2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

AeroScene: Progressive Scene Synthesis for Aerial Robotics

Nghia Vu[†,2], Tuong Do[†,1,2,3], Dzung Tran[4], Binh X. Nguyen[2], Hoan Nguyen[5], Erman Tjiputra[2], Quang D. Tran[1,2], Hai-Nguyen Nguyen[4], Anh Nguyen[1]

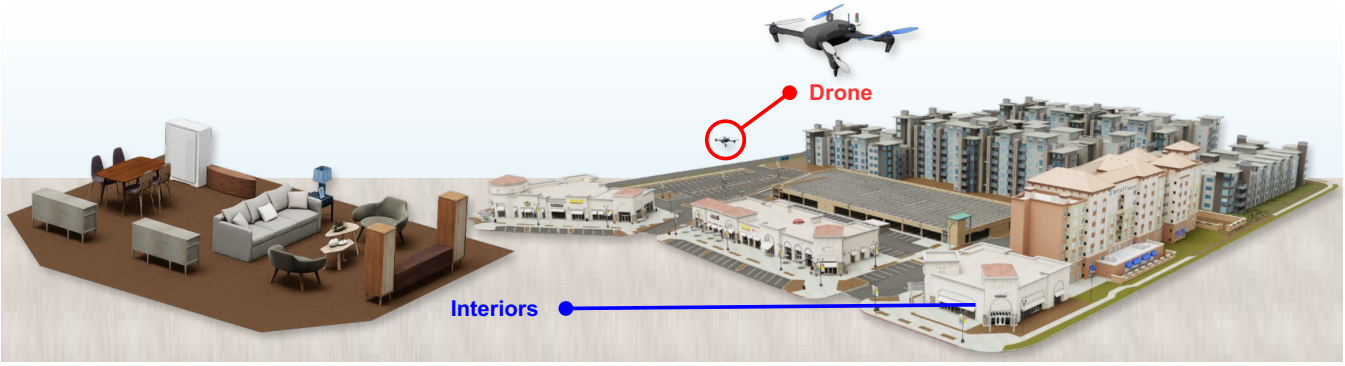


Fig. 1: We introduce AeroScene, a progressive scene synthesis method and dataset for aerial robotics.

***Abstract*— Generative models have shown substantial impact across multiple domains, their potential for scene synthesis remains underexplored in robotics. This gap is more evident in drone simulators, where simulation environments still rely heavily on manual efforts, which are time-consuming to create and difficult to scale. In this work, we introduce AeroScene, a hierarchical diffusion model for progressive 3D scene synthesis. Our approach leverages hierarchy-aware tokenization and multi-branch feature extraction to reason across both global layouts and local details, ensuring physical plausibility and semantic consistency. This makes AeroScene particularly suited for generating realistic scenes for aerial robotics tasks such as navigation, landing, and perching. We demonstrate its effectiveness through extensive experiments on our newly collected dataset and a public benchmark, showing that AeroScene significantly outperforms prior methods. Furthermore, we use AeroScene to generate a large-scale dataset of over 1,000 physics-ready, high fidelity 3D scenes that can be directly integrated into NVIDIA Isaac Sim. Finally, we illustrate the utility of these generated environments on downstream drone navigation tasks. Our code and dataset are publicly available at aioz-ai.github.io/AeroScene/**

I. Introduction

Drones are increasingly applied in delivery, inspection, and surveillance, requiring them to navigate and operate within complex 3D environments [1], [2]. Synthesizing realistic scenes for such applications necessitates hierarchical layout generation, where coarse-scale structures (e.g., rooms, terrains, building layouts) establish navigable flight corridors, while fine-scale details (e.g., obstacles, landing areas) ensure task-specific feasibility. However, existing scene creation methods for drone simulators are designed primarily by humans, making them challenging to scale, and often fail to accommodate both physical and fidelity requirements such as unobstructed aerial navigation [3], accessible interaction areas (e.g., landing pads, inspection points) [4], and coherent indoor-outdoor transitions [5].

While several drone simulators have been developed to support research in navigation and control [6], [7], most remain limited in providing diverse and realistic environments for evaluating aerial tasks. Many simulators focus on accurate physics and sensor fidelity, yet often rely on static, handcrafted environments, hindering scalability, diversity, and the realism necessary for advanced testing [8]. Moreover, interaction areas critical to aerial robotics, such as landing zones, cluttered corridors, and inspection surfaces, are frequently simplified or entirely absent, limiting full task realism [9]. As a result, existing platforms offer limited support for benchmarking higher-level autonomy, where navigation, task execution, and environment understanding must be jointly evaluated in realistic, hierarchical settings [10].

In this paper, we propose **AeroScene**, a hierarchical diffusion-based framework for 3D scene generation tailored to drone tasks. Our approach operates across scales: coarse-scale synthesis generates high-level structures that preserve airspace and navigability, while fine-scale synthesis refines object placement and specifies drone-interaction areas for task execution. AeroScene includes Cross-scale Progressive Attention, which explicitly models dependencies across scales, ensuring that fine-scale details remain consistent with coarse-scale spatial structures. In addition, we design task-aware guidance functions that encourage collision-free plausibility, maintain semantic correlations in hierarchical orders, and handle relationships between indoor and outdoor objects, thereby aligning generated layouts with real-world aerial operation requirements. To support downstream tasks, scenes created by our method are directly embedded into NVIDIA Isaac Sim [11] for physics-ready simulation.

† Equal contribution
[1] University of Liverpool, UK
[2] AIOZ Ltd., Singapore
[3] National Tsing Hua University, Taiwan
[4] RMIT University, Vietnam Campus
[5] University of Information Technology, VNUHCM, Vietnam.

Our main contributions are as follows:

- We introduce a new framework that generates realistic and high-fidelity scenes for aerial robotics.
- We contribute a large-scale dataset with more than 1000 scenes and embed them into Isaac Sim to serve as a benchmark for drone-related tasks.

II. Related Works

Drone Simulators. Numerous simulators have been developed to support aerial robotics research, with varying emphasis on physics fidelity, sensor modeling, and environmental complexity [6], [7]. While these platforms provide valuable testbeds for navigation and perception, most rely on static or handcrafted environments, limiting their ability to represent diverse and scalable 3D scenes. NVIDIA Isaac Sim [11] offers a modern foundation with high-quality rendering, physics, and integration with learning frameworks, making it well-suited as a base platform. However, existing simulators focus primarily on physics and sensing rather than adaptive scene synthesis. To highlight these differences, Table I summarizes key features of widely used drone simulators. Our work builds AeroScene to generate large-scale, physics-ready high fidelity scenes for drone-related tasks that can be embedded directly into Issac Sim.

Simulator	Physics	Diversity	Scalability	Indoor	Outdoor	Multi-scale
RotorS [6]	✓	Low	Limited	✓	✗	✗
AirSim [7]	✓	Medium	Limited	✓	✓	✗
OmniDrones [12]	✓	Medium	High	✓	Partial	✗
QuadSwarm [13]	✓	Low	High	✓	✗	✗
VisFly [14]	✓	High	Medium	Partial	✓	✗
IAP [15]	✓	High	Medium	✓	Partial	✗
AeroScene (Ours)	✓	**High**	**High**	✓	✓	✓

TABLE I: Comparison of drone simulators.

Scene Synthesis. Scene synthesis aims to generate structured 3D layouts of objects in indoor and outdoor environments. Early approaches relied on rule-based or probabilistic grammars [16] and heuristic priors [17], but lacked scalability. Deep generative models improved plausibility, with GAN- and autoregressive frameworks producing realistic layouts [18]–[21], though they often struggle to balance global structure and local detail. Diffusion-based methods offer greater stability and diversity [22]–[25], but typically treat layouts as flat sets, limiting cross-scale reasoning. We tackle this problem with a hierarchical-scale modeling approach, which routes scene elements into coarse and fine branches, fusing them through alternating cross-scale attention. This explicitly propagates global layout context while refining local details, a design particularly suited to drone tasks, where both macro-structure (e.g., roads, buildings) and fine geometry (e.g., vehicles, furniture) are critical for perception, planning, and simulation fidelity [26]–[28].

3D Layout Representations. 3D scenes have been represented using voxel grids [29]–[31], meshes [23], [32], [33], point clouds [34], [35], and object-centric layouts [23], [36]–[39]. While voxels and meshes capture high-resolution geometry, they are computationally costly and less interpretable for task-level reasoning. Object-centric layouts abstract scenes into discrete entities with positions, orientations, scales, and semantic labels, providing a compact and interpretable structure suited for simulation and planning [40], [41]. Our method builds on this line and routes objects into coarse-to-fine representations for drone-related tasks.

Guided Diffusion Models. Guided diffusion has become a popular generative model for task-specific objectives. Classifier guidance [42], [43], classifier-free guidance [44], and score distillation techniques [45] have enabled control over semantics or conditions. Training-free approaches such as energy-based guidance [46], constraint-driven sampling [47], and physical priors [48] have extended diffusion models to respect external objectives. Prior works in 3D synthesis often adapt generic guidance strategies, such as collision penalties or semantic constraints [49]–[52], but typically treat them as auxiliary heuristics rather than deeply integrated objectives. In contrast, we directly incorporate task-specific objectives into the hierarchical scene synthesis process.

III. Methodology

A. Diffusion Process

We adopt a denoising diffusion probabilistic model (DDPM) [53] to learn the distribution of plausible scene layouts. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ for a fixed noise schedule $\{\beta_t\}_{t=1}^{T}$. The forward process gradually adds Gaussian noise to a clean layout x_0:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t I), \tag{1}$$

The reverse process removes noise step-by-step:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{2}$$

where μ_θ is predicted by a hierarchy-aware pipeline every denoising timestep. Concretely, at each timestep t, we apply the Initial Layout and Hierarchy Embedding (Sec. III-B) to convert x_t into hierarchy tokens, then extract coarse and fine features (Sec. III-C), fuse them with the Cross-scale Progressive Attention (Sec. III-D), and finally condition the denoising UNet on the fused features. Guidance objectives (Sec. III-E) are applied to adjust μ_θ before sampling x_{t-1}. This design ensures the denoiser performs hierarchical reasoning at each diffusion step, matching the iterative sampling loop. Fig. 2 shows the details of our method.

B. Initial Layout and Hierarchy Embedding

At each diffusion step, the noisy layout x_t is represented as

$$x_t = \{o_i \mid o_i = (\mathbf{p}_i, \mathbf{q}_i, \mathbf{s}_i, c_i)\}_{i=1}^{N_o}, \tag{3}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ is the position, $\mathbf{q}_i \in \mathbb{R}^4$ is the orientation quaternion, $\mathbf{s}_i \in \mathbb{R}^3$ is the scale, $c_i \in \{1, \ldots, C\}$ is the semantic category label represented as a C-dimensional one-hot vector which conditioned as learned soft-embeddings, and N_o is the number of objects in x_t.

Each element is mapped to a d_m-dimensional token:

$$\mathbf{h}_i = \mathbf{f}_i^{(0)} + \mathbf{e}_i^{\text{pos}} + \mathbf{e}_i^{\text{dom}}, \tag{4}$$

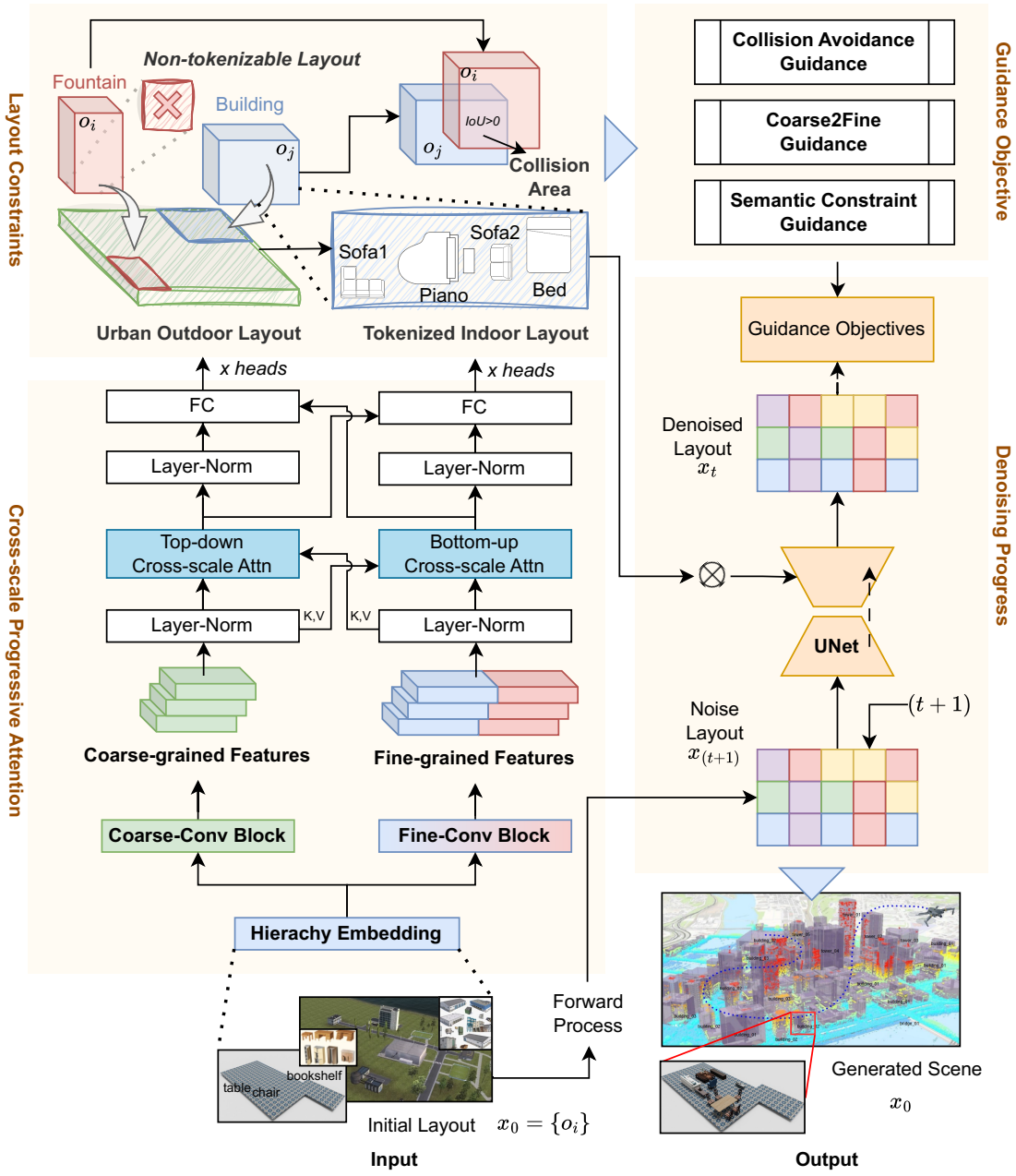


Fig. 2: An overview of our AeroScene method.

where $\mathbf{f}_i^{(0)} = \text{MLP}([\mathbf{p}_i, \mathbf{q}_i, \mathbf{s}_i, \text{Emb}(c_i)])$ encodes geometry and semantics, $\mathbf{e}_i^{\text{pos}}$ is sinusoidal positional encoding [54], and $\mathbf{e}_i^{\text{dom}}$ is a learned indoor/outdoor domain embedding parameterized by a small trainable embedding vector per domain, following domain-adaptive encodings as in [55].

We predict a tokenizability score $\tau_i \in [0, 1]$ for each object at the same timestep:

$$\tau_i = \sigma\left(\mathbf{w}_\tau^\top \text{MLP}(\mathbf{f}_i^{(0)})\right), \tag{5}$$

Then, a coarse- or fine-grained route check is performed on tokens based on their tokenizability scores, by comparing them against a learned gating threshold γ, which determines whether each token is classified as a coarse token $\mathcal{T}_{\text{coarse}}$ or a fine-grained token $\mathcal{T}_{\text{fine}}$.

$$\mathcal{T}_{\text{coarse}} = \{\mathbf{h}_i \mid \tau_i < \gamma\}, \qquad \mathcal{T}_{\text{fine}} = \{\mathbf{h}_i \mid \tau_i \geq \gamma\}. \tag{6}$$

In our implementation γ is a learnable scalar (optimized jointly with network parameters).

C. Coarse and Fine Feature Extraction

Tokens $\mathcal{T}_{\text{coarse}}$ are passed through a lightweight 3D CNN to extract coarse-level features F_{coarse}, which are essential for constructing exteriors (e.g., large-scale structural elements like buildings or terrain):

$$F_{\text{coarse}} = \text{CNN}_{\text{coarse}}(\mathcal{T}_{\text{coarse}}), \quad F_{\text{coarse}} \in \mathbb{R}^{N_c \times d_m}. \tag{7}$$

Tokens $\mathcal{T}_{\text{fine}}$ are used to construct a spatial adjacency graph $G_{\text{fine}} = (V, E)$: each node corresponds to a fine token and edges connect pairs with Euclidean distance $\leq \delta_f$. A two-layer GNN refines these local features:

$$F_{\text{fine}} = \text{GNN}_{\text{fine}}(\mathcal{T}_{\text{fine}}, G_{\text{fine}}), \quad F_{\text{fine}} \in \mathbb{R}^{N_f \times d_m}, \tag{8}$$

with node updates

$$\mathbf{f}_i^{(l+1)} = \text{MLP}\Big(\mathbf{f}_i^{(l)} \,\|\, \sum_{j \in \mathcal{A}_i} \text{MLP}(\mathbf{f}_j^{(l)})\Big). \tag{9}$$

where $\mathcal{A}_i$ denotes the set of neighboring nodes of token i in G_{fine}, and $\|$ indicates vector concatenation.

D. Cross-scale Progressive Attention

Cross-scale Progressive Attention is composed of stacked cross-scale attention blocks that alternate between top-down (coarse→fine) and bottom-up (fine→coarse) interactions:

$$\text{Top-down: } Q = F_{\text{fine}}, \quad K, V = F_{\text{coarse}}, \tag{10}$$

$$\text{Bottom-up: } Q = F_{\text{coarse}}, \quad K, V = F_{\text{fine}}. \tag{11}$$

Each block follows the pattern

$$F' = \text{FC}(\text{Attn}(\text{LN}(Q), \text{LN}(K), \text{LN}(V))) + F, \tag{12}$$

where F is the residual input to the block. By stacking L alternating top-down and bottom-up blocks, coarse tokens propagate structural context while fine tokens inject local detail. The outputs are concatenated and projected to form the final feature:

$$F_{\text{attn}} \in \mathbb{R}^{(N_c+N_f)\times d_m}, \tag{13}$$

which condition the UNet denoiser at each diffusion step via cross-attention layers in the UNet.

E. Guidance Objectives

We ensure the physical plausibility and interactivity of generated scenes by guiding the conditional scene diffusion process with physics-based guidance functions. Additionally, the extensibility of fine-grained layout placement must align with coarse-scale structures while ensuring consistency in object categories and spatial relationships. This motivation led us to implement three guidance objectives:

1) Collision Avoidance Guidance: Penalizes spatial overlap above a small tolerance δ_d using 3D IoU [56]:

$$\mathcal{L}_{\text{col}}(x_t) = \sum_{i \neq j} \max(0, \text{IoU}(B_i, B_j) - \delta_d), \tag{14}$$

where each B_i is an oriented 3D bounding box parameterized by its center $\mathbf{p}_i$, orientation quaternion $\mathbf{q}_i$, and scale $\mathbf{s}_i$, with B_j defined analogously.

2) Coarse-to-Fine Guidance: Encourages fine-grained placement to remain consistent with the coarse-scale structural plan:

$$\mathcal{L}_{\text{c2f}}(x_t) = \sum_{o_i \in \text{fine}} \text{dist}\big(\mathbf{p}_i, \mathcal{R}_{\text{coarse}}(o_i)\big), \tag{15}$$

where $\mathcal{R}_{\text{coarse}}(o_i)$ is the spatial region assigned by the corresponding coarse token (determined via Euclidean distance to coarse centers), following the idea of region decomposition in [21], and $\text{dist}(\cdot,\cdot)$ denotes the Euclidean distance between the object center $\mathbf{p}_i$ and the region's centroid.

3) Semantic Constraint Guidance: Ensures object categories and spatial relations follow learned semantic priors:

$$\mathcal{L}_{\text{sem}}(x_t) = -\sum_{(o_i,o_j)} \log P_{\text{sem}}(c_i, c_j, r_{ij}), \tag{16}$$

where P_{sem} is an MLP estimated from pairwise statistics of the training set and r_{ij} denotes the spatial displacement between objects o_i and o_j.

We define the combined guidance loss as

$$\mathcal{L}_{\text{guide}}(x_t) = \mathcal{L}_{\text{col}}(x_t) + \mathcal{L}_{\text{c2f}}(x_t) + \mathcal{L}_{\text{sem}}(x_t). \tag{17}$$

During training, $\mathcal{L}_{\text{guide}}$ enters the total loss with a small weight (Algorithm 1). During inference, $\nabla_{x_t}\mathcal{L}_{\text{guide}}$ is normalized and scaled before adjusting μ_θ (Algorithm 2). This is depicted in Fig. 2 as a feedback arrow from Guidance Objectives to the denoiser output.

F. Training Algorithm

Training proceeds by sampling a batch of ground-truth layouts and a noise timestep t, forming the noisy input x_t . For each x_t, we compute hierarchy tokens and extract multi-scale conditioning via the coarse/fine branches and the Cross-scale Progressive Attention (Sec. III-D), then predict $\epsilon_\theta(x_t, t)$ and minimize the reconstruction loss

$$\mathcal{L}_{\text{rec}} = \|\epsilon - \epsilon_\theta(x_t, t)\|^2. \tag{18}$$

In parallel, we compute differentiable guidance losses $\mathcal{L}_{\text{guide}}$ on object parameters (using smooth surrogates such as soft IoU, signed distance functions, or soft adjacency). The total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{guide}}\mathcal{L}_{\text{guide}}, \tag{19}$$

where λ_{guide} is a scalar value controlling the influence of guidance. Guidance is applied softly (i.e., small λ_{guide} value) during training to stabilize optimization, while at inference (Algorithm 2), it is always enforced with step-size scaling.

G. Inference Algorithm

Inference performs the reverse diffusion starting from $x_T \sim \mathcal{N}(0, I)$ and iterating $t = T, \ldots, 1$. At each step we compute hierarchy conditioning for the current x_t, predict $\epsilon_\theta(x_t, t)$, and obtain the denoising mean via the epsilon-parameterization

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\Big). \tag{20}$$

Algorithm 1: Training Procedure

Initialize parameters θ (diffusion UNet) and ψ (embedding).
for *each batch* (x_0) *in dataset* **do**
  Sample timestep t and noise ϵ.
  Generate $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$.
  Compute hierarchy tokens for x_t, extract coarse/fine features, and fuse with Cross-scale Progressive Attention (Sec. III-B, III-D).
  Predict noise $\epsilon_\theta(x_t, t)$ and compute reconstruction loss $\mathcal{L}_{\text{rec}}$.
  Compute differentiable guidance loss $\mathcal{L}_{\text{guide}}$ on object parameters.
  Form total loss $\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{guide}}\mathcal{L}_{\text{guide}}$.
  Update θ, ψ by gradient descent on $\mathcal{L}$.

We then evaluate the differentiable guidance loss $\mathcal{L}_{\text{guide}}$ on object parameters, compute its gradient

$$g = \nabla_{x_t} \mathcal{L}_{\text{guide}}(x_t), \tag{21}$$

normalize it $\tilde{g} = g/(\|g\| + \varepsilon)$, and scale it by a timestep-dependent step-size

$$\eta_t = \eta_0 \sqrt{1-\bar{\alpha}_t}. \tag{22}$$

The denoising mean is adjusted as

$$\mu'_\theta = \mu_\theta - \eta_t \tilde{g}, \tag{23}$$

and the next state is sampled

$$x_{t-1} \sim \mathcal{N}(\mu'_\theta, \Sigma_\theta(x_t, t)). \tag{24}$$

After sampling, normalize quaternions $\mathbf{q}_i \leftarrow \mathbf{q}_i/\|\mathbf{q}_i\|$ is applied for each object. Unlike training, guidance is always applied during inference to enforce physical plausibility and semantic consistency.

Algorithm 2: Inference Procedure

Initialize $x_T \sim \mathcal{N}(0, I)$.
for $t = T$ *to* 1 **do**
  Compute hierarchy tokens for x_t, extract coarse/fine features, and fuse with Cross-scale Progressive Attention (Sec. III-B, III-D).
  Predict $\epsilon_\theta(x_t, t)$ and compute the denoising mean $\mu_\theta(x_t, t)$.
  Compute guidance gradient $g = \nabla_{x_t}\mathcal{L}_{\text{guide}}(x_t)$, normalize $\tilde{g}$, and scale with $\eta_t = \eta_0\sqrt{1-\bar{\alpha}_t}$.
  Adjust mean: $\mu'_\theta = \mu_\theta - \eta_t\tilde{g}$.
  Sample $x_{t-1} \sim \mathcal{N}(\mu'_\theta, \Sigma_\theta(x_t, t))$.
Output x_0 as synthesized layout.

IV. Scene Synthesis Experiment

We first compare our scene generation method with recent approaches. In Sec. V, we provide the details of our dataset statistics and its application in drone tasks.

A. Implementation, baseline, and evaluation metrics

Implementation Details. Our method is implemented in PyTorch and validated on the 3D-FRONT [36] and our dataset. Training is performed using the Adam optimizer with a learning rate of 2×10^{-4}, batch size of 64, and a cosine learning rate scheduler. We train for 800 epochs on a single NVIDIA A100 GPU. During training, coarse-to-fine guidance objectives are applied to encourage logical placement of objects across scales, while collision and semantic consistency losses are included to ensure physically plausible and semantically coherent layouts. At inference time, we employ 100 reverse diffusion steps, which provide a balance between synthesis quality and computational efficiency.

Baselines. We compare our framework against established baselines for scene layout synthesis. ATISS [21] employs an autoregressive transformer to generate indoor scenes. Diffusion-SDF [57] uses a diffusion model with signed distance fields to model object placements. DiffuScene [23] is a compositional diffusion model that generates scenes without explicit hierarchical modeling. PhyScene [37] uses physical constraints to generate indoor environments with layouts and articulated objects.

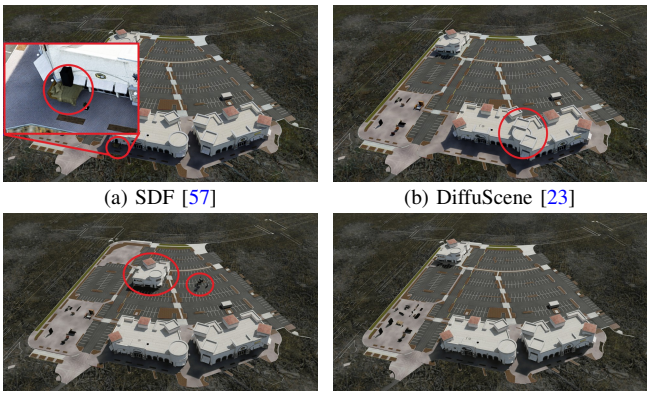
Evaluation Metrics. We evaluate generated layouts using different metrics: FID [58] and KID [59] measure perceptual similarity. Collision Rate (CR) quantifies physical plausibility as the percentage of object pairs with IoU > 0.01. Coarse-to-Fine Consistency (CFC) evaluates hierarchical alignment by computing the average normalized distance between fine placements and their assigned coarse regions. Semantic Plausibility (SP) measures the similarity with spatial category priors, calculated as the negative log-likelihood under empirical pairwise category distributions from the training set. Metrics are averaged over 1,000 scenes per method.

Method	FID↓	KID↓	CR(%)↓	CFC↓	SP↓
Our Dataset					
ATISS [21]	45.2	0.032	12.5	0.21	3.8
Diffusion-SDF [57]	38.7	0.028	10.1	0.18	3.5
DiffuScene [23]	32.4	0.025	8.3	0.15	3.2
PhyScene [37]	29.8	0.023	7.1	0.13	3.0
Ours	**27.3**	**0.021**	**6.2**	**0.12**	**2.7**
3D-FRONT Dataset [36]					
ATISS [21]	42.1	0.030	11.8	0.19	3.6
Diffusion-SDF [57]	35.6	0.026	9.4	0.16	3.3
DiffuScene [23]	30.2	0.023	7.6	0.14	3.0
PhyScene [37]	27.9	0.021	6.3	0.12	2.7
Ours	**25.8**	**0.019**	**5.5**	**0.11**	**2.5**

TABLE II: Quantitative results on scene synthesis.

B. Scene Generation Results

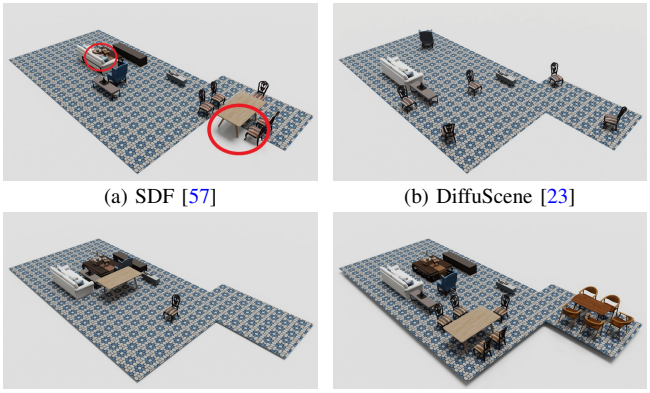
Table II shows quantitative results of our method. This table shows that our method outperforms baselines in all metrics. Qualitatively, Fig. 3 illustrates that the generated layouts in our model produce coherent hierarchies, grouping fine objects (e.g., tables, chairs, sofas) within coarse structures (e.g., room partitions), whereas baselines often



(a) SDF [57] (b) DiffuScene [23]

(c) PhyScene [37] (d) Ours

Fig. 3: Outdoor scene generation visual comparison. The red circle shows the collision or incorrect position.



(a) SDF [57] (b) DiffuScene [23]

(c) PhyScene [37] (d) Ours

Fig. 4: Indoor scene generation visual comparison. The red circle shows the collision or incorrect position.

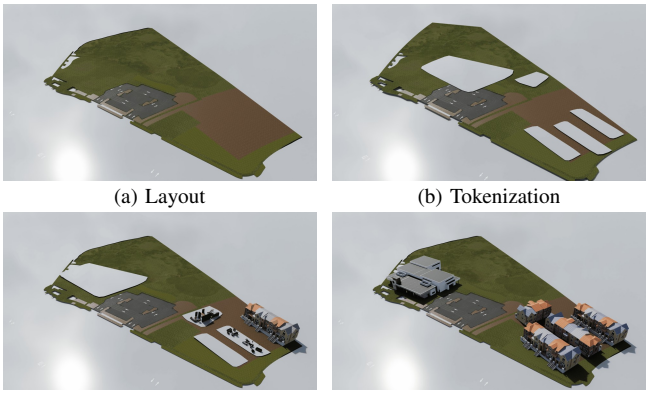
exhibit overlaps or implausible placements. On the 3D-FRONT dataset, similar trends hold on both qualitative and quantitative evaluations. Our method yields more realistic indoor arrangements compared with other solutions (Fig. 4). We also visualize the generation progress of our guided substitution for diffirent objects in Fig. 5.

C. Ablation Study on Guidance

To assess the impact of guidance objectives, we conduct ablations by removing individual components during training and inference. Table III shows results on our dataset. Removing Collision Avoidance increases CR by 40%, indicating its role in physical plausibility. Without Coarse-to-Fine Guidance, CFC degrades by 25%, leading to misaligned hierarchies. Semantic Constraint Guidance is crucial for SP, with its removal causing a 30% worsening. Combining all guidance yields the best performance, confirming their complementary nature. Note that our inference time is approximately 2 minutes per scene on an NVIDIA A100 GPU, comparable to other diffusion baselines.

Configuration	FID↓	CR(%)↓	CFC↓	SP↓
Ours (full)	**27.3**	**6.2**	**0.12**	**2.7**
w/o Collision	32.1	8.7	0.13	2.8
w/o C2F	30.5	6.5	0.15	2.9
w/o Semantic	31.8	6.4	0.13	3.5
w/o All Guidance	35.4	9.2	0.17	3.9

TABLE III: Ablation on guidance objectives.



(a) Layout (b) Tokenization

(c) Generation w. Constraints (d) Finalization

Fig. 5: The generation sequence of objects in our method.

V. AeroScene Dataset for Aerial Robotic Tasks

A. AeroScene Dataset Statistic and Labels

Using our proposed method, we create the AeroScene dataset with more than 1000 scenes, which are specifically curated to emphasize multi-scale hierarchical structures in indoor and outdoor environments. All scenes are embedded into NVIDIA Isaac Sim [11] to facilitate downstream drone-related tasks. Scene creation began with initializing empty indoor (e.g., rooms, offices) and outdoor (e.g., parks, urban areas) settings, where base layouts were formed by importing pre-built assets such as walls, floors, and terrain. Objects were then placed in a hierarchical manner: coarse-scale structural elements like buildings, walls, or terrains were positioned first to define the overall structure, followed by fine-scale details such as utensils, decorations, or debris, which were arranged relative to the larger elements to preserve logical groupings and spatial relationships. Finally, all objects were annotated with bounding boxes, orientations, scales, and semantic categories, while hierarchies explicitly linked fine objects to their coarse parents. In addition, interaction areas suitable for drone landing were annotated to support downstream robotics tasks. Specifically, all scenes are stored within a consistent schema: each object has `id`, `scene_id`, `parent_id`, `category_id` $\in \{1, \ldots, C\}$, `bbox` given as $(\mathbf{p}, \mathbf{q}, \mathbf{s})$ where $\mathbf{p} \in \mathbb{R}^3$ (center), $\mathbf{q} \in \mathbb{R}^4$ (quaternion), $\mathbf{s} \in \mathbb{R}^3$ (local extents), plus precomputed `bbox_corners` for fast IoU tests, a `domain` flag (indoor/outdoor), and optional `attributes`; interaction areas (e.g., landing zones) are stored as polygonal regions with centroid and radius. The dataset statistical information can be found in Table IV.

Criteria	Train	Test	Total
#Scenes	812	204	1016
#Objects	122,356	37,654	160,010
Avg. objects/scene	149	152	149
Avg. objs / coarse-scale level	35	32	34
Avg. objs / fine-scale level	111	114	112
Avg. Landing Areas for Drones	55	53	54
Small Drone (Avg.)	42	43	42
Medium Drone(Avg.)	11	13	12
Large Drone(Avg.)	4	4	5
#Categories (coarse-scale)	23		
#Categories (fine-scale)	47		

TABLE IV: Statistics of our dataset.

B. AeroScene for Navigation and Interaction Tasks

To demonstrate the practical use of the AeroScene dataset, we define a unified aerial robotics task that combines *long-range navigation* and *close-range physical interaction* within a single scene. In this setup, an aerial robot starts at a designated location and must autonomously navigate to a *pre-annotated interaction area* before performing a controlled landing or perching maneuver. This task demonstrates how AeroScene's realistic and richly annotated environments can evaluate both high-level planning and fine-grained physical interaction capabilities under diverse conditions.

Task Design. The drone task is divided into two sequential phases: *(i) Navigation Phase:* The drone uses global semantic information and dynamics constraints to plan a trajectory to the target area [60]. A geometric controller [61] executes this path, ensuring stable navigation through complex environments. *(ii) Interaction Phase:* Once near the target, the drone switches to local sensing. Point-cloud data is analyzed to identify surface normals, slope, and clearance, allowing the system to select a safe landing or perching zone. The drone then performs a precise descent and touchdown.

Example Scenario. Fig. 6 illustrates a representative mission: navigating a generated urban-style scene to land on the red roof of the Weefit building in the scene. In this example, the environment contains detailed objects and varying elevations, requiring the drone to plan a path and then transition to close-proximity perception for accurate landing. This scenario, combined with AeroScene's dataset generation pipeline, demonstrates the usefulness of our work in benchmarking aerial robotics algorithms in perception, mapping, trajectory planning, and interaction control.

Scene Utility and Data Generation. AeroScene's hierarchical object labeling, annotated landing zones, and physics-aware surfaces create challenging, realistic environments for aerial autonomy and manipulation research. Each trial not only evaluates planning and control strategies but also generates a rich dataset for future development. For every trajectory, we record:

- **Visual Data:** RGB and depth streams from simulated onboard cameras.
- **State Estimates:** Ground-truth absolute position, orientation, and velocity.

Metric	Value
Drone Platforms	2
Trajectories per Scene	300
Overall Success Rate	**91%**

TABLE V: Evaluation summary of navigation and interaction tasks on the AeroScene dataset.

Fig. 6: Generated navigation and interaction trajectories for the example mission: landing on the red roof of the Weefit building. Green lines represent feasible navigation and interaction trajectories, and red lines denote failed attempts. Sampled point clouds are displayed within the blue box, with red dots indicating failure landing points.

- **Inertial Measurements:** IMU sensor readings for accelerations and angular rates.
- **Control Signals:** Low-level motor or actuator commands used during trajectory execution.
- **Planned and Executed Trajectories:** Waypoints and actual flight paths for benchmarking performance.

In total, we record each scene 300 planned trajectories for each scene. Small- and medium-sized drone platforms, i.e., 3DR Iris and AscTec Hummingbird, were used to test the unified navigation and interaction pipeline. Across all trials, the system achieved an overall success rate of 91%, demonstrating the utility of AeroScene as a challenging yet tractable benchmark for aerial robotics research. Details are summarized in Table V.

VI. Discussion and Conclusion

Limitations. The proposed AeroScene, while effective in demonstrating hierarchical scene synthesis, has certain limitations. Current experiments are mainly conducted in simulation, which may not fully represent the diversity and complexity of real-world aerial environments (such as wind conditions). In addition, the framework focuses on static scene layouts, without explicitly modeling temporal dynamics or handling uncertainty from dynamic environments. Therefore, an interesting future work is to extend the synthesis process to generate dynamic elements. Furthermore, performing sim-to-real validation on real aerial robots would be an interesting direction for future work.

Conclusion. We introduced AeroScene, a hierarchical diffusion framework for 3D scene synthesis that combines hierarchy-scale tokenization, multi-branch feature extraction, and a cross-scale attention with gradient-based guidance objectives. Our approach enables structured reasoning across global layouts and local details while enforcing physical and semantic plausibility. Using our method, we generate a large-scale benchmark of 3D environments for drone interaction. Our code and dataset are publicly available at aioz-ai.github.io/AeroScene/.

REFERENCES

[1] B. Sandikci and I. Colak, "Autonomous drone for room exploration and 3d reconstruction," in *SmartNets*, 2025.

[2] S. Cascarano, M. Milazzo, A. Vannini, A. Spezzaneve, and S. Roccella, "Design and development of drones to autonomously interact with objects in unstructured outdoor scenarios," *Field Robotics*, 2021.

[3] Y. Fan, W. Chen, T. Jiang, C. Zhou, Y. Zhang, and X. E. Wang, "Aerial vision-and-dialog navigation," *arXiv*, 2022.

[4] Y. Liu, M. Zhao, K. Hou, J. Xia, C. Carver, S. Xia, X. Zhou, and X. Jiang, "Aira: A low-cost ir-based approach towards autonomous precision drone landing and nlos indoor navigation," *arXiv*, 2024.

[5] K. Pluckter and S. Scherer, "Precision uav landing in unstructured environments," in *ISER*, 2018.

[6] F. Furrer, M. Burri, and M. Achtelik, *RotorS—A modular gazebo MAV simulator framework*, 2016.

[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *FSR*, 2017.

[8] M. Nikolaiev and M. Novotarskyi, "Comparative review of drone simulators," *Information, Computing and Intelligent systems*, 2024.

[9] M. Sabet, P. Palanisamy, and S. Mishra, "Scalable modular synthetic data generation for advancing aerial autonomy," *RA-S*, 2023.

[10] C. A. Dimmig, G. Silano, K. McGuire, C. Gabellieri, W. Hšnig, J. Moore, and M. Kobilarov, "Survey of simulators for aerial robots: An overview and in-depth systematic comparisons," *RA-M*, 2024.

[11] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv*, 2021.

[12] B. Xu, F. Gao, C. Yu, R. Zhang, Y. Wu, and Y. Wang, "Omnidrones: An efficient and flexible platform for reinforcement learning in drone control," *RA-L*, 2024.

[13] Z. Huang, S. Batra, T. Chen, R. Krupani, T. Kumar, A. Molchanov, A. Petrenko, J. A. Preiss, Z. Yang, and G. S. Sukhatme, "Quadswarm: A modular multi-quadrotor simulator for deep reinforcement learning with direct thrust control," *arXiv*, 2023.

[14] F. Li, F. Sun, T. Zhang, and D. Zou, "Visfly: An efficient and versatile simulator for training vision-based flight," *arXiv*, 2024.

[15] J. Du, K. Wang, Y. Fan, G. Lai, and Y. Yu, "High-fidelity integrated aerial platform simulation for control, perception, and learning," *IEEE Transactions on Automation Science and Engineering*, 2025.

[16] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Automatic furniture layout with a single image," in *IEEE ICCV*, 2017.

[17] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan, "Synthesizing open worlds with constraints using locally annealed reversible jump mcmc," in *ToG*, 2012.

[18] S.-H. Zhang, Z. Zhang, J. Wu, S. Tulsiani, and A. X. Chang, "Learning generative models of scene graphs," in *NIPS*, 2020.

[19] C. H. Lin, H.-Y. Lee, W. Menapace, M.-H. Yang, and S. Tulyakov, "Infinicity: Infinite-scale city synthesis," in *ICCV*, 2023.

[20] H. Xie, Z. Chen, F. Hong, and Z. Liu, "Citydreamer: Compositional generative model of unbounded 3d cities," in *CVPR*, 2024.

[21] D. Paschalidou, A. Kar, M. Shugrina, A. Geiger, and S. Fidler, "Atiss: Autoregressive transformers for indoor scene synthesis," *NIPS*, 2021.

[22] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling, "Equivariant diffusion for molecule generation in 3d," in *ICLR*, 2022.

[23] J. Tang, Y. Nie, and M. Nießner, "Diffuscene: Denoising diffusion models for generative indoor scene synthesis," in *CVPR*, 2024.

[24] A. D. Vuong, M. N. Vu, T. Nguyen, B. Huang, D. Nguyen, T. Vo, and A. Nguyen, "Language-driven scene synthesis using multi-conditional diffusion model," *NeurIPS*, 2023.

[25] A. Bokhovkin, Q. Meng, and A. Dai, "Scenefactor: Factored latent 3d diffusion for controllable 3d scene generation," in *CVPR*, 2025.

[26] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *FSR*, 2018.

[27] R. Madaan, H. Zhu, D. Hsu, and W. S. Lee, "Airs: Aerial indoor robot simulation for navigation," in *ICRA*, 2020.

[28] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," in *ICLRW*, 2018.

[29] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IROS*, 2015.

[30] Z. Wu, L. Song, Shuranand Zhang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.

[31] X. Ren, J. Huang, S. Fidler, and F. Williams, "Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies," in *CVPR*, 2024.

[32] C. Lin and Y. Mu, "Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior," in *ICLR*, 2024.

[33] H.-H. Lee, Q. Han, and A. X. Chang, "Nuiscene: Exploring efficient generation of unbounded outdoor scenes," *arXiv*, 2025.

[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[35] C. R. Qi, L. Yi, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.

[36] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao, *et al.*, "3d-front: 3d furnished rooms with layouts and semantics," in *ICCV*, 2021.

[37] Y. Yang, B. Jia, P. Zhi, and S. Huang, "Physcene: Physically interactable 3d scene synthesis for embodied ai," in *CVPR*, 2024.

[38] K. Yamazaki, T. Hanyu, K. Vo, T. Pham, M. Tran, G. Doretto, A. Nguyen, and N. Le, "Open-fusion: Real-time open-vocabulary 3d mapping and queryable scene representation," in *ICRA*, 2024.

[39] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi, "Procthor: Large-scale embodied ai using procedural generation," *NIPS*, 2022.

[40] S. Lee and H. Kim, "Dynscene: Scalable generation of dynamic robotic manipulation scenes for embodied ai," in *CVPR*, 2025.

[41] Y. Wang, X. Qiu, J. Liu, Z. Chen, J. Cai, Y. Wang, T.-H. Wang, Z. Xian, and C. Gan, "Architect: Generating vivid and interactive 3d scenes with hierarchical 2d inpainting," *NIPS*, 2024.

[42] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *NIPS*, 2021.

[43] N. Nguyen, M. N. Vu, B. Huang, A. Vuong, N. Le, T. Vo, and A. Nguyen, "Lightweight language-driven grasp detection using conditional consistency model," in *IROS*, 2024.

[44] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *arXiv*, 2022.

[45] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *NIPS*, 2022.

[46] C. Meng, J. Ho, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," in *ICLR*, 2023.

[47] X. Liu, Z. Li, Y. Song, and S. Ermon, "Compositional visual generation with energy-based diffusion," in *NIPS*, 2022.

[48] X. Jiang, F. Yang, W. Xu, and B. Chen, "Motion guidance for human-scene interaction synthesis with diffusion models," in *ToG*, 2023.

[49] N. Le, T. Do, K. Do, H. Nguyen, E. Tjiputra, Q. D. Tran, and A. Nguyen, "Controllable group choreography using contrastive diffusion," *TOG*, 2023.

[50] A. Jain, B. Zhang, B. Poole, and P. Abbeel, "Zero-1-to-3: Controllable object synthesis with diffusion," in *NIPS*, 2022.

[51] T. Nguyen, M. N. Vu, B. Huang, A. Vuong, Q. Vuong, N. Le, T. Vo, and A. Nguyen, "Language-driven 6-dof grasp detection using negative prompt guidance," in *ECCV*, 2024.

[52] J. Ni, Y. Chen, B. Jing, N. Jiang, S.-C. Zhu, and S. Huang, "Phyrecon: Physically plausible neural scene reconstruction," *NIPS*, 2024.

[53] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NIPS*, 2020.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NIPS*, 2017.

[55] H. Chen, F. Wei, B. Ni, J. Bao, D. Zhang, D. Chen, and B. Guo, "Vision transformer adapter for dense predictions," in *ICLR*, 2022.

[56] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "Iou loss for 2d/3d object detection," in *3DV*, 2019.

[57] G. Chou, Y. Bahat, and F. Heide, "Diffusion-sdf: Conditional generative modeling of signed distance functions," in *ICCV*, 2023.

[58] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *NIPS*, 2017.

[59] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," *arXiv*, 2018.

[60] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *Transactions on Robotics*, 2015.

[61] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *CDC*, 2010.