotics Science and Systems*, 2022. 3, 7

[13] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. Oakland, CA, USA, 1967. 8

[14] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. In *European Conference on Computer Vision*, pages 230–246. Springer, 2022. 3, 7

[15] 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. 1, 2, 3, 5, 6, 7, 8, 9

[16] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. 13

[17] Yunlong Ran, Jing Zeng, Shibo He, Jiming Chen, Lincheng Li, Yingfeng Chen, Gimhee Lee, and Qi Ye. Neurar: Neural uncertainty for autonomous 3d reconstruction with implicit neural representations. *IEEE Robotics and Automation Letters*, 8(2):1125–1132, 2023. 3, 7

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 13

[19] Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9370–9376. IEEE, 2023. 3, 7

[20] Yuezhan Tao, Yuwei Wu, Beiming Li, Fernando Cladera, Alex Zhou, Dinesh Thakur, and Vijay Kumar. Seer: Safe efficient exploration for aerial robots using learning to predict information gain. *arXiv preprint arXiv:2209.11034*, 2022. 7

[21] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04): 376–380, 1991. 11

[22] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 3

[23] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 803–814, 2023. 1, 4, 5, 7, 8, 9

[24] Jing Zeng, Yanxu Li, Yunlong Ran, Shuo Li, Fei Gao, Lincheng Li, Shibo He, Qi Ye, et al. Efficient view path planning for autonomous implicit reconstruction. *arXiv preprint arXiv:2209.13159*, 2022. 7

[25] Huangying Zhan, Jiyang Zheng, Yi Xu, Ian Reid, and Hamid Rezatofighi. Activermap: Radiance field for active mapping and planning. *arXiv preprint arXiv:2211.12656*, 2022. 3, 7

[26] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 8