

SUPPLEMENTARY MATERIAL FOR "NEXTBESTPATH: EFFICIENT 3D MAPPING OF UNSEEN ENVIRONMENTS"

Anonymous authors

Paper under double-blind review

1 DATASET

Dataset construction. To ensure that each map offers full accessibility for various robotic platforms such as unmanned aerial vehicles (UAVs) and wheeled robots, we configure all doors and windows to remain open during map generation. However, we observe that Obsidian does not consistently guarantee accessibility to all areas. To resolve this, we manually edited each scene to ensure the traversability of windows, doors, and hidden passages.

Dataset statistic. To calculate navigation complexity, we sampled location pairs from four environments ranging in difficulty from simple to insane, at rates of 1%, 0.1%, 0.05%, and 0.02%, constrained by computational resource limitations. Consequently, our data effectively represents a lower bound of navigation complexity. Despite this conservative sampling approach, our dataset remains the most complex one, offering a significant challenge for future research.

Figure 1 presents more examples of maps from our dataset, showcasing various levels and diverse scenarios.

2 DETAILS OF MAPPING PROCESS ENCODER

We provide more details of the Mapping Process Encoder of our proposed approach in this section.

The mapping encoding is predicted from both the current reconstruction progress and historical trajectory data. At each time step $t \geq 0$, we construct and refine a surface point cloud \mathcal{P}_t by integrating information from newly captured depth map $D_t : \Omega \rightarrow \mathbb{R}^+$ and merging it with our existing reconstructed point cloud. For each camera pose $c_t = (c_t^{\text{pos}}, c_t^{\text{rot}})$, we transform the corresponding depth map D_t into a set of 3D points. This transformation makes use of the camera’s intrinsic matrix $K \in \mathbb{R}^{3 \times 3}$ and the pose matrix $T_t \in SE(3)$, derived from the 6D pose c_t :

$$\mathbf{p}_{\text{surface}}(u, v) = T_t \cdot \left(D_t(u, v) \cdot K^{-1} \cdot [u \ v \ 1]^\top \right), \quad (u, v) \in \Omega, \quad (1)$$

where $\Omega \subset \mathbb{R}^2$ represents the domain of the depth map. We accumulate points over time:

$$\mathcal{P}_t = \mathcal{P}_{t-1} \cup \{ \mathbf{p}_{\text{surface}}(u, v) \mid (u, v) \in \Omega, D_t(u, v) > 0 \}. \quad (2)$$

To enhance scalability and generalization, we introduce a slice mapping approach that transforms the point cloud into a set of K images. We begin by filtering the point cloud based on the camera’s position:

$$\mathcal{P}_{c_t}^f = \{ \mathbf{p} = (p_x, p_y, p_z) \in \mathcal{P}_t \mid |p_x - x_{c_t}| \leq r \text{ and } |p_z - z_{c_t}| \leq r \}, \quad (3)$$

where r is the radius of our observation window and $(x_{c_t}, y_{c_t}, z_{c_t})$ is the current camera position. We then divide $\mathcal{P}_{c_t}^f$ into n equal vertical slices along the Y-axis, y_{\min} and y_{\max} come from a defined exploration bounding box, as Guédon et al. (2023) did:

$$\mathcal{S}_{c_t, j} = \{ \mathbf{p} = (p_x, p_y, p_z) \in \mathcal{P}_{c_t}^f \mid y_{\min} + (j-1)h_{\text{slice}} \leq p_y < y_{\min} + jh_{\text{slice}} \}, \quad (4)$$

where $h_{\text{slice}} = (y_{\max} - y_{\min})/n$ and $j \in 1, \dots, n$. Each slice $\mathcal{S}_{c_t, j}$ is mapped to an image $I_{c_t, j}$ of size $H \times W$ using a projection function $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\phi(\mathbf{p}) = \left(\left\lfloor \frac{(p_x - x_{c_t} + r) \cdot W}{2r} \right\rfloor, \left\lfloor \frac{(p_z - z_{c_t} + r) \cdot H}{2r} \right\rfloor \right). \quad (5)$$

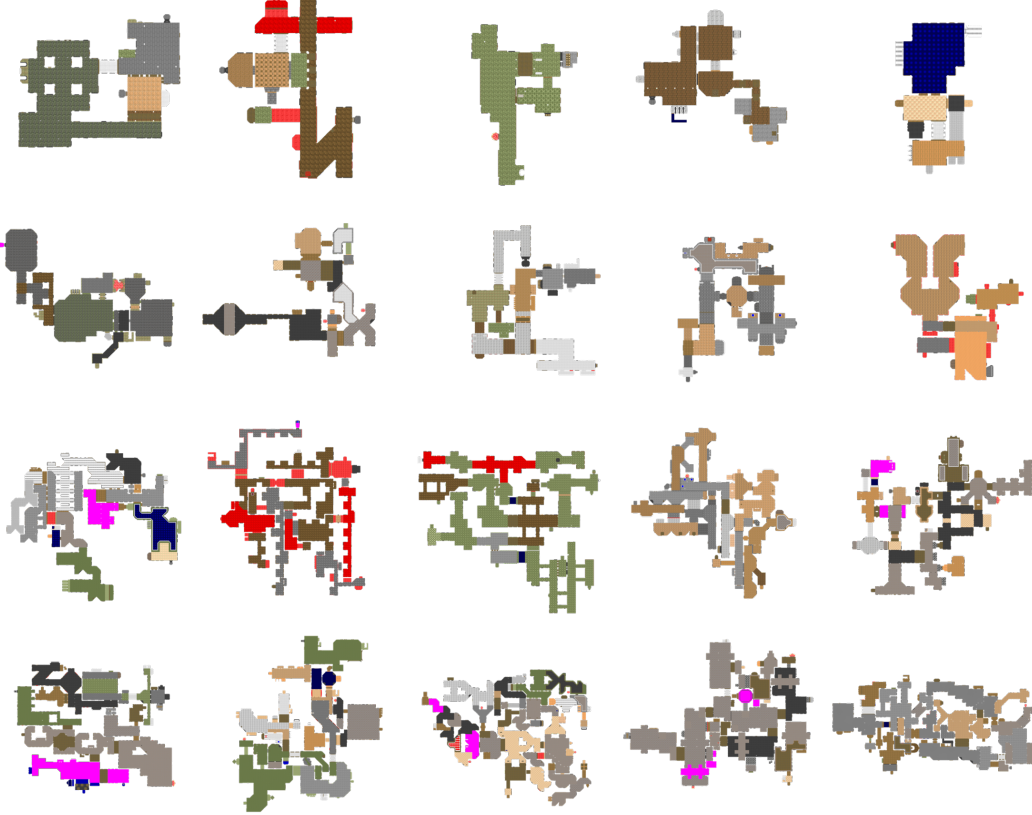


Figure 1: **More maps from our dataset.** Rows from top to bottom represent increasing scene complexity, categorized into four levels: Simple, Normal, Hard, Insane.

This projection centres the camera in the middle of the image. Finally, we calculate the point density for each pixel (u, v) in the image $I_{c_t, j}$:

$$I_{c_t, j}(u, v) = \int_{\mathcal{S}_{c_t, j}} \delta(\phi(\mathbf{p}) - (u, v)) d\mathbf{p}. \quad (6)$$

Here, $\delta(\cdot)$ is the Dirac delta function and \mathbf{p} represents points from the slice $\mathcal{S}_{c_t, j}$. This process yields n density images $I_{c_t, 1}, \dots, I_{c_t, n}$ for each time step t , effectively transforming 3D point cloud data into 2D representations.

In addition, we apply a similar approach to project the camera’s historical trajectory, resulting in a single 2D image. We filter the camera’s historical positions based on their proximity to the current camera position in the XZ-plane, using the same threshold τ_{xz} :

$$\mathcal{C}_t^f = \{c_k^{pos} = (x_k, y_k, z_k) \mid k < t, |x_k - x_t| \leq \tau_{xz} \text{ and } |z_k - z_t| \leq \tau_{xz}\}. \quad (7)$$

We then map these filtered positions onto a single image H_{c_t} of the same size $H \times W$:

$$H_{c_t}(u, v) = \sum_{c_k^{pos} \in \mathcal{C}_t^f} \delta(\phi(c_k^{pos}) - (u, v)) \quad (8)$$

This results in a single-density image H_{c_t} representing the camera’s historical trajectory near its current position. To synthesize the information obtained, we define a set \mathcal{E}_{c_t} encapsulating the entirety of the current exploration embedding: $\mathcal{E}_{c_t} = \{I_{c_t, 1}, \dots, I_{c_t, n}, H_{c_t}\}$.

Scene	Rooms	Comp. (%) \uparrow						Comp. (cm) \downarrow					
		Random	FBE	UPEN	OccAnt	ANM	NBP (ours)	Random	FBE	UPEN	OccAnt	ANM	NBP (ours)
GdvgF*	6	68.45	81.78	82.39	80.24	80.99	87.80	11.67	5.48	5.14	5.66	5.69	4.92
gZ6f7	1	29.81	81.01	82.96	82.02	80.68	89.91	46.48	7.06	6.14	6.19	7.43	3.31
HxpKQ*	8	46.93	58.71	52.70	60.50	48.34	66.28	19.10	11.75	14.11	11.75	15.96	8.12
pLe4w	2	32.92	66.09	66.76	67.13	76.41	71.34	30.79	12.78	11.82	11.51	8.03	9.53
YmJkq	4	50.26	68.32	60.47	68.70	79.35	81.57	24.61	11.85	15.77	11.90	8.46	8.01
mean	4	45.67	71.18	69.06	71.72	73.15	79.38	26.53	9.78	10.60	9.40	9.11	6.78

Table 1: Evaluation results for each test scene on MP3D dataset.

	AiMDoom Simple				AiMDoom Normal			
	Seen		Unseen		Seen		Unseen	
	Final Cov.	AUC	Final Cov.	AUC	Final Cov.	AUC	Final Cov.	AUC
Random Walk	0.362	0.306	0.323	0.270	0.198	0.159	0.190	0.152
	± 0.175	± 0.156	± 0.156	± 0.135	± 0.125	± 0.104	± 0.124	± 0.103
FBE	0.770	0.628	0.760	0.605	0.564	0.423	0.565	0.415
	± 0.163	± 0.147	± 0.174	± 0.171	± 0.171	± 0.127	± 0.139	± 0.109
SCONE	0.597	0.482	0.577	0.483	0.421	0.315	0.412	0.313
	± 0.177	± 0.158	± 0.173	± 0.138	± 0.138	± 0.102	± 0.114	± 0.087
MACARONS	0.600	0.483	0.599	0.479	0.442	0.332	0.418	0.314
	± 0.176	± 0.145	± 0.200	± 0.172	± 0.135	± 0.104	± 0.120	± 0.088
NBP (Ours)	0.870	0.697	0.879	0.692	0.746	0.538	0.734	0.526
	± 0.121	± 0.134	± 0.142	± 0.156	± 0.152	± 0.142	± 0.142	± 0.112

Table 2: Evaluation results on AiMDoom dataset (Simple and Normal).

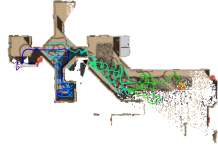
3 EXPERIMENTS

Detailed quantitative results. Table 2 and Table 3 show our superior performance on both the AiMDoom training set and the test set. Furthermore, we offer detailed results for each test scene in MP3D, as illustrated in Table 1.

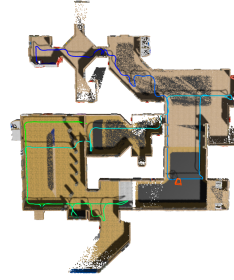
Qualitative results. We provide additional visual comparisons between our method and the state-of-the-art NBV-based method: MACARONS (Guédon et al., 2023), from Figure. 2, Figure. 3 and Figure. 4. These comparisons demonstrate that our trajectories consistently exhibit superior performance, whether in simple or complex scenarios. Both methods start from the same pose, indicated by a deep blue colour in the visualizations of trajectories.

	AiMDoom Hard				AiMDoom Insane			
	Seen		Unseen		Seen		Unseen	
	Final Cov.	AUC	Final Cov.	AUC	Final Cov.	AUC	Final Cov.	AUC
Random Walk	0.121	0.086	0.124	0.088	0.070	0.048	0.074	0.050
	± 0.081	± 0.062	± 0.082	± 0.060	± 0.049	± 0.038	± 0.048	± 0.035
FBE	0.426	0.310	0.425	0.311	0.313	0.226	0.330	0.239
	± 0.119	± 0.091	± 0.114	± 0.080	± 0.082	± 0.066	± 0.097	± 0.079
SCONE	0.271	0.199	0.290	0.210	0.204	0.146	0.196	0.140
	± 0.100	± 0.172	± 0.093	± 0.072	± 0.069	± 0.052	± 0.079	± 0.060
MACARONS	0.316	0.202	0.302	0.218	0.201	0.143	0.192	0.139
	± 0.106	± 0.074	± 0.097	± 0.070	± 0.068	± 0.051	± 0.078	± 0.058
NBP (Ours)	0.627	0.430	0.618	0.432	0.486	0.315	0.472	0.312
	± 0.144	± 0.111	± 0.153	± 0.115	± 0.106	± 0.047	± 0.095	± 0.073

Table 3: Evaluation results on AiMDoom dataset (Hard and Insane).



(a) MACARONS

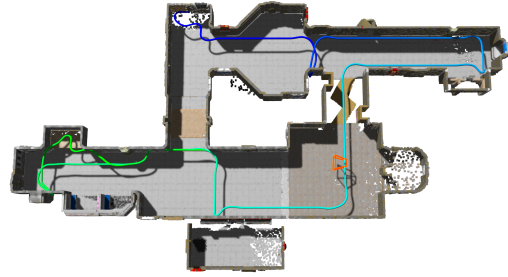


(b) NBP (Ours)

Figure 2: **Comparison 1:** In complex and narrow spaces, the NBV (next-best-view) based method can easily get trapped in a local area. Although our method did not manage to reconstruct all areas in this complex scene, it covered most of the areas.

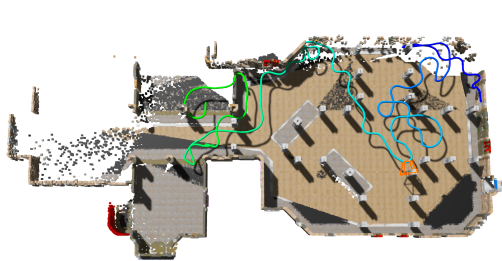


(a) MACARONS

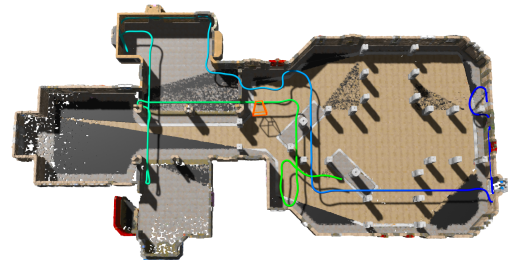


(b) NBP (Ours)

Figure 3: **Comparison 2:** The NBV-based method can easily "assume" that an area has been fully explored, as it focuses solely on local optimal solutions, similar to this sample. In complex indoor environments, it is often necessary to skip some locally optimal poses.



(a) MACARONS



(b) NBP (Ours)

Figure 4: **Comparison 3:** In relatively simple scenes with some obstacles, the NBV exploration can also become trapped in one area.

As Figure. 5 and Figure. 6 illustrated, we also show that in very complex environments, we could only achieve about 65% coverage. This is because, in complex environments, our method prioritizes the exploration of areas with multiple valuable goals, ignoring places of lesser current value. After the initial exploration is complete, it is likely to explore other regions, overlooking previously encountered areas with higher value. Consequently, developing methods that aim to achieve a global optimum is a promising and valuable direction for future research.



Figure 5: **Failure case 1:** Our method initially prioritizes the exploration of high-value areas, inadvertently neglecting regions of secondary importance. Thus, it results in incomplete reconstruction in the initial area of the beginning trajectory.

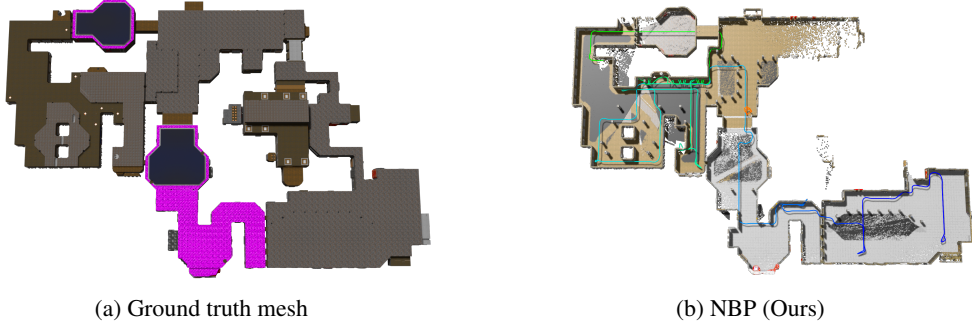


Figure 6: **Failure case 2:** This scene contains multiple narrow areas, prompting our method to depend more heavily on our precise prediction of obstacles. Under these challenging conditions, our approach may overlook exploring this area.

REFERENCES

Antoine Guédon, Tom Monnier, Pascal Monasse, and Vincent Lepetit. Macarons: Mapping and Coverage Anticipation with RGB Online Self-Supervision. In *Conference on Computer Vision and Pattern Recognition*, 2023.